ELVIS

- a clone of vi/ex - version 1.8

In Finnland also known by the name **Melvis Pressula**

Compiled to Windows Help file by

Timo Kinnunen
Särkiniementie 16 A 41
70700 Kuopio
Finland

I have compiled this manual for those, who are using other operating systems than Microsoft Windows. You cannot run all of Windows programs in other operating systems - but Elvis is the program which can be transferred to most of them. For Windows user Elvis may look like empty table, which has no hints on the screen how to use it. It lacks those conveniences, and rows of buttons for mouse. But you must read this manual first, and then critisize Elvis. Whatever you might say, there is an inevitavle truth that there has been a lot of work when developing Elvis, and that it is very complicated program. As you see, I have not translated this into Finnish, and there is a good reason for it. Namely, that those who are going to use Elvis, or whoi are familiar to it, can pretty well do it without Finnish manual, and Elvis is not very young any more. But Elvis is not dead, either.

Hurry down doomsday, the bugs are taking over! - **E.C**.

Author:
Steve Kirkendall
14407 SW Teal Blvd., Apt C
Beaverton, OR 97005

E-Mail: kirkenda@cs.pdx.edu
Phone: (503) 643-6980

Table of Contents

VISUAL MODE COMMANDS *Normal interactive editing, Input mode, Arrow keys, Digraphs, Abbreviations, Auto-indentation*

COLON MODE COMMANDS *Line specifiers, Text entry, Cut & paste, Display text, Global operations, Line editing, Undo, Configuration & status, Multiple files, Switching files, Working with a compiler, Exiting, File I/O, Directory & shell, Debugging*

REGULAR EXPRESSIONS *Syntax, Options, Substitutions, Examples*

OPTIONS *Autoindent, Autoprint, etc*.

INITIALIZATION *Start-up initialization, File initialization, The :mkexrc command, Other techniques*

CUT BUFFERS *Putting text into a cut buffer, Pasting from a cut buffer, Macros, The effect of switching files*

DIFFERENCES BETWEEN Elvis AND THE REAL VI/EX *Extensions, Omissions*

INTERNAL *For programmers only, The temporary file, Implementation of editing, Marks and the cursor, Colon command interpretation, Screen control, Portability*

MAKEFILE

CFLAGS

TERMCAP

ENVIRONMENT VARIABLES

VERSIONS

QUESTIONS & ANSWERS

ELVIS (1) ELVIS (1)

ELVPRSV

ELVREC

FMT

REF

1. INTRODUCTION

Elvis is a clone of vi/ex, the standard UNIX editor. Elvis supports nearly all of the vi/ex commands, in both visual mode and colon mode.

Like vi/ex, Elvis stores most of the text in a temporary file, instead of RAM. This allows it to edit files that are too large to fit in a single process' data space. Also, the edit buffer can survive a power failure or crash.

Elvis runs under BSD UNIX, AT&T SysV UNIX, Minix, MS-DOS, Atari TOS, Coherent, OS9/68000, VMS, AmigaDos, and OS/2. The next version is also expected to add MS-Windows and MacOS. Contact me before you start porting it to some other OS, because somebody else may have already done it for you.

Elvis is freely redistributable, in either source form or executable form. There are no restrictions on how you may use it.

1.1 Compiling

See the "Versions" section of this manual for instructions on how to compile Elvis. If you want to port Elvis to another O.S. or compiler, then you should start by reading the "Portability" part of the "Internal" section.

1.2 Overview of Elvis

The user interface of Elvis/vi/ex is weird. There are two major command modes in Elvis, and a few text input modes as well. Each command mode has a command which allows you to switch to the other mode. You will probably use the visual command mode most of the time. This is the mode that Elvis normally starts up in. In visual command mode, the entire screen is filled with lines of text from your file. Each keystroke is interpreted as part of a visual command. If you start typing text, it will not be inserted, it will be treated as part of a command. To insert text, you must first give an "insert text" command. This will take some getting used to. (An alternative exists. Lookup the "input mode" option.)

The colon mode is quite different. Elvis displays a ":" character on the bottom line of the screen, as a prompt. You are then expected to type in a command line and hit the *Return* key. The set of commands recognized in the colon mode is different from visual mode's.

2. VISUAL MODE COMMANDS

Most visual mode commands are one keystroke long. The following table lists the operation performed by each keystroke, and also denotes any options or arguments that it accepts. Notes at the end of the table describe the notation used in this table.

In addition to the keys listed here, your keyboard's "arrow" keys will be interpreted as the appropriate cursor movement commands. The same goes for *PgUp* and *PgDn*, if your keyboard has them. The *Insert* key will toggle between insert mode and replace mode. There is a colon mode command (":map", to be described later) which will allow you to define other keys, such as function keys.

A tip: visual command mode looks a lot like text input mode. If you forget which mode you're in, just hit the *Esc* key. If Elvis beeps, then you're in visual command mode. If Elvis does not beep, then you were in input mode, but by hitting *Esc* you will have switched to visual command mode. So, one way or another, after *Esc* Elvis will be ready for a command.

COMMAND DESCRIPTION

^A Search for next occurrence of word at cursor (MOVE)(EXT)
^B Move toward the top of the file by 1 screenful
^C --- (usually sends SIGINT, to interrupt a command)
count ^D Scroll down [count] lines (default 1/2 screen)
count ^E Scroll up [count] lines
^F Move toward the bottom of the file by 1 screenful
^G Show file status, and the current line #
count ^H Move left, like h (MOVE)
^I ---
count ^J Move down (MOVE)
^K ---
^L Redraw the screen
count ^M Move to the front of the next line (MOVE)
count ^N Move down (MOVE)
^O ---
count ^P Move up (MOVE)
^Q --- (typically XON, which restarts screen updates)
^R Redraw the screen
^S --- (typically XOFF, which stops screen updates)
^T Return to source of previous :tag or ^] command.
count ^U Scroll up [count] lines (default 1/2 screen)
^V ---
^W ---
count ^X Move to a physical column number on the screen (MOVE) (EXT)
count ^Y Scroll down [count] lines
^Z --- (sometimes sends SIGSUSP, to suspend execution)
ESC ---
^\ --- (usually sends SIGQUIT, which is ignored)
^] If the cursor is on a tag name, go to that tag
^^ Switch to the previous file, like ":e #"
count ^_ Move to a given screen-relative row (MOVE) (EXT)

count SPC Move right,like l (MOVE)
! mv Run the selected lines thru an external filter program
" key Select which cut buffer to use next
count # + Increment a number (EDIT) (EXT)
$Move to the rear of the current line (MOVE)
count %Move to matching (){}[] or to a given % of file (MOVE) (EXT)
count &Repeat the previous ":s//" command here (EDIT)
' key Move to a marked line (MOVE)
count (Move backward [count] sentences (MOVE)
count)Move forward [count] sentences (MOVE)
 *Go to the next error in the errlist (EXT)
count +Move to the front of the next line (MOVE)
count ,Repeat the previous [fFtT] but in the other direction (MOVE)
count -Move to the front of the preceding line (MOVE)
count .Repeat the previous "edit" command
 / textSearch forward for a given regular expression (MOVE)
 0If not part of count, move to 1st char of this line (MOVE)
 1Part of count
 2Part of count
 3Part of count
 4Part of count
 5Part of count
 6Part of count
 7Part of count
 8Part of count
 9Part of count
 : textRun single EX cmd
count ;Repeat the previous [fFtT] cmd (MOVE)
 < mv Shift text left (EDIT)
= mv Reformat
 > mv Shift text right (EDIT)
? textSearch backward for a given regular expression (MOVE)
@ key Execute the contents of a cut-buffer as VI commands
count A inp Append at end of the line (EDIT)
count BMove back Word (MOVE)
C inp Change text from the cursor through the end of the line (EDIT)
DDelete text from the cursor through the end of the line (EDIT)
count EMove end of Word (MOVE)
count F key Move leftward to a given character (MOVE)
count GMove to line #[count] (default is the bottom line) (MOVE)
count HMove to home row (the line at the top of the screen)
count I inp Insert at the front of the line (after indents) (EDIT)
count JJoin lines, to form one big line (EDIT)
KLook up keyword (EXT)
count LMove to last row (the line at the bottom of the screen)
MMove to middle row
NRepeat previous search, but in the opposite direction (MOVE)

count O inp Open up a new line above the current line (EDIT)
PPaste text before the cursor (EDIT)
QQuit to EX mode
R inp Overtype (EDIT)
count S inp Change lines, like [count]cc
count T key Move leftward *almost* to a given character (MOVE)
UUndo all recent changes to the current line
VStart marking lines for c/d/y/ </ >/!/\ (EXT)
count WMove forward [count] Words (MOVE)
count XDelete the character(s) to the left of the cursor (EDIT)
count YYank text line(s) (copy them into a cut buffer)
Z Z Save the file & exit
[[Move back 1 section (MOVE)
\ mv Pop-up menu for modifying text (EXT)
]] Move forward 1 section (MOVE)
^Move to the front of the current line (after indent) (MOVE)
count _Move to the current line
` key Move to a marked character (MOVE)
count a inp Insert text after the cursor (EDIT)
count bMove back [count] words (MOVE)
c mv Change text (EDIT)
d mv Delete text (EDIT)
count eMove forward to the end of the current word (MOVE)
count f key Move rightward to a given character (MOVE)
g---
count hMove left (MOVE)
count i inp Insert text at the cursor (EDIT)
count jMove down (MOVE)
count kMove up (MOVE)
count lMove right (MOVE)
m key Mark a line or character
nRepeat the previous search (MOVE)
count o inp Open a new line below the current line (EDIT)
pPaste text after the cursor (EDIT)
q---
count r key Replace [count] chars by a given character (EDIT)
count s inp Replace [count] chars with text from the user (EDIT)
count t key Move rightward *almost* to a given character (MOVE)
uUndo the previous edit command
vStart marking characters for c/d/y/ </ >/!/\ (EXT)
count wMove forward [count] words (MOVE)
count xDelete the character that the cursor's on (EDIT)
y mv Yank text (copy it into a cut buffer)
z key Scroll current line to the screen's +=top -=bottom .=middle
count {Move back [count] paragraphs (MOVE)
count |Move to column [count] (the leftmost column is 1)
count }Move forward [count] paragraphs (MOVE)

count ~Switch a character between uppercase & lowercase (EDIT)
DEL --- (usually mapped to shift-X, so it deletes one character)

count

Many commands may be preceded by a count. This is a sequence of digits representing a decimal number. For most commands that use a count, the command is repeated [count] times. The count is always optional, and usually defaults to 1.

key

Some commands require two keystrokes. The first key always determines which command is to be executed. The second key is used as a parameter to the command.

mv

Some commands (! < > c d y \ =) operate on text between the cursor and some other position. There are three ways that you can specify that other position.

The first way is to follow the command keystroke with a movement command. For example, "dw" deletes a single word. "d3w" and "3dw" both delete three words.

The second way is to type the command keystroke twice. This causes whole lines to be acted upon. For example, " > >" indents the current line. "3 > >" indents the current line and the following two lines.

The last way is to move the cursor to one end of the text, type 'v' or 'V' to start marking, move the cursor to the other end, and then type the desired command key.

inp

Many commands allow the user to interactively enter text. See the discussion of "input mode" in the following section.

(EXT)

These commands are extensions -- the real vi doesn't have them.

(EDIT)

These commands affect text, and may be repeated by the "." command.

(MOVE)

These commands move the cursor, and may be used to specify the extent of a member of the "mv" class of commands.

2.1 Input Mode

You can't type text into your file directly from visual command mode. Instead, you must first give a command which will put you into input mode. The commands to do this are A/C/I/O/R/S/a/i/o/s.

The S/s/C/c commands temporarily place a $ at the end of the text that they are going to change.

In input mode, all keystrokes are inserted into the text at the cursor's position, except for the following:

^Ainsert a copy of the last input text
^Ddelete one indent character
^H(backspace) erase the character before the cursor
^Lredraw the screen
^M(carriage return) insert a newline (^J, linefeed)
^Oexecute next key as a visual command (limited!)
^Pinsert the contents of the cut buffer
^Rredraw the screen, like ^L
^Tinsert an indent character
^Ubackspace to the beginning of the line
^Vinsert the following keystroke, even if special
^Wbackspace to the beginning of the current word
^Z^Zwrite the file & exit Elvis
^[(ESCape) exit from input mode, back to command mode

Also, on some systems, ^S may stop output, ^Q may restart output, and ^C may interrupt execution. ^@ (the NUL character) cannot be inserted.

The R visual command puts you in overtype mode, which is a slightly different form of input mode. In overtype mode, each time you insert a character, one of the old characters is deleted from the file.

2.2 Arrow keys in Input Mode

The arrow keys can be used to move the cursor in input mode. (This is an extension; the real Vi doesn't support arrow keys in input mode.) The *PgUp*, *PgDn*, *Home*, and *End* keys work in input mode, too. The *Delete* key deletes a single character in input mode. The *Insert* key toggles between input mode and replace mode.

The best thing about allowing arrow keys to work in input mode is that as long as you're in input mode, Elvis seems to have a fairly ordinary user interface. With most other text editors, you are always in either insert mode or replace mode, and you can use the arrow keys at any time to move the cursor.Now, Elvis can act like that, too. In fact, with the

new "inputmode" option and the "control-Z control-Z" input command, you may never have to go into visual command mode for simple edit sessions.

2.3 Digraphs

Elvis supports digraphs as a way to enter non-ASCII characters. A digraph is a character which is composed of two other characters. For example, an apostrophe and the letter i could be defined as a digraph which is to be stored & displayed as an accented i.

There is no single standard for extended ASCII character sets. Elvis can be compiled to fill the digraph with values appropriate for either the IBM PC character set, or the LATIN-1 character set used by X windows, or neither.(See the discussions of -DCS_IBMPC and -DCS_LATIN1 in the CFLAGS section of this manual.) You can view or edit the digraph table via the ":digraph" colon command.

Digraphs will not be recognized until you've entered ":set digraph".

To actually use a digraph type the first character, then hit *Backspace*, and then type the second character. Elvis will then substitute the non-ASCII character in their place.

2.4 Abbreviations

Elvis can expand abbreviations for you. You define an abbreviation with the :abbr command, and then whenever you type in the abbreviated form while in input mode, Elvis will immediately replace it with the long form. COBOL programmers should find this useful. :-)

Elvis doesn't perform the substitution until you type a nonalphanumeric character to mark the end of the word. If you type a control-V before that non-alphanumeric character, then Elvis will not perform the substitution.

2.5 Auto-Indent

With the ":set autoindent" option turned on, Elvis will automatically insert leading whitespace at the beginning of each new line that you type in. The leading whitespace is copied from the preceding line.

To add more leading whitespace, type control-T.To remove some whitespace, type control-D.

If you ":set noautotab", then the whitespace generated by control-T will always consist of spaces -- never tabs. Some people seem to prefer this.

Elvis' autoindent mode isn't 100% compatible with vi's. In Elvis, 0^D and ^^D don't work, ^U can wipeout all indentation, and sometimes Elvis will use a different amount of indentation than vi would.

3. COLON MODE COMMANDS

To use colon mode commands, you must switch from visual command mode to colon command mode. The visual mode commands to do this are ":" for a single colon command, or "Q" for many colon mode commands.

In general, command lines begin with 0, 1, or 2 line specifiers, followed by a command name, and perhaps some arguments after that.

Lines which don't access the text, such as ":quit", don't allow any line specifiers. Other commands, such as ":mark", only allow a single line specifier. Most commands, though, allow two line specifiers; the command is applied to all lines between the two specified lines, inclusive. The table below indicates how many line specifiers each command allows.

Command names can usually be abbreviated; in the table below, the extra part of command names has is enclosed in square brackets.Square brackets are also used to indicate which arguments are optional.

LINES COMMAND ARGUMENTS

ab[br] [short] [expanded form]
an[d]condition
[line] a[ppend][!]
ar[gs] [files]
cc[files]
cd[!][directory]
[line][,line] c[hange]
chd[ir][!][directory]
[line][,line] co[py] line
col[or][when] [[light] color] [on color]
[line][,line] d[elete] [x]
dig[raph][!][XX [Y]]
e[dit][!] [file]
el[se] commands
er[rlist][!][errlist]
f[ile] [file]
[line][,line] g[lobal] /regexp/ command
ifcondition
[line] i[nsert]

[line][,line] j[oin][!]
[line][,line] l[ist]
mak[e] [target]
map[!] key mapped_to
[line] ma[rk] x
mk[exrc]
[line][,line] m[ove] line
n[ext][!] [files]
N[ext][!]
[line][,line] nu[mber]
o[r] condition
po[p][!]
[line][,line] p[rint]
[line] pu[t][x]
q[uit][!]
[line] r[ead] file
rew[ind][!]
se[t][options]
so[urce] file
[line][,line] s[ubstitute]/regexp/replacement/[p][g][c]
[line][,line] t line
ta[g][!] tagname
th[en] commands
una[bbr] [short]
u[ndo]
unm[ap][!]key
ve[rsion]
[line][,line] v[global] /regexp/ command
vi[sual] [filename]
wq
[line][,line] w[rite][!][[> >]file]
x[it][!]
[line][,line] y[ank] [x]
[line][,line] ! command
[line][,line] <
[line][,line] =
[line][,line] >
[line][,line] &
@x

3.1 Line Specifiers

Line specifiers are always optional. The first line specifier of most commands usually defaults to the current line. The second line specifier usually defaults to be the same as the first line specifier. Exceptions are :write, :global, and :vglobal, which act on all lines

of the file by default, and :!, which acts on no lines by default.

If you use the visual V command to mark a range of lines, and then use the visual : command to execute a single ex command, then the default range affected by the ex command will be the visibly marked text.

Line specifiers consist of an absolute part and a relative part.The absolute part of a line specifier may be either an explicit line number, a mark, a dot to denote the current line, a dollar sign to denote the last line of the file, or a forward or backward search.

An explicit line number is simply a decimal number, expressed as a string of digits.

A mark is typed in as an apostrophe followed by a letter. Marks must be set before they can be used. You can set a mark in visual command mode by typing "m" and a letter, or you can set it in colon command mode via the "mark" command.

A forward search is typed in as a regular expression surrounded by slash characters; searching begins at the default line. A backward search is typed in as a regular expression surrounded by question marks; searching begins at the line before the default line.

If you omit the absolute part, then the default line is used.

The relative part of a line specifier is typed as a "+" or "-" character followed by a decimal number.The number is added to or subtracted from the absolute part of the line specifier to produce the final line number.

As a special case, the % character may be used to specify all lines of the file. It is roughly equivalent to saying 1,$. This can be a handy shortcut.

Some examples:

:p print the current line
:37p print line 37
:'gp print the line which contains mark g
:/foo/p print the next line that contains "foo"
:$pprint the last line of the file
:20,30p print lines 20 through 30
:1,$pprint all lines of the file
:%pprint all lines of the file
:/foo/-2,+4p print 5 lines around the next "foo"

3.2 Text Entry Commands

[line] append
[line][,line] change ["x]

[line] insert

The append command inserts text after the specified line.

The insert command inserts text before the specified line.

The change command copies the range of lines into a cut buffer, deletes them, and inserts new text where the old text used to be.

For all of these commands, you indicate the end of the text you're inserting by hitting ^D or by entering a line which contains only a period.

3.3 Cut & Paste Commands

[line][,line] delete ["x]
[line][,line] yank ["x]
[line] put ["x]
[line][,line] copy line
[line][,line] to line
[line][,line] move line

The delete command copies the specified range of lines into a cut buffer, and then deletes them.

The yank command copies the specified range of lines into a cut buffer, but does *not* delete them.

The put command inserts text from a cut buffer after the specified line.

The copy and to commands yank the specified range of lines and then immediately paste them after some other line.

The move command deletes the specified range of lines and then immediately pastes them after some other line. If the destination line comes after the deleted text, then it will be adjusted automatically to account for the deleted lines.

3.4 Display Text Commands

[line][,line] print
[line][,line] list
[line][,line] number

The print command displays the specified range of lines.

The number command displays the lines, with line numbers.

The list command also displays them, but it is careful to make control characters visible.

3.5 Global Operations Commands

[line][,line] global /regexp/ command
[line][,line] vglobal /regexp/ command

The global command searches through the lines of the specified range (or through the whole file if no range is specified) for lines that contain a given regular expression. It then moves the cursor to each of these lines and runs some other command on them.

The vglobal command is similar, but it searches for lines that don't contain the regular expression.

3.6 Line Editing Commands

[line][,line] join[!]
[line][,line] ! program
[line][,line] <
[line][,line] >
[line][,line] substitute /regexp/replacement/[p][g][c]
[line][,line] &

The join command catenates all lines in the specified range together to form one big line. If only a single line is specified, then the following line is catenated onto it. The normal ":join" inserts one or two spaces between the lines; the ":join!" variation (with a '!') doesn't insert spaces.

The ! command runs an external filter program, and feeds the specified range of lines to it's stdin. The lines are then replaced by the output of the filter. A typical example would be ":'a,'z!sort" to sort the lines 'a,'z.

The [and > commands shift the specified range of lines left or right, normally by the width of 1 tab character. The "shiftwidth" option determines the shifting amount.

The substitute command finds the regular expression in each line, and replaces it with the replacement text. The "p" option causes the altered lines to be printed. The "g" option permits all instances of the regular expression to be found & replaced. (Without "g", only the first occurrence in each line is replaced.) The "c" option asks for confirmation before each substitution.

The & command repeats the previous substitution command. Actually, "&" is

equivalent to "s//~/" with the same options as last time. It searches for the last regular expression that you specified for any purpose, and replaces it with the the same text that was used in the previous substitution.

3.7 Undo Command

undo

> The undo command restores the file to the state it was in before your most recent command which changed text.

3.8 Configuration & Status Commands

map[!] [key mapped_to]
unmap[!] key
abbr [word expanded_form_of_word]
unabbr word
digraph[!] [XX [Y]]
set [options]
mkexrc
[line] mark "x
visual
version
[line][,line] =
file [file]
source file
@ "x
color [when] [["light"] color] ["on" color]

The map command allows you to configure Elvis to recognize your function keys, and treat them as though they transmitted some other sequence of characters. Normally this mapping is done only when in the visual command mode, but with the [!] present it will map keys under input and replace modes as well. When this command is given with no arguments, it prints a table showing all mappings currently in effect. When called with two arguments, the first is the sequence that your function key really sends, and the second is the sequence that you want Elvis to treat it as having sent. As a special case, if the first argument is a '#' sign followed by a number then Elvis will map the corresponding function key; for example, ":map #7 dd" will cause the *F7* key to delete a line. Also, on some systems, ":map #7s ..." may map *Shift-F7*, ":map #7c ..." may map *Control-F7*, and ":map #7a ..." may map *Alt-F7*.

The unmap command removes key definitions that were made via the map command.

The abbr command is used to define/list a table of abbreviations. The table contains

both the abbreviated form and the fully spelled-out form. When you're in visual input mode, and you type in the abbreviated form, Elvis will replace the abbreviated form with the fully spelled-out form. When this command is called without arguments, it lists the table; with two or more arguments, the first argument is taken as the abbreviated form, and the rest of the command line is the fullyspelled out form.

The unabbr command deletes entries from the abbr table.

The digraph command allows you to display the set of digraphs that Elvis is using, or add/remove a digraph. To list the set of digraphs, use the digraph command with no arguments. To add a digraph, you should give the digraph command two arguments. The first argument is the two ASCII characters that are to be combined; the second is the non-ASCII character that they represent. The non-ASCII character's most significant bit is automatically set by the digraph command, unless to append a ! to the command name. Removal of a digraph is similar to adding a digraph, except that you should leave off the second argument.

The set command allows you examine or set various options. With no arguments, it displays the values of options that have been changed. With the single argument "all" it displays the values of all options, regardless of whether they've been explicitly set or not. Otherwise, the arguments are treated as options to be set.

The mkexrc command saves the current configuration to a file called ".exrc" in the current directory.

The mark command defines a named mark to refer to a specific place in the file. This mark may be used later to specify lines for other commands.

The visual command puts the editor into visual mode. Instead of emulating ex, Elvis will start emulating vi.

The version command tells you that what version of Elvis this is.

The = command tells you what line you specified, or, if you specified a range of lines, it will tell you both endpoints and the number of lines included in the range.

The file command tells you the name of the file, whether it has been modified, the number of lines in the file, and the current line number. You can also use it to change the name of the current file.

The source command reads a sequence of colon mode commands from a file, and interprets them.

The @ command executes the contents of a cut-buffer as EX commands.

The color command only works under MS-DOS, or if you have an ANSI-compatible color terminal.It allows you to set the foreground and background colors for different

types of text: normal, bold, italic, underlined, standout, pop-up menu, and visible selection. By default, it changes the "normal" colors; to change other colors, the first argument to the :color command should be the first letter of the type of text you want.The syntax for the colors themselves is fairly intuitive. For example, ":color light cyan on blue" causes normal text to be displayed in light cyan on a blue background, and ":color b bright white" causes bold text to be displayed in bright white on a blue background. The background color always defaults to the current background color of normal text. Your first :color command must specify both the foreground and background for normal text.

3.9 Conditional Commands

if condition
and condition
or condition
then commands
else commands

These commands allow Elvis to execute a set of commands only if a given condition is valid. The if, and, and or commands set or clear a flag, and the then and else commands test that flag to decide whether to execute their arguments as commands. This can be handy in .exrc files.

A condition can test the following types of values:

filetype - asterisk, followed by filename extension constants - either a number or a quoted string options - the name of a :set option termcap fields - a two letter name, enclosed in colons environment variables - the name, preceded by a dollar sign

The condition can involve either one boolean value, two strings compared for equality ("=" or "==") or inquality ("!="), or two numbers compared with any comparison operator.

The if command sets the conditional flag equal to the results of the condition. The and command performs a logical AND of the conditional flag and the new condition. The or command performs a logical OR of the conditional flag and the new condition.

The then command's arguments are one or more commands. (Commands can be delimited by placing a '|' character between them.)The commands are executed if the conditional flag is true, or skipped if it is false. Similarly, else executes its arguments only if the conditional flag is false.

For example, on my Linux system the console can handle color commands, but xterms can't. To have colors set on the console but not on an xterm, I added the following to my .exrc file...

```
if term="console"
then color yellow on blue | color quit white on blue
```

Note: The .exrc file is executed before elvis loads the first file, so you can't test for a specific filename there, or modify text, or adjust cut buffers. A new initialization file, ".exfilerc", is now supported to fill this need. It resides in your home directory. The .exfilerc file is executed after each file is loaded. A typical .exfilerc file might look like...

```
if *.c
or *.h
and newfile
then 1!mkskel %
```

3.10 Multiple File Commands

```
args [files]
next[!] [files]
Next[!]
previous[!]
rewind[!]
```

When you invoke Elvis from your shell's command line, any filenames that you give to Elvis as arguments are stored in the args list. The args command will display this list, or define a new one.

The next command switches from the current file to the next one in the args list. You may specify a new args list here, too.

The Next and previous commands (they're really aliases for the same command) switch from the current file to the preceding file in the args list.

The rewind command switches from the current file to the first file in the args list.

3.11 Switching Files

```
edit[!] [file]
tag[!] tagname
pop[!]
```

The edit command allows to switch from the current file to some other file. This has nothing to do with the args list, by the way.

The tag command looks up a given tagname in a file called "tags".This tells it which file the tag is in, and how to find it in that file. Elvis then switches to the tag's file and finds the tag.

The pop command reverses a tag command. It switches back to the file and line number from which you invoked the tag command.Up to 15 tag commands can be reversed; the filenames and line numbers are saved on a stack, so you can perform multiple tag commands, and then reverse them with multiple pop commands.

3.12 Working with a Compiler

cc [files]
make [target]
errlist[!] [errlist]

The cc and make commands execute your compiler or "make" utility and redirect any error messages into a file called "errlist". By default, cc is run on the current file.(You should write it before running cc.) The contents of the "errlist" file are then scanned for error messages. If an error message is found, then the cursor is moved to the line where the error was detected, and the description of the error is displayed on the status line.

After you've fixed one error, the errlist command will move the cursor to the next error. In visual command mode, hitting `*' will do this, too.

You can also create an "errlist" file from outside of Elvis, and use "elvis -m" to start Elvis and have the cursor moved to the first error. Note that you don't need to supply a filename with "elvis -m" because the error messages always say which source file an error is in.

Note: When you use errlist repeatedly to fix several errors in a single file, it will attempt to adjust the reported line numbers to allow for lines that you have inserted or deleted.

These adjustments are made with the assumption that you will work though the file from the beginning to the end.

3.13 Exit Commands

quit[!]
wq
xit

The quit command exits from the editor without saving your file.

The wq command writes your file out, then then exits.

The xit command is similar to the wq command, except that xit won't bother to write your file if you haven't modified it.

3.14 File I/O Commands

[line] read file
[line][,line] write[!] [[> >]file]

The read command gets text from another file and inserts it after the specified line. It can also read the output of a program; simply precede the program name by a '!' and use it in place of the file name.

The write command writes the whole file, or just part of it, to some other file. The !, if present, will permit the lines to be written even if you've set the readonly option. If you precede the filename by > > then the lines will be appended to the file.You can send the lines to the standard input of a program by replacing the filename with a '!' followed by the command and its arguments.

Note: Be careful not to confuse ":w!filename" and ":w !command". To write to a program, you must have at least one blank before the '!'.

3.15 Directory Commands

cd [directory]
chdir [directory]
shell

The cd and chdir commands (really two names for one command) switch the current working directory.

The shell command starts an interactive shell.

3.16 Debugging Commands

[line][,line] debug[!]
validate[!]

These commands are only available if you compile Elvis with the -DDEBUG flag.

The debug command lists statistics for the blocks which contain the specified range of lines. If the ! is present, then the contents of those blocks is displayed, too.

The validate command checks certain variables for internal consistency. Normally it doesn't output anything unless it detects a problem. With the !, though, it will always produce *some* output.

4. REGULAR EXPRESSIONS

Elvis uses regular expressions for searching and substitutions. A regular expression is a text string in which some characters have special meanings. This is much more powerful than simple text matching.

Syntax

Elvis' regexp package treats the following one- or two-character strings (called meta-characters) in special ways:

\(subexpression\)
The \(and \) metacharacters are used to delimit subexpressions. When the regular expression matches a particular chunk of text, Elvis will remember which portion of that chunk matched the subexpression.The :s/regexp/newtext/ command makes use of this feature.

^ The ^ metacharacter matches the beginning of a line. If, for example, you wanted to find "foo" at the beginning of a line, you would use a regular expression such as /^foo/. Note that ^ is only a metacharacter if it occurs at the beginning of a regular expression; anyplace else, it is treated as a normal character.

$ The $ metacharacter matches the end of a line. It is only a metacharacter when it occurs at the end of a regular expression; elsewhere, it is treated as a normal character.For example, the regular expression /$$/ will search for a dollar sign at the end of a line.

\. < .The \. <. metacharacter matches a zero-length string at the beginning of a word. A word is considered to be a string of 1 or more letters and digits. A word can begin at the beginning of a line or after 1 or more non-alphanumeric characters.

\. >. The \. >. metacharacter matches a zero-length string at the end of a word. A word can end at the end of the line or before 1 or more non-alphanumeric characters. For example, /\. <.end\. >./ would find any instance of the word "end", but would ignore any instances of e-n-d inside another word such as "calendar".

. The . metacharacter matches any single character.

[character-list]

This matches any single character from the characterlist. Inside the character-list, you can denote a span of characters by writing only the first and last characters, with a hyphen between them. If the character-list is preceded by a ^ character, then the list is inverted -- it will match character that isn't mentioned in the list.For example, /[a-zA-Z]/ matches any letter, and /[^]/ matches anything other than a blank.

\{n\}This is a closure operator, which means that it can only be placed after something that matches a single character. It controls the number of times that the singlecharacter expression should be repeated.

The \{n\} operator, in particular, means that the preceding expression should be repeated exactly n times.For example, /^-\{80\}$/ matches a line of eighty hyphens, and /\ <[a-zA-Z]\{4\}\ >/ matches any four-letter word.

\{n,m\} This is a closure operator which means that the preceding single-character expression should be repeated between n and m times, inclusive. If the m is omitted (but the comma is present) then m is taken to be infinity. For example, /"[^"]\{3,5\}"/ matches any pair of quotes which contains three, four, or five non-quote characters.

* The * metacharacter is a closure operator which means that the preceding single-character expression can be repeated zero or more times. It is equivalent to \{0,\}. For example, /.*/ matches a whole line.

\+ The \+ metacharacter is a closure operator which means that the preceding single-character expression can be repeated one or more times. It is equivalent to \{1,\}. For example, /.\+/ matches a whole line, but only if the line contains at least one character. It doesn't match empty lines.

\? The \? metacharacter is a closure operator which indicates that the preceding single-character expression is optional -- that is, that it can occur 0 or 1 times. It is equivalent to \{0,1\}.For example, /no[-]\?one/ matches "no one", "no-one", or "noone".

Anything else is treated as a normal character which must exactly match a character from the scanned text.The special strings may all be preceded by a backslash to force them to be treated normally.

Substitutions

The :s command has at least two arguments: a regular expression, and a substitution string. The text that matched the regular expression is replaced by text which is derived from the substitution string.

Most characters in the substitution string are copied into the text literally but a few have special meaning:

&Insert a copy of the original text
~Insert a copy of the previous replacement text
\1 Insert a copy of that portion of the original text which matched the first set of \(\) parentheses

\2-\9 Do the same for the second (etc.) pair of \(\)
\U Convert all chars of any later & or \# to uppercase
\L Convert all chars of any later & or \# to lowercase
\E End the effect of \U or \L
\u Convert the first char of the next & or \# to uppercase
\l Convert the first char of the next & or \# to lowercase

These may be preceded by a backslash to force them to be treated normally.If "nomagic" mode is in effect, then & and ~ will be treated normally, and you must write them as \& and \~ for them to have special meaning.

Options

Elvis has two options which affect the way regular expressions are used. These options may be examined or set via the :set command.

The first option is called "[no]magic". This is a boolean option, and it is "magic" (TRUE) by default.While in magic mode, all of the meta-characters behave as described above. In nomagic mode, only ^ and $ retain their special meaning.

The second option is called "[no]ignorecase".This is a boolean option, and it is "noignorecase" (FALSE) by default. While in ignorecase mode, the searching mechanism will not distinguish between an uppercase letter and its lowercase form. In noignorecase mode, uppercase and lowercase are treated as being different.

Also, the "[no]wrapscan" option affects searches.

Examples

This example changes every occurrence of "utilize" to "use":

:%s/utilize/use/g

This example deletes all whitespace that occurs at the end of a line anywhere in the file. (The brackets contain a single space and a single tab.):

:%s/[]\+$//

This example converts the current line to uppercase:

:s/.*/\U&/

This example underlines each letter in the current line, by changing it into an "underscore backspace letter" sequence. (The ^H is entered as "control-V backspace".):

:s/[a-zA-Z]/_^H&/g

This example locates the last colon in a line, and swaps the text before the colon with the text after the colon. The first \(\) pair is used to delimit the stuff before the colon, and the second pair delimit the stuff after. In the substitution text, \1 and \2 are given in reverse order to perform the swap:

:s/\(.*\):\(.*\)/\2:\1/

5. OPTIONS

Options may be set or examined via the colon command "set". The values of options will affect the operation of later commands.

For convenience, options have both a long descriptive name and a short name which is easy to type. You may use either name interchangeably. I like the short names, myself.

There are three types of options: Boolean, string, and numeric. Boolean options are made TRUE by giving the name of the option as an argument to the "set" command; they are made FALSE by prefixing the name with "no". For example, "set autoindent" makes the autoindent option TRUE, and "set noautoindent" makes it FALSE. Elvis also allows boolean options to be toggled by prefixing the name with "neg". So, ":map g :set neglist^M" will cause the <g> key to alternately toggle the "list" option on and off. (The "neg" prefix is an extension; the real vi doesn't support it.)

To change the value of a string or numeric option, pass the "set" command the name of the option, followed by an "=" sign and the option's new value. For example, "set tabstop=8" will give the tabstop option a value of 8. For string options, you may enclose the new value in quotes.

NAMES TYPE DEFAULT MEANING

autoindent, aiBool noaiauto-indent during input autoprint, ap Bool ap in EX, print the current line autotab, atBool at auto-indent allowed to use tabs? autowrite, aw Bool noawauto-write when switching files beautify, bf Bool nobfstrip control chars from file? charattr, ca Bool nocainterpret \fX sequences? cc, ccStr cc="cc -c" name of the C compiler columns, coNum co=80 width of the screen digraph, dig Bool nodig recognize

digraphs? directory, dirStr dir="/usr/tmp"where tmp files are kept edcompatible, edBool noedremember ":s//" options equalprg, ep Bool ep="fmt" program to run for = operator errorbells, ebBool eb ring bell on error exrc, exrc Bool noexrcread "./.exrc" file? exrefresh, er Bool er write lines individually in EX flash, vbell Bool flash use visible alternative to bell flipcase, fc Str fc="" non-ASCII chars flipped by ~ hideformat, hfBool hf hide text formatter commands ignorecase, icBool noicupper/lowercase match in search inputmode, im Bool noimstart vi in insert mode? keytime, ktNum kt=2timeout for mapped key entry keywordprg, kpStr kp="ref" full pathname of shift-K prog lines, lnNum ln=25 number of lines on the screen list, li Bool nolidisplay lines in "list" mode magic, maBool ma use regular expression in search make, mk Str mk="make"name of the "make" program mesg, ms Bool ms allow messages from other users? modelines, ml Bool nomlare modelines processed? more, more Bool morepause between messages? nearscroll, nsNum ns=15 when to scroll vs. redraw newfile, new BOOL nonew is current file new? novice, novBool nonovice set options for ease of use number, nu Bool nonumber show line numbers paragraphs, paraStr para="PPppIPLPQP"names of "paragraph" nroff cmd prompt, pr Bool pr show ':' prompt in ex mode readonly, ro Bool noroprevent overwriting of orig file remap, rem Bool remap allow key maps to call key maps report, re Num re=5report when 5 or more changes ruler, ruBool norudisplay line/column numbers scroll, sc Num sc=12 scroll amount for ^U and ^D sections, sectStr sect="NHSHSSSEse"names of "section" nroff cmd shell, shStr sh="/bin/sh" full pathname of the shell showmatch, sm Bool nosmshow matching ()[]{} showmode, smd Bool nosmd say when we're in input mode shiftwidth, swNum sw=8shift amount for < and > sidescroll, ssNum ss=8amount of sideways scrolling sync, sy Bool nosycall sync() often tabstop, tsNum ts=8width of tab characters taglength, tl Num tl=0significant chars in tag name tags, tagStr tags="tags" list of tags files tagstack, tgs Bool tgs enable tagstack? term, te Str te="$TERM" name of the termcap entry terse, trBool notrgive shorter error messages timeout, toBool to distinguish <esc > from <arrow >? warn, wa Bool wa warn for ! if file modified window, wi Num wi=24 lines to redraw after long move wrapmargin, wmNum wm=0wrap long lines in input mode wrapscan, ws Bool ws at EOF, searches wrap to line 1 writeany, wr Bool nowrallow :w to clobber files.

autoindent, ai

> During input mode, the autoindent option will cause each added line to begin with the same amount of leading whitespace as the line above it. Without autoindent, added lines are initially empty.

autoprint, ap

> This option only affects EX mode. If the autoprint option on, and either the cursor has moved to a different line or the previous command modified the file, then Elvis will print the current line.

autotab, at

> This option affects the behavior of the autoindent mode. If autoindent is turned

off, then autotab has no effect.

When autotab is turned on, elvis will use a mixture of spaces and tabs to create the proper amount of indentation. This is the default. When autotab is turned off, elvis will only use spaces for auto-indent. Elvis will still insert a real tab character when you hit the *Tab* key, though; the autotab option only affects automatic indentation.

autowrite, aw

When you're editing one file and decide to switch to another-via the :tag command, or :next command, perhaps - if your current file has been modified, then Elvis will normally print an error message and refuse to switch. However, if the autowrite option is on, then Elvis will write the modified version of the current file and successfully switch to the new file.

beautify, bf

This option causes all control characters to be deleted from the text file, at the time when you start editing it.If you're already editing a file when you turn on the beautify option, then that file won't be affected. ccThe :cc command runs the C compiler. This option should be set to the name of your compiler.

charattr, ca

Many text formatting programs allow you to designate portions of your text to be underlined, italicized, or boldface by embedding the special strings \fU, \fI, and \fB in your text. The special string \fP marks the end of underlined or boldface text. Elvis normally treats those special strings just like any other text. However, if the charattr option is on, then Elvis will interpret those special strings correctly, to display underlined or boldface text on the screen. (This only works, of course, if your terminal can display underlined and boldface, and if the TERMCAP entry says how to do it.)

columns, co

This option shows how wide your screen is.

digraph, dig

This option is used to enable/disable recognition of digraphs.The default value is nodigraph, which means that digraphs will not be recognized.

directory, dir

Elvis stores text in temporary files.This option allows you to control which

directory those temporary files will appear in. The default is /usr/tmp. This option can only be set in a .exrc file; after that, Elvis will have already started making temporary files in some other directory, so it would be too late.

edcompatible, ed

This option affects the behavior of the ":s/regexp/text/options" command.It is normally off (:se noed) which causes all of the substitution options to be off unless explicitly given. However, with edcompatible on (:se ed), the substitution command remembers which options you used last time.Those same options will continue to be used until you change them. In edcompatible mode, when you explicitly give the name of a substitution option, you will toggle the state of that option. This all seems very strange to me, but its implementation was almost free when I added the ":&" command to repeat the previous substitution, so there it is.

equalprg, ep

This holds the name & arguments of the external filter program used the the visual = operator. The default value is "fmt", so the = operator will adjust line breaks in text.

errorbells, eb

Elvis normally rings a bell when you do something wrong. This option lets you disable the bell.

exrc

This option specifies whether a .exrc file in the current directory should be executed. By default, this option is off (":set noexrc") which prevents elvis from executing .exrc in the current directory. If the .exrc file in your home directory turns this option on (":set exrc") then the Elvis will attempt to execute the .exrc file in the current directory. This option exist mainly for security reasons. A meanspirited person could do something like echo >/tmp/.exrc '!rm -rf $HOME' and then anybody who attempted to edit or view a file in the /tmp directory would lose most of their files. With the exrc option turned off, this couldn't happen to you.

exrefresh, er

The EX mode of Elvis writes many lines to the screen.You can make Elvis either write each line to the screen separately, or save up many lines and write them all at once. The exrefresh option is normally on, so each line is written to the screen separately. You may wish to turn the exrefresh option off (:se noer) if the "write" system call is costly on your machine, or if you're using a windowing environment. (Windowing environments scroll text a lot faster when

you write many lines at once.) This option has no effect in visual command mode or input mode.

flash, vbell

If your termcap entry describes a visible alternative to ringing your terminal's bell, then this option will say whether the visible version gets used or not.Normally it will be. If your termcap does NOT include a visible bell capability, then the flash option will be off, and you can't turn it on.

flipcase, fc

The flipcase option allows you to control how the non-ASCII characters are altered by the "~" command. The string is divided into pairs of characters. When "~" is applied to a non-ASCII character, Elvis looks up the character in the flipcase string to see which pair it's in, and replaces it by the other character of the pair. hideformat, hf Many text formatters require you to embed format commands in your text, on lines that start with a "." character.Elvis normally displays these lines like any other text, but if the hideformat option is on, then format lines are displayed as blank lines.

ignorecase, ic

Normally, when Elvis searches for text, it treats uppercase letters as being different for lowercase letters. When the ignorecase option is on, uppercase and lowercase are treated as equal.

inputmode, im

This option allows you to have Elvis start up in insert mode. You can still exit insert mode at any time by hitting the ESC key, as usual. Usually, this option would be set in your ".exrc" file.

keytime, kt

The arrow keys of most terminals send a multi-character sequence.It takes a measurable amount of time for these sequences to be transmitted. The keytime option allows you to control the maximum amount of time to allow for an arrow key (or other mapped key) to be received in full. On most systems, the setting is the number of tenths of a second to allow between characters. On some other systems, the setting is in whole seconds. Try to avoid setting keytime=1.Most systems just count clock beats, so if you tried to read a character shortly before a clock beat, you could allow almost no time at all for reading the characters.For higher keytime settings, the difference is less critical. If your system's response time is poor, you might want to increase the keytime. In particular, I've found that when keystrokes must be sent through a network (via X windows, rlogin, or telnet, for example) the keytime should be set to at least 1 second. As a special

case, you can set keytime to 0 to disable this time limit stuff altogether. The big problem here is: If your arrow keys' sequences start with an ESC, then every time you hit your ESC key Elvis will wait... and wait... to see if maybe that ESC was part of an arrow key's sequence.

NOTE: this option is a generalization of the timeout option of the real vi. keywordprg, kp Elvis has a special keyword lookup feature. You move the cursor onto a word, and hit shift-K, and Elvis uses another program to look up the word and display information about it.

This option says which program gets run. The default value of this option is "ref", which is a program that looks up the definition of a function in C. It looks up the function name in a file called "refs" which is created by ctags. You can substitute other programs, such as an English dictionary program or the online manual. Elvis runs the program, using the keyword as its only argument. The program should write information to stdout. The program's exit status should be 0, unless you want Elvis to print " < < < failed > > >".

lines, ln

This option says how many lines you screen has.

list, li

In nolist mode (the default), Elvis displays text in a "normal" manner -- with tabs expanded to an appropriate number of spaces, etc. However, sometimes it is useful to have tab characters displayed differently.In list mode, tabs are displayed as "^I", and a "$" is displayed at the end of each line.

magic, ma

The search mechanism in Elvis can accept "regular expressions" -- strings in which certain characters have special meaning. The magic option is normally on, which causes these characters to be treated specially. If you turn the magic option off (:se noma), then all characters except ^ and $ are treated literally. ^ and $ retain their special meanings regardless of the setting of magic.

make, mk

The :make command runs your "make" program. This option defines the name of your "make" program.

mesg

With the real vi, running under real UNIX, ":set nomesg" would prevent other users from sending you messages. Elvis ignores it, though.

modelines, ml

Elvis supports modelines. Modelines are lines near the beginning or end of your text file which contain "ex:yowza:", where "yowza" is any EX command. A typical "yowza" would be something like "set ts=5 ca kp=spell wm=15". Other text may also appear on a modeline, so you can place the "ex:yowza:" in a comment: /* ex:set sw=4 ai: */ Normally these lines are ignored, for security reasons, but if you have "set modelines" in your .exrc file then "yowza" is executed.

nearscroll, ns

The line that contains the cursor will always be on the screen. If you move the cursor to a line that isn't on the screen, then elvis will either scroll (if the cursor's line is nearly on the screen already) or redraw the screen completely with the cursor's line centered (if the cursor line is not near the screen already) This option allows you to control elvis' idea of "near".A value of 15 is typical. A value of 1 would cause elvis to scroll no more that one line.A value of 0 disables scrolling.

newfile, new

The "newfile" option is an unsettable boolean option. Its value is automatically set to FALSE when you start editing a file that already exists, or TRUE if the file doesn't exist yet. This can be handy in ".exfilerc" initialization files.

novice, nov

The command ":set novice" is equivalent to ":set nomagic report=1 showmode".

number, nu

The "number" option causes Elvis to display line numbers at the start of each line. The numbers are not actually part of the text; when the file is written out, it will be written without line numbers.

paragraphs, pa

The { and } commands move the cursor forward or backward in increments of one paragraph. Paragraphs may be separated by blank lines, or by a "dot" command of a text formatter. Different text formatters use different "dot" commands. This option allows you to configure Elvis to work with your text formatter. It is assumed that your formatter uses commands that start with a "." character at the front of a line, and then have a one- or two-character command name. The value of the paragraphs option is a string in which each pair of characters is one possible form of your text formatter's paragraph command. more When Elvis must display a sequence of messages at the bottom line of the

screen in visual mode, it normally pauses after all but the last one, so you have time to read them all. If you turn off the "more" option, then Elvis will not pause.This means you can only read the last message, but it is usually the most important one anyway.

prompt, pr

If you ":set noprompt", then Elvis will no longer emit a ':' when it expects you to type in an ex command. This is slightly useful if you're using an astonishingly slow UNIX machine, but the rest of us can just ignore this one.

readonly, ro

Normally, Elvis will let you write back any file to which you have write permission. If you don't have write permission, then you can only write the changed version of the file to a different file. If you set the readonly option, then Elvis will pretend you don't have write permission to any file you edit. It is useful when you really only mean to use Elvis to look at a file, not to change it. This way you can't change it accidentally. This option is normally off, unless you use the "view" alias of Elvis. "View" is like "vi" except that the readonly option is on.

remap

The ":map" command allows you to convert one key sequence into another.The remap option allows you to specify what should happen if portions of that other sequence are also in the map table. If remap is on, then those portions will also be mapped, just as if they had been typed on the keyboard. If remap is off, then the matching portions will not be mapped. For example, if you enter the commands ":map A B" and ":map B C", then when remap is on, A will be converted to C. But when remap is off, A will be converted only to B.

report, re

Commands in Elvis may affect many lines. For commands that affect a lot of lines, Elvis will output a message saying what was done and how many lines were affected. This option allows you to define what "a lot of lines" means. The default is 5, so any command which affects 5 or more lines will cause a message to be shown.

ruler, ru

This option is normally off. If you turn it on, then Elvis will constantly display the line/column numbers of the cursor, at the bottom of the screen.

scroll, sc

The ^U and ^D keys normally scroll backward or forward by half a screenful, but this is adjustable. The value of this option says how many lines those keys should scroll by. If you invoke ^U or ^D with a count argument (for example, "33^D") then this option's value is set to the count.

sections, se

The [[and]] commands move the cursor backward or forward in increments of 1 section. Sections may be delimited by a { character in column 1 (which is useful for C source code) or by means of a text formatter's "dot" commands. This option allows you to configure Elvis to work with your text formatter's "section" command, in exactly the same way that the paragraphs option makes it work with the formatter's "paragraphs" command.

shell, sh

When Elvis forks a shell (perhaps for the :! or :shell commands) this is the program that is uses as a shell. This is "/bin/sh" by default, unless you have set the SHELL (or COMSPEC, for MS-DOS) environment variable, it which case the default value is copied from the environment.

shiftwidth, sw

The < and > commands shift text left or right by some uniform number of columns. The shiftwidth option defines that "uniform number". The default is 8.

showmatch, sm

With showmatch set, in input mode every time you hit one of)}], Elvis will momentarily move the cursor to the matching ({[.

showmode, smd

In visual mode, it is easy to forget whether you're in the visual command mode or input/replace mode.Normally, the showmode option is off, and you haven't a clue as to which mode you're in. If you turn the showmode option on, though, a little message will appear in the lower right-hand corner of your screen, telling you which mode you're in.

sidescroll, ss

For long lines, Elvis scrolls sideways. (This is different from the real vi, which wraps a single long line onto several rows of the screen.) To minimize the number of scrolls needed, Elvis moves the screen sideways by several characters at a time. The value of this option says how many characters' widths to scroll at a time. Generally, the faster your screen can be redrawn, the lower the value you will want in this option.

sync, sy

If the system crashes during an edit session, then most of your work can be recovered from the temporary file that Elvis uses to store changes. However, sometimes the OS will not copy changes to the hard disk immediately, so recovery might not be possible. The [no]sync option lets you control this. In nosync mode (which is the default, for UNIX), Elvis lets the operating system control when data is written to the disk. This is generally faster. In sync mode (which is the default for MS-DOS, AmigaDos, and Atari TOS), Elvis forces all changes out to disk every time you make a change. This is generally safer, but slower. It can also be a rather rude thing to do on a multi-user system.

tabstop, ts

Tab characters are normally 8 characters wide, but you can change their widths by means of this option.

taglength, tl

This option allows you to specify how many characters of a tag's name must match when performing tag lookup. As a special case, ":set taglength=0" means that all characters of a tag's name must match. Note: some configurations of Elvis don't support this option.

tags, tag

If your version of elvis is compiled with -DINTERNAL_TAGS, then this is a space-delimited list of tags files. When you tell elvis to look up a tag, it searches though each file in turn until it finds the tag. If your version of elvis is compiled without -DINTERNAL_TAGS, then you can achieve the same effect via an environment variable called TAGPATH.TAGPATH's value is a colon-delimited list of file or directory names. (For some operating systems, including MS-DOS, the list is delimited by semicolons instead of colons.)

tagstack

This option allows you to disable the tagstack.I can't think of any reason why you would want to do that.

term, te

This read-only option shows the name of the termcap entry that Elvis is using for your terminal.

terse, tr

The real vi uses this option to select longer vs. shorter error messages.Elvis has only one set of error messages, though, so this option has no effect.

timeout, to

The command ":set notimeout" is equivalent to ":set keytime=0", and ":set timeout" is equivalent to ":set keytime=1". This affects the behavior of the *Esc* key. See the discussion of the "keytime" option for more information.

warn, wa

If you have modified a file but not yet written it back to disk, then Elvis will normally print a warning before executing a ":!cmd" command. However, in nowarn mode, this warning is not given. Elvis also normally prints a message after a successful search that wrapped at EOF. The [no]warn option can also disable this warning.

window, wi

This option controls how many lines are redrawn after a long move. On fast terminals, this is usually set to the number of rows that the terminal can display, minus one. This causes the entire screen to be filled with text around the cursor. On slow terminals, you may wish to reduce this value to about 7 or so. That way, if you're doing something like repeatedly hitting 'n' to search for each occurrence of some string and trying to find a particular occurrence, then you don't need to wait as long for Elvis to redraw the screen after each search.

wrapmargin, wm

Normally (with wrapmargin=0) Elvis will let you type in extremely long lines, if you wish. However, with warpmargin set to something other that 0 (wrapmargin=10 is nice), Elvis will automatically cause long lines to be "wrapped" on a word break for lines come too close to the right-hand margin. For example: On an 80-column screen, ":set wm=10" will cause lines to wrap when their length exceeds 70 columns.

wrapscan, ws

Normally, when you search for something, Elvis will find it no matter where it is in the file. Elvis starts at the cursor position, and searches forward. If Elvis hits EOF without finding what you're looking for, then it wraps around to continue searching from line 1. If you turn off the wrapscan option (:se nows), then when Elvis hits EOF during a search, it will stop and say so.

writeany, wr

With "writeany" turned off, elvis will prevent you from accidentally overwriting a file. For example, if "foo" exists then ":w foo" will fail. If you turn on the "writeany" option, then ":w foo" will work. Regardless of the setting of "writeany", though, ":w! foo" will work.The '!' forces the ":w" command to write the file unless the operating system won't allow it.

6. INITIALIZATION

Many features of Elvis are configurable at runtime. There are commands for assigning actions to keys (:map), defining abbreviations (:abbr), non-ASCII keying sequences (:digraph), setting screen colors (:color), and miscellaneous other options (:set).

All of these commands can be issued interactively. Experienced vi users generally prefer to have some options set every time they run they execute Elvis, and Elvis has ways to support this.

Start-up Initialization

When Elvis starts, it executes the following algorithm in an attempt to locate initialization commands: If this version of Elvis supports a system-wide initialization file and that file exists, Interpret that file's contents as a series of "ex" commands If the EXINIT environment variable is set Interpret the value of EXINIT as an "ex" command line. Else if the home directory contains a file named ".exrc" Interpret that file's contents as a series of "ex" commands If the "exrc" option is set, and the current directory contains a file named ".exrc". Interpret that file's contents as a series of "ex" commands If a tag was specified via a "-t" command line argument, Execute a tag look-up, and load file if successful If no tag was specified, or the specified tag wasn't found, Load the first file named on the command line, or start empty buffer. If a command was specified via "+command" or "-c command"

Execute the given command

Note that most of this initialization occurs before the first file is loaded. Consequently, commands which examine or change the edit buffer can't be used there. Only "+command" or "-c command" is executed after the text file has been loaded. On non-UNIX systems, ".exrc" is usually an invalid filename so the file is called "ELVIS.RC" instead. Also, the home directory is the directory named by the HOME environment variable; on DOS and a few other systems, if HOME is unset then Elvis will use the directory which contains the executable file (ELVIS.EXE) as your home directory.

File Initialization

Loading a file, too, can cause commands to be executed. Each time any file is loaded into the edit buffer, the following algorithm is used to locate file-specific initialization commands. Fill the edit buffer with the file's contents, and set various options and

variables accordingly. If the home directory contains a file involves named ".exfilerc" Interpret the contents of that file as a series of "ex" commands. If the "modelines" option is set, Search the first 5 & last 5 lines of the text for lines which contain "ex: <command >:" or "vi: <command >:", and interpret any <command > as an "ex" command line. On non-UNIX systems, ".exfilerc" is usually an invalid filename, so the file is called "EXFILE.RC" instead.

The :mkexrc Command

Elvis has a special command, ":mkexrc [filename]", to help you create ".exrc" files. It creates a file which sets all nonstandard options, maps, and so on. By default, the created file's name will be ".exrc" in the current directory. You can either add ":set exrc" to the .exrc file in your home directory to force Elvis to read this new .exrc in your current directory, or you can move this new .exrc file into your home directory. Alternatively, you can supply an explicit filename as an argument to :mkexrc. Afterward, you may wish to edit the created file. For example, some options may be conditional; the :mkexrc file doesn't distinguish between options which were set unconditionally from those that were set in a text file's modelines or other conditional context.

Warning: the :mkexrc command will happily overwrite any file that you tell it to, if your operating system permits.

Other Techniques

Elvis has commands for conditional execution, but the standard vi doesn't. If you often use the real vi, you may want to avoid Elvis's extensions. To have a terminal-dependent initialization file, you can add ":so $HOME/.exrc.$TERM" to the end of your.exrcfile, and then create files with names like ".exrc.vt100" and ".exrc.ansi" in your home directory which contain the terminal-dependent commands.

Another good technique is to write a shell-script "wrapper" around Elvis/vi. Here's one of my favorites. It uses "grep" to locate files containing a given regular expression, and then starts vi on those files with the cursor positioned on the first occurrence in the first file. I call this script "vg".

```
#!/bin/sh
case "$#" in
 0) echo "usage: vg regexp [files]..."  >&2; exit;;
 1) set -- "$1" *.[ch];;
esac
regexp="$1"
shift
vi +/"$regexp" `grep -l "$regexp" "$@"`
```

7. CUT BUFFERS

When Elvis deletes text, it stores that text in a cut buffer. This happens in both visual mode and EX mode. There is no practical limit to how much text a cut buffer can hold. There are 36 cut buffers: 26 named buffers ("a through "z), 9 anonymous buffers ("1 through "9), and 1 extra cut buffer (".). In EX mode, the :move and :copy commands use a cut buffer to temporarily hold the text to be moved/copied.

7.1 Putting text into a Cut Buffer

In visual mode, text is copied into a cut buffer when you use the d, y, c, C, s, or x commands. There are also a few others. By default, the text goes into the "1 buffer. The text that used to be in "1 gets shifted into "2, "2 gets shifted into "3, and so on. The text that used to be in "9 is lost.This way, the last 9 things you deleted are still accessible. You can also put the text into a named buffer -- "a through "z.To do this, you should type the buffer's name (two keystrokes: a double-quote and a lowercase letter) before the command that will cut the text. When you do this, "1 through "9 are not affected by the cut.

You can append text to one of the named buffers. To do this, type the buffer's name in uppercase (a double-quote and an uppercase letter) before the d/y/c/C/s/x command. The ". buffer is special.It isn't affected by the d/y/c/C/s/x command. Instead, it stores the text that you typed in the last time you were in input mode. It is used to implement the . visual command, and ^A in input mode.

In EX mode (also known as colon mode), the :delete, :change, and :yank commands all copy text into a cut buffer. Like the visual commands, these EX commands normally use the "1 buffer, but you can use one of the named buffers by giving its name after the command.

For example, :20,30y a will copy lines 20 through 30 into cut buffer "a.

You can't directly put text into the ". buffer, or the "2 through "9 buffers.

7.2 Pasting from a Cut Buffer

There are two styles of pasting: line-mode and charactermode. If a cut buffer contains whole lines (from a command like "dd") then line-mode pasting is used; if it contains partial lines (from a command like "dw") then character-mode pasting is used. The EX commands always cut whole lines. Character-mode pasting causes the text to be inserted into the line that the cursor is on. Line-mode pasting inserts the text on a new line above or below the line that the cursor is on. It doesn't affect the cursor's line at all. In visual mode, the p and P commands insert text from a cut buffer. Uppercase P will insert it before the cursor, and lowercase p will insert it after the cursor. Normally,

these commands will paste from the "1 buffer, but you can specify any other buffer to paste from. Just type its name (a double-quote and another character) before you type the P or p. In EX mode, the (pu)t command pastes text after a given line. To paste from a buffer other that "1, enter its name after the command.

7.3 Macros

The contents of a named cut buffer can be executed as a series of ex/vi commands. To put the instructions into the cut buffer, you must first insert them into the file, and then delete them into a named cut buffer. To execute a cut buffer's contents as EX commands, you should give the EX command "@" and the name of the buffer. For example, :@z will execute "z as a series of EX commands. To execute a cut buffer's contents as visual commands, you should give the visual command "@" and the letter of the buffer's name. The visual "@" command is different from the EX "@" command. They interpret the cut buffer's contents differently. The visual @ command can be rather finicky. Each character in the buffer is interpreted as a keystroke. If you load the instructions into the cut buffer via a "zdd command, then the newline character at the end of the line will be executed just like any other character, so the cursor would be moved down 1 line. If you don't want the cursor to move down 1 line at the end of each @z command, then you should load the cut buffer by saying 0"zD instead.

Although cut buffers can hold any amount of text, Elvis can only execute small buffers. The size limit is roughly 1000 characters, for either EX macros or VI macros. If a buffer is too large to execute, an error message is displayed.

You can't nest :@ commands. You can't run :@ commands from your .exrc file, or any other :source file either.Similarly, you can't run a :source command from within an @ command. Hopefully, these restrictions will be lifted in a later version.

7.4 The Effect of Switching Files

When Elvis first starts up, all cut buffers are empty. When you switch to a different file (via the :n or :e commands perhaps) the 9 anonymous cut buffers are emptied again, but the other 27 buffers ("a through "z, and ".) retain their text.

8. DIFFERENCES BETWEEN Elvis & BSD VI/EX

Elvis is not 100% compatible with the real vi/ex. Elvis has many small extensions, some omissions, and a few features which are implemented in a slightly different manner.

8.1 Extensions

Save Configuration

The :mkexrc command saves the current :set, :map, :ab, :color, and :digraph configurations in the ".exrc" file in your current directory.

Previous File

The :N or :prev command moves backwards through the args list.

Center Current Row

In visual command mode, the (lowercase) "zz" command will center the current line on the screen, like "z=".

Changing Repeat Count

The default count value for . is the same as the previous command which . is meant to repeat. However, you can supply a new count if you wish. For example, after "3dw", "." will delete 3 words, but "5." will delete 5 words.

Previous Text

The text which was most recently input (via a "cw" command, or something similar) is saved in a cut buffer called ". (which is a pretty hard name to write in an English sentence).

Keyword Lookup

In visual command mode, you can move the cursor onto a word and press shift-K to have Elvis run a reference program to look that word up. This command alone is worth the price of admission! See the ctags and ref programs.

Increment/Decrement

In visual command mode, you can move the cursor onto a number and then hit ## or #+ to increment that number by 1. To increment it by a larger amount, type in the increment value before hitting the initial #. The number can also be decremented or set by hitting #- or #=, respectively. Input ModeYou can backspace past the beginning of the line.

The arrow keys work in input mode.

If you type control-A, then the text that you input last time is inserted. You will remain in input mode, so you can backspace over part of it, or add more to it. (This is sort of like control-@ on the real vi, except that control-A really works.) Control-P will insert the contents of the cut buffer. Real vi can only remember up to 128 characters of input, but Elvis can remember any amount. The ^T and ^D keys can adjust the indent of a line no matter where the cursor happens to be

in that line. You can save your file and exit Elvis directly from input mode by hitting control-Z twice. Elvis supports digraphs as a way to enter non-ASCII characters.

Start in Input Mode

If you ":set inputmode" in your .exrc file, then Elvis will start up in input mode instead of visual command mode.

Visible Fonts

With ":set charattr", Elvis can display "backslash-f" style character attributes on the screen as you edit. The following example shows the recognized attributes:

normal \fBboldface\fR\fIitalics\fR\fUunderlined\fR normal

NOTE: you must compile Elvis without the -DNO_CHARATTR flag for this to work.

File Syncing

After a crash, you can usually recover the altered form of the file from the temporary file that Elvis uses -- unless the temporary file was corrupted. UNIX systems use a delayed-write cache, which means that when Elvis tries to write to the temporary file, the information might still be in RAM instead of on the disk. A power failure at that time would cause the in-RAM information to be lost. UNIX's sync() call will force all such information to disk. MS-DOS and Atari TOS don't write a file's length to disk until that file is closed. Consequently, the temporary file would appear to be 0 bytes long if power failed when we were editing. To avoid this problem, a sync() function has been written which will close the temporary file and then immediately reopen it.

Cursor Shape

Elvis changes the shape of the cursor to indicate which mode you're in, if your terminal's termcap entry includes the necessary capabilities.

Hide nroff Lines

The ":set hideformat" option hides nroff format control lines.(They are displayed on the screen as blank lines.)

Compiler Interface

Elvis is clever enough to parse the error messages emitted by many compilers. To use this feature, you should collect your compiler's error messages into a file called "errlist"; Elvis will read this file, determine which source file caused the error messages, start editing that file, move the cursor to the line where the error

was detected, and display the error message on the status line. Nifty!

Visible Text Selection

In visual command mode, 'v' starts visibly selecting characters and 'V' starts visibly selecting whole lines.The character or line where the cursor is located becomes one endpoint of the selection.You can then use the standard cursor movement commands to move the cursor to the other endpoint, and then press one of the operator commands (c/d/y/ </ >/!/=/\). The operator will then immediately be applied to the selected text. Pop-up Menu Operator The '\' key is a new operator, similar in operation to the c/d/y/ </ >/! operators. It conjures up a menu, from which you can select any of the other operators plus a few other common commands.

Preset Filter Operator

The '=' key is another new operator. It is similar to the '!' operator, except that while '!' asks you to type in a filter command each time, '=' assumes it should always run the command stored in the equalprg option.

Move to a Given Percentage

The '%' movement key can now accept an optional count. Without a count, the '%' key still moves to a matching parenthesis like it always did. With a count somewhere between 1 and 100, though, it moves the cursor to approximately a given percentage of the way through the file.For example, typing "50%" will move the cursor to the middle of the file.

Regular Expressions

In regular expressions, several new forms of closure operators are supported: \{n}, \{n,m}, \+, and \?.

8.2 Omissions

The replace mode is a hack. It doesn't save the text that it overwrites.

Long lines are displayed differently -- where the real vi would wrap a long line onto several rows of the screen, Elvis simply displays part of the line, and allows you to scroll the screen sideways to see the rest of it. The ":preserve" and ":recover" commands are missing. So is the -r flag. I've never had a good reason to use ":preserve", and since ":recover" is used so rarely I decided to implement it as a separate program. There's no need to load the recovery code into memory every time you edit a file, I figured. LISP support is missing. However, the = key is still an operator that reformats lines of text. By default, it reformats lines by sending them through the fmt filter, but you could

write your own LISP beautifier and configure elvis to use it. Key mappings could take care of most other differences. Auto-indent is the only thing that is irrecoverably lost. Autoindent mode acts a little different from the real vi, anyway.It doesn't handle ^^D or 0^D correctly. On the other hand, it does allow ^D and ^T to be used anywhere in the line, to adjust the indentation for the whole line.

9. INTERNAL

You don't need to know the material in this section to use Elvis. You only need it if you intend to modify Elvis. You should also check out the CFLAGS, TERMCAP, ENVIRONMENT VARIABLES, VERSIONS, and QUIESTIONS & ANSWERS sections of this manual.

9.1 The temporary file

The temporary file is divided into blocks of 1024 bytes each. The functions in "blk.c" maintain a cache of the five most recently used blocks, to minimize file I/O. When Elvis starts up, the file is copied into the temporary file by the function tmpstart() in "tmp.c". Small amounts of extra space are inserted into the temporary file to insure that no text lines cross block boundaries. This speeds up processing and simplifies storage management. The extra space is filled with NUL characters.the input file must not contain any NULs, to avoid confusion. This also limits lines to a length of 1023 characters or less.

The data blocks aren't necessarily stored in sequence. For example, it is entirely possible that the data block containing the first lines of text will be stored after the block containing the last lines of text. In RAM, Elvis maintains two lists: one that describes the "proper" order of the disk blocks, and another that records the line number of the last line in each block. When Elvis needs to fetch a given line of text, it uses these tables to locate the data block which contains that line. Before each change is made to the file, these lists are copied.The copies can be used to "undo" the change. Also, the first list -- the one that lists the data blocks in their proper order -- is written to the first data block of the temp file. This list can be used during file recovery. When blocks are altered, they are rewritten to a different block in the file, and the order list is updated accordingly. The original block is left intact, so that "undo" can be performed easily. Elvis will eventually reclaim the original block, when it is no longer needed.

9.2 Implementation of Editing

There are three basic operations which affect text:

o delete text - delete(from, to)
o add text- add(at, text)
o yank text - cut(from, to)

To yank text, all text between two text positions is copied into a cut buffer. The original text is not changed. To copy the text into a cut buffer, you need only remember which physical blocks that contain the cut text, the offset into the first block of the start of the cut, the offset into the last block of the end of the cut, and what kind of cut it was. (Cuts may be either character cuts or line cuts; the kind of a cut affects the way it is later "put".) Yanking is implemented in the function cut(), and pasting is implemented in the function paste(). These functions are defined in "cut.c". To delete text, you must modify the first and last blocks, and remove any reference to the intervening blocks in the header's list.The text to be deleted is specified by two marks.

This is implemented in the function delete().

> To add text, you must specify the text to insert (as a NUL-terminated string) and the place to insert it (as a mark). The block into which the text is to be inserted may need to be split into as many as four blocks, with new intervening blocks needed as well... or it could be as simple as modifying a single block.

This is implemented in the function add().

> There is also a change() function, which generally just calls delete() and add(). For the special case where a single character is being replaced by another single character, though, change() will optimize things somewhat. The add(), delete(), and change() functions are all defined in "modify.c". The input() function reads text from a user and inserts it into the file. It makes heavy use of the add(), delete(), and change() functions. It inserts characters one at a time, as they are typed.

When text is modified, an internal file-revision counter, called changes, is incremented. This counter is used to detect when certain caches are out of date. (The "changes" counter is also incremented when we switch to a different file, and also in one or two similar situations -- all related to invalidating caches.)

9.3 Marks and the Cursor

Marks are places within the text. They are represented internally as 32-bit values which are split into two bitfields: a line number and a character index. Line numbers start with 1, and character indexes start with 0. Lines can be up to 1023 characters long, so the character index is 10 bits wide and the line number fills the remaining 22 bits in the long int.

Since line numbers start with 1, it is impossible for a valid mark to have a value of 0L. 0L is therefore used to represent unset marks.

When you do the "delete text" change, any marks that were part of the deleted text are

unset, and any marks that were set to points after it are adjusted. Marks are adjusted similarly after new text is inserted. The cursor is represented as a mark.

9.4 Colon Command Interpretation

Colon commands are parsed, and the command name is looked up in an array of structures which also contain a pointer to the function that implements the command, and a description of the arguments that the command can take. If the command is recognized and its arguments are legal, then the function is called. Each function performs its task; this may cause the cursor to be moved to a different line, or whatever.

9.5 Screen Control

In input mode or visual command mode, the screen is redrawn by a function called redraw(). This function is called in the getkey() function before each keystroke is read in, if necessary. Redraw() writes to the screen via a package which looks like the "curses" library, but isn't. It is actually much simpler. Most curses operations are implemented as macros which copy characters into a large I/O buffer, which is then written with a single large write() call as part of the refresh() operation.

(Note: Under MS-DOS, the pseudo-curses macros check to see whether you're using the pcbios interface. If you are, then the macros call functions in "pc.c" to implement screen updates.) The low-level functions which modify text (namely add(), delete(), and change()) supply redraw() with clues to help redraw() decide which parts of the screen must be redrawn. The clues are given via a function called redrawrange(). Most EX commands use the pseudo-curses package to perform their output, like redraw(). There is also a function called msg() which uses the same syntax as printf(). In EX mode, msg() writes message to the screen and automatically adds a newline.In VI mode, msg() writes the message on the bottom line of the screen with the "standout" character attribute turned on.

9.6 Options

For each option available through the ":set" command, Elvis contains a character array variable, named "o_option". For example, the "lines" option uses a variable called "o_lines".

For boolean options, the array has a dimension of 1.

> The first (and only) character of the array will be NUL if the variable's value is FALSE, and some other value if it is TRUE. To check the value, just by

dereference the array name, as in "if (*o_autoindent)".

For number options, the array has a dimension of 3.

The array is treated as three unsigned one-byte integers.The first byte is the current value of the option. The second and third bytes are the lower and upper bounds of that option. For string options, the array usually has a dimension of about 60 but this may vary. The option's value is stored as a normal NUL-terminated string.

All of the options are declared in "opts.c".

Most are initialized to their default values; the initopts() function is used to perform any environment-specific initialization.

9.7 Portability

To improve portability, Elvis collects as many of the system-dependent definitions as possible into the "config.h" file. This file begins with some preprocessor instructions which attempt to determine which compiler and operating system you have. After that, it conditionally defines some macros and constants for your system.

One of the more significant macros is ttyread(). This macro is used to read raw characters from the keyboard, possibly with timeout.For UNIX systems, this basically reads bytes from stdin. For MSDOS, TOS, and OS9, ttyread() is a function defined in curses.c. There is also a ttywrite() macro.

The tread() and twrite() macros are versions of read() and write() that are used for text files. On UNIX systems, these are equivalent to read() and write().On MS-DOS, these are also equivalent to read() and write(), since DOS libraries are generally clever enough to convert newline characters automatically.

For Atari TOS, though, the MWC library is too stupid to do this, so we had to do the conversion explicitly. Other macros may substitute index() for strchr(), or bcopy() for memcpy(), or map the "void" data type to "int", or whatever.

The file "tinytcap.c" contains a set of functions that emulate the termcap library for a small set of terminal types.The terminal-specific info is hard-coded into this file. It is only used for systems that don't support real termcap. Another alternative for screen control can be seen in the "curses.h" and "pc.c" files. Here, macros named VOIDBIOS and CHECKBIOS are used to indirectly call functions which perform low-level screen manipulation via BIOS calls.

The stat() function must be able to come up with UNIX-style major/minor/inode numbers that uniquely identify a file or directory.

Please try to keep you changes localized, and wrap them in #if/#endif pairs, so that Elvis can still be compiled on other systems. And PLEASE let me know about it, so I can incorporate your changes into my latest-and-greatest version of Elvis.

10. MAKEFILE

On most Operating Systems, and with most compilers, the "Makefile.mix" file is used to control compilation and installation of Elvis. This section of the manual describes the overall structure of "Makefile.mix", and the various configuration options in it.

10.1 Configuring the Makefile

Begin by copying "Makefile.mix" to "Makefile". Never alter the original "Makefile.mix". Most of the configuration options are controlled via a group of macros. Makefile.mix begins with several pre-configured sets of macro definitions - one group for each of the most common supported systems.As shipped, all of these macro definitions are commented out; you must either uncomment out one of the groups, or (for less common systems) construct an entirely new group.

10.2 Using the Makefile

After configuring the Makefile, you can run "make" to compile the programs. There are also some other useful things that the Makefile can do...

COMMAND RESULT

make compile all programs
make install copy the programs to the BIN directory
make clean remove all object files
make clobber remove everything except source & documentation
Note that the last two will probably work only under UNIX.

10.3 What "make install" does

To install elvis, we should copy all of the executables into a directory where users can find them; copy the documentation into a directory where the on-line manual program can find them; and arrange for edit buffers to be preserved after a system crash.The "make install" command tries to do this automatically, but there are problems. Practically all operating systems allow programs to be installed different directories. As shipped, Makefile.mix contains somebody's best guess as to where you'd like them to go. You should double check it, though. The BIN macro controls where the programs will be installed.

On UNIX systems the "elvprsv" and "elvrec" programs need to be installed as SUID-

root programs. Consequently, you must run "make install" as root; then they will automatically be installed as SUID-root.

For text to be recovered after a crash, you need to arrange for the "elvprsv" program to be run before the /tmp file is cleaned. This means that the /etc/rc file (or whatever) needs to be edited.If you have a SysV UNIX system which uses a /etc/rc2.d directory for storing start-up commands, then you're lucky. "make install" will detect that /etc/rc2.d exists and attempt to automatically create a file called "/etc/rc2.d/S03elvis" which runs elvprsv. However, for non-UNIX systems, or UNIX systems which don't have a /etc/rc2.d directory, you'll need to do this by hand. See the "Versions" section of the manual for hints about doing this on your particular system.

Non-UNIX systems don't have a standard place where UNIX-style man-pages go, so "make install" doesn't attempt to install documentation on those systems. On UNIX systems, there is no standard place either, but you can be pretty sure that your system has a non-standard one.

There is a shell script called "instman.sh" which attempts to figure out where the man-pages belong on your system, and then copies them there. You might need to edit "instman.sh" to make it work, but try it as-is first. "instman.sh" is automatically run by "make install".

Note: It is safe to run "make install" more than once.

10.4 Summary of Macros

The following describes the configuration macros. With most versions of make, a blank macro can simply be left undefined.

OBJ

> This is the filename extension for unlinked object files - usually .o, but MS-DOS uses .obj. EXE This is the filename extension for elvis executable file - usually nothing, but MS-DOS uses .exe, and other operating systems may use something else.

COM

> This is the filename extension for the executables of elvis' support programs - usually the same as the EXE macro, but since the support programs are all much smaller that elvis, MS-DOS can use the .com format.

EXTRA

This is a space-delimited list of version-specific object files to be linked into elvis. Typically, this list will contain at least one object file which was written specifically for a given operating system. It may also contain "tinytcap$(OBJ)" or "tinyprnt$(OBJ)".

EXTRA2

This is a space-delimited list of version-specific object files used in elvis and a few of the support programs. For UNIX-like systems, this is typically an empty list.For non-UNIX systems, it will usually either be empty, or it will contain the name of an object file which contains functions which emulate certain UNIX system calls. (Not all non-UNIX systems need any special emulation functions, because all C libraries try to emulate UNIX. You only need an EXTRA2 list if the library doesn't emulate UNIX well enough.)

LIBS

This is a list of library flags used while linking elvis. UNIX systems need "-ltermcap" or something similar, unless the EXTRA macro includes "tinytcap$(OBJ)". Most other operating systems use "tinytcap$(OBJ)" and don't need anything else, so they leave the LIBS list empty.

BIN

This is the directory where executables should be installed by "make install".

CC

This is the C compiler command, possibly with "memory model" flags.

CFLAGS

This lists the compiler flags used to select compile-time options. The "CFLAGS" section of this manual describes this in detail.

LNK

This is the name of the linker. If you want to use $(CC) as your linker, then you can leave LNK undefined.

LFLAGS

This is a list of linker flags used to select link-time options. It is almost always blank. SMALLThe flag for special small memory model compilation - usually blank.

OF

The link flag to control the output file's name - usually -o <space >.The Sun version of "make" strips off trailing whitespace, so a pair of empty quotes has been added after the space, to protect it. On non-Suns, this isn't necessary.

RF

The flag used to denote "compile but don't link" - usually -c

PROGS

This is a space-delimited list of all programs. This list always includes elvis, ctags, ref, elvrec, and elvprsv. Also, everybody gets fmt except for BSD UNIX; it already has its own version of fmt as standard equipment. Most non-UNIX systems also include the vi, ex, and view aliases. (UNIX doesn't need those aliases in the PROGS list because it creates them via file links during installation.) OS-9 doesn't include the ex alias, because there is already a command by that name built into its standard shell. Note: some MS-DOS configurations break this list into two smaller lists, to compensate for MS-DOS's limitations on command line length.

CHMEM

This is either blank, or a command to be run immediately after linking elvis. Under Minix and Coherent, elvis needs to have extra space assigned for the stack & heap after it has been linked, so their commands to do that are placed here. Most other operating systems generally either don't need to have their stacks enlarged, or they enlarge it during linking.

SORT

This should be defined to be -DSORT if you want your tags list to be sorted, or blank if you want it unsorted. The real vi requires a sorted tags file, so for the sake of compatibility all of the UNIX configurations use -DSORT. Elvis doesn't need a sorted tags file, though, so on non-UNIX systems you can leave this macro blank.

RM

This is the name of a program that deletes files unconditionally. It is used during "make clean". RM is defined as "rm -f" for UNIX systems, or "del" for most others.

CP

This is the name of a program that copies files. - usually "cp" or "copy". It is

used during "make install".

SYS

This is the type of system. It is used to select an appropriate style of linking and installation that are used by "make" and "make install", respectively. The available types are:

unx UNIX and UNIX-like systems
dos MS-DOS
ami AmigaDos
tos Atari TOS
os9 OS-9/68k
vms VAX/VMS
xdos cross-compiled on SCO for MS-DOS

DUMMY

This is used as the "source" filename in the dependency list of targets which are supposed to be unconditionally compiled.It is usually nothing since most versions of "make" treat an empty source file list as a special case, but OS-9 needs it defined as "dummy" and further requires that there be no actual file named dummy.

CFG

The is the name of the compiler configuration file - usually blank, since most compilers don't need a configuration file. Some MS-DOS compilers need it, though.

10.5 Structure of Makefile.mix

Makefile.mix begins with several sets of commented out configuration macro definitions, as described above. A comment saying "The rest of this Makefile contains no user-serviceable parts" marks the end of this section. Most people won't need to edit anything after that.

This is followed by macro definitions which are identical, regardless of your operating system. The OBJS macros list the object files that form the portable parts of elvis, and are used together with the EXTRA and EXTRA2 configuration macros during linking. The SRC macros list all of the files mentioned in the "MANIFEST" file.These are used to bundle the source code via "make uue" or "make sh".

This is followed by a target named "all" which depends on all of the programs listed in the PROGS configuration macro. This is followed by detailed instructions describing

how each file is compiled and linked. The only exceptions are the "elvis" program, and the various forms of the "alias" program. Linking a big program like elvis is non-standard on some systems.To support this, we just say that elvis depends on "linkelv.$(SYS)", where "$(SYS)" is replaced by whatever you defined the SYS configuration macro to be. The various link styles are listed after that. The only really tricky one is for DOS.

Since the list of files to be linked is too long to fit on a DOS command line, a customized response file is created, and the name of the response file is passed instead. The exact format of the response file depends on the compiler you're using. This is followed by system-dependent ways of linking the "alias" object file to create multiple executables. For most systems, we only really link it once to form the "ex" executable, and then copy that executable to form the "vi", "view", and "input" executables. OS-9, though, doesn't need an "ex" executable and it requires actual linking for each alias.

Next comes installation, in all its system dependent forms. This uses the now-familiar trick of saying that the "install" target depends on a bogus file named "inst.$(SYS)" and then listing each installation technique after that. There should be no surprises here. The rest of Makefile.mix contains a few handy pseudo-targets, such as "make clean".

11. CFLAGS

Elvis uses many preprocessor symbols to control compilation. Some of these control the sizes of buffers and such.The "-DNO_XXXX" options remove small sets of related features. Most Elvis users will probably want to keep all features available. Minix-PC users, though, will have to sacrifice some sets because otherwise Elvis would be too bulky to compile. The "asld" phase of the compiler craps out.

-DM_SYSV, -Dbsd, -DTOS, -DCOHERENT, -Damiga

> These flags tell the compiler that Elvis is being compiled for System-V UNIX, BSD UNIX, Atari TOS, Coherent, or Amiga-Dos, respectively. For other systems, the config.h file can generally figure it out automatically.

-DRAINBOW

> For MS-DOS systems, this causes support for the DEC Rainbow to be compiled into Elvis.

-DNO_S5WINSIZE

> Some versions of SysV UNIX don't support support the "winsize" style of screen-size testing. If you have a SysV system and can't compile"curses.c",then try adding -DNO_S5WINSIZE to the CFLAGS.

-DTERMIOS

POSIX is a SysV-derived specification which uses a terminal control package called "termios", instead of "termio". Some other SysV systems may also use termios. You can make elvis uses termios instead of the more common termio by adding -DTERMIOS to CFLAGS.(Note: This hasn't been tested very well.)

-DNBUFS=number

Elvis keeps most of your text in a temporary file; only a small amount is actually stored in RAM. This flag allows you to control how much of the file can be in RAM at any time. The default is 5 blocks, and the minimum is 3 blocks. (See the -DBLKSIZE flag, below.) More RAM allows global changes to happen a little faster. If you're just making many small changes in one section of a file, though, extra RAM won't help much.

-DBLKSIZE=number

This controls the size of blocks that Elvis uses internally. The value of BLKSIZE must be a power of two. Every time you double BLKSIZE, you quadruple the size of a text file that Elvis can handle, but you also cause the temporary file to grow faster. For MS-DOS, Coherent, and Minix-PC, the default value is 1024, which allows you to edit files up to almost 512K bytes long. For all other systems, the default value is 2048, which allows you to edit files that are nearly 2 megabytes long. The BLKSIZE also determines the maximum line length, and a few other limits. BLKSIZE should be either 256, 512, 1024, or 2048. Values other than these can lead to strange behavior.

-DTMPDIR=string

This sets the default value of the "directory" option, which specifies where the temporary files should reside. The value of TMPDIR must be a string, so be sure your value includes the quote characters on each end.

-DEXRC=str, -DHMEXRC=str, -DSYSEXRC=str, -DEXINIT=str

This lets you control the names of the initialization files. Their values must be strings, so be careful about quoting. EXRC is the name of the initialization file in the current directory. Its default value is ".exrc" on UNIX systems -- the same as the real vi. Since that isn't a legal DOS file- name, under DOS the default is "elvis.rc". For other systems, check the config.h file. HMEXRC is the name of the initialization file in your home directory. By default, it is the same as EXRC. Elvis will automatically prepend the name of your home directory to HMEXRC at run time, so don't give a full path name. SYSEXRC is the name of a system-wide initialization file. It has no default value; if you don't define a value for it, then the code that supports SYSEXRC just isn't compiled. The value of SYSEXRC should be a full pathname, in quotes. EXINIT is the name of an

environment variable that can contain initialization commands. Normally, its value is "EX-INIT".

-DKEYWORDPRG=string

This flag determines the default value of the "keywordprg" option. Its value must be a string, so be careful about quoting.The default value of this flag is "ref", which is a C reference program.

-DCC_COMMAND=string -DMAKE_COMMAND=string -DERRLIST=string

These control the names of the C compiler, the "make" utility, and the error output file, respectively. They are only used if -DNO_ERRLIST is not given. The default value of CC_COMMAND depends on the Operating System and compiler that you use to compile elvis; for UNIX, the default is "cc". The default values of MAKE_COMMAND and ERRLIST are "make" and "errlist", respectively.

-DMAXRCLEN=number

This determines how large a :@ macro command can be (measured in bytes). The default is 1000 bytes.If you increase this value significantly, then you may need to allocate extra memory for the stack. See the "CHMEM" setting in the Makefile.

-DSHELL=string

This is the default value of the "shell" option, and hence the default shell used from within Elvis.This only controls the default; the value you give here may be overridden at run-time by setting an environment variable named SHELL (or COMSPEC for MS-DOS). Its value must be a string constant, so be careful about quoting.

-DMAILER=string

This is the name of the program that Elvis uses to send mail to a user whose text has just been preserved. (See the manual page for the elvprsv program.) If your system doesn't use electronic mail, then this option is irrelevant. For UNIX and OS-9 systems, though, the value should be a quoted string.The default value is "mail", but SysV users may prefer to use "mailx", and BSD users may prefer "Mail".

-DTAGS=string

This sets the name of the "tags" file, which is used by the :tag command.Its value must be a string constant, so be careful about quoting.

-DCS_IBMPC -DCS_LATIN1 -DCS_SPECIAL

The digraph table and flipcase option will normally start out empty. However, if you add -DCS_IBMPC or -DCS_LATIN1 to your CFLAGS, then they will start out filled with values that are appropriate for the IBM PC character set or the ISO Latin-1 character set, respectively. You can also use -DCS_IBMPC and -DCS_SPECIAL together to get digraphs that produce the PC's graphic characters.

-DDEBUG -DEBUG2

-DDEBUG adds the ":debug" and ":validate" commands, and also adds many internal consistency checks.It increases the size of the ".text" segment by about 6K. -DDEBUG2 causes a line to be appended to a file called "debug.out" everytime any change is made to the edit buffer.

-DCRUNCH

This flag removes some non-critical code, so that Elvis is smaller.For example, it removes a short-cut from the regexp package, so that text searches are slower. Also, screen updates are not as efficient. A couple of obscure features are disabled by this, too.

-DNO_MKEXRC

This removes the ":mkexrc" command, so you have to create any .exrc files manually.The size of the .text segment will be reduced by about 1500 bytes.

-DNO_CHARATTR

Permanently disables the charattr option. This reduces the size of your ".text" segment by about 850 bytes.

-DNO_RECYCLE

Normally, Elvis will recycle space (from the temporary file) which contains totally obsolete text.This flag disables this recycling.Without recycling, the ".text" segment is about 1K smaller than it would otherwise be, but the tmp file grows much faster. If you have a lot of free space on your hard disk, but Elvis is too bulky to run with recycling, then try it without recycling. When using a version of Elvis that has been compiled with -DNO_RECYCLE, you should be careful to avoid making many small changes to a file because each individual change will cause the tmp file to grow by at least 1k.Hitting "x" thirty times counts as thirty changes, but typing "30x" counts as one change. Also, you should occasionally do a ":w" followed by a ":e" to start with a fresh tmp file.

Interestingly, the real vi never recycles space from its temporary file.

-DNO_SENTENCE

Leaves out the "(" and ")" visual mode commands. Also, the "[[", "]]", "{", and "}" commands will not recognize *roff macros. The sections and paragraphs options go away.This saves about 650 bytes in the ".text" segment.

-DNO_CHARSEARCH

Leaves out the visual commands which locate a given character in the current line: "f", "t", "F", "T", "," and ";". This saves about 900 bytes.

-DNO_EXTENSIONS

Leaves out the "K" and "#" visual commands. Also, the arrow keys will no longer work in input mode. Regular expressions will no longer recognize the \{\} operator. (Other extensions are either inherent in the design of Elvis, or are controlled by more specific flags, or are too tiny to be worth removing.) This saves about 250 bytes.

-DNO_MAGIC

Permanently disables the "magic" option, so that most metacharacters in a regular expression are *NOT* recognized. This saves about 3k of space in the ".text" segment, because he complex regular expression code can be replaced by much simpler code.

-DNO_SHOWMODE

Permanently disables the "showmode" option, saving about 250 bytes.

-DNO_CURSORSHAPE

Normally, Elvis tries to adjust the shape of the cursor as a reminder of which mode you're in. The -DNO_CURSORSHAPE flag disables this, saving about 150 bytes.

-DNO_DIGRAPH

To allow entry of non-ASCII characters, Elvis supports digraphs. A digraph is a single (non-ASCII) character which is entered as a combination of two other (ASCII) characters. If you don't need to input non-ASCII characters, or if your keyboard supports a better way of entering non-ASCII characters, then you can disable the digraph code and save about 450 bytes.

-DNO_ERRLIST

Elvis adds a ":errlist" command, which is useful to programmers. If you don't need this feature, you can disable it via the -DNO_ERRLIST flag. This will reduce the .text segment by about 900 bytes, and the .bss segment by about 300 bytes.

-DNO_ABBR

The -DNO_ABBR flag disables the ":abbr" command, and reduces the size of Elvis by about 250 bytes.

-DNO_OPTCOLS

When Elvis displays the current options settings via the ":set"command, the options are normally sorted into columns. The -DNO_OPTCOLS flag causes the options to be sorted across the rows, which is much simpler for the computer. The -DNO_OPTCOLS flag will reduce the size of your .text segment by about 500 bytes.

-DNO_MODELINES

This removes all support for modelines.

-DNO_TAG

This disables tag lookup. It reduces the size of the .text segment by about 750 bytes.

-DNO_TAGSTACK

This disables the tagstack. The ^T and :pop commands will no longer be available.

-DNO_ALT_FKEY, -DNO_CTRL_FKEY, -DNO_SHIFT_FKEY, -DNO_FKEY

These remove explicit support of function keys. -DNO_ALT_FKEY removes support for the *alternate* versions function keys. -DNO_CTRL_FKEY removes support for the *control* and *alternate* versions function keys. -DNO_SHIFT_FKEY removes support for the shift, control, and alternate. versions function keys. -DNO_FKEY removes all support of function keys. Elvis's ":map" command normally allows you to use the special sequence "# <n >" to map function key <n >. For example, ":map #1 {!}fmt^M" will cause the *F1* key to reformat a paragraph. Elvis checks the :k1=: field in the termcap description of your terminal to figure out what code is sent by the *F1* key. This is handy because it allows you to create a .exrc file which maps function keys the same way regardless of what type of terminal you use. That behavior is standard; most implementations of the real vi supports it too. Elvis extends this

to allow you to use "#1s" to refer to *shift+F1*, "#1c" to refer to *control+F1*, and "#1a" to refer to *alt+F1*. The termcap description for the terminal should have fields named :s1=:c1=:a1=: respectively, to define the code sent by these key combinations. (You should also have :k2=:s2=:c2=:a2=: for the *F2* key, and so on.) But there may be problems. The terminfo database doesn't support :s1=:c1=:a1=:, so no terminfo terminal description could ever support shift/control/alt function keys; so you might as well add -DNO_SHIFT_FKEY to CFLAGS if you're using terminfo.

Note that, even if you have -DNO_FKEYS, you can still configure Elvis to use your function keys my mapping the literal character codes sent by the key. You just couldn't do it in a terminal-independent way.

-DTERM_925, -DTERM_AMIGA, -DTERM_VT100, -DTERM_VT52, etc.

The tinytcap.c file contains descriptions of several terminal types. For each system that uses tinytcap, a reasonable subset of the available descriptions is actually compiled into Elvis.If you wish to enlarge this subset, then you can add the appropriate -DTERM_XXX flag to your CFLAGS settings. For a list of the available terminal types, check the tinytcap.c file.

-DINTERNAL_TAGS

Normally, Elvis uses the "ref" program to perform tag lookup.This is more powerful than the real vi's tag lookup, but it can be much slower. If you add -DINTERNAL_TAGS to your CFLAGS setting, then Elvis will use its own internal tag lookup code, which is faster.

-DPRSVDIR=directory

This controls where preserved files will be placed. An appropriate default has been chosen for each Operating System, so you probably don't need to worry about it.

-DFILEPERMS=number

This affects the attributes of files that are created by Elvis; it is used as the second argument to the creat() function. The default is 0666 which (on UNIX systems at least) means that anybody can read or write the new file, but nobody can execute it. On UNIX systems, the creat() call modifies this via the umask setting.

-DKEYBUFSIZE=number

This determines the size of the type-ahead buffer that elvis uses. It also limits the size of keymaps that it can handle.The default is 1000 characters, which should be plenty.

12. TERMCAP

Elvis uses fairly standard termcap fields for most things. I invented the cursor shape names and some of the function key names, but other than that there should be few surprises.

Required numeric fields

:co#: number of columns on the screen (chars per line)
:li#: number of lines on the screen

On many systems, Elvis has other ways to find out how many rows and columns your screen can show, so these values might not be very relevent. If these numbers aren't given in your termcap entry, and Elvis can't find the screen size any other way, then it will default to 80x24.

Required string fields

:ce=: clear to end-of-line
:cm=: move the cursor to a given row/column
:up=: move the cursor up one line

If these fields are missing, then Elvis will still run fairly well in "ex" mode, but "vi" mode requires these capabilities as an absolute minimum.

Boolean fields

:am:auto margins - wrap when char is written in last column?
:xn:brain-damaged auto margins - newline ignored after wrap
:pt:physical tabs?

Optional string fields

:al=: insert a blank row on the screen
:cl=: home the cursor & clear the screen
:dl=: delete a row from the screen
:cd=: clear to end of display
:ei=: end insert mode
:ic=: insert a blank character
:im=: start insert mode
:dc=: delete a character
:sr=: scroll reverse (insert row at top of screen)
:vb=: visible bell

:ks=: keypad enable
:ke=: keypad disable
:ti=: terminal initialization string, to start full-screen mode
:te=: terminal termination, to end full-screen mode

Optional strings received from the keyboard

:kd=: sequence sent by the <down arrow > key
:kl=: sequence sent by the <left arrow > key
:kr=: sequence sent by the <right arrow > key
:ku=: sequence sent by the <up arrow > key
:kP=: sequence sent by the <PgUp > key
:kN=: sequence sent by the <PgDn > key
:kh=: sequence sent by the <Home > key
:kH=: sequence sent by the <End > key
:kI=: sequence sent by the <Insert > key

Originally, termcap didn't have any names for the *PgUp*, *PgDn*, *Home*, and *End* keys. Although the capability names shown in the table above are the most common, they are not universal. SCO Xenix uses :PU=:PD=:HM=:EN=: for those keys. Also, if the four arrow keys happen to be part of a 3x3 keypad, then the five non-arrow keys may be named :K1=: through :K5=:, so an IBM PC keyboard may be described using those names instead. Elvis can find any of these names.

Optional strings sent by function keys

:k1=:...:k9=:k0=:codes sent by <F1 > through <F10 > keys
:s1=:...:s9=:s0=:codes sent by <Shift F1 > ... <Shift F10 >
:c1=:...:c9=:c0=:codes sent by <Ctrl F1 > ... <Ctrl F10 >
:a1=:...:a9=:a0=:codes sent by <Alt F1 > ... <Alt F10 >

Note that :k0=: is used to describe the *F10* key. Some termcap documents recommend :ka=: or even :k;=: for describing the *F10* key, but Elvis doesn't support that. Also, the :s1=:..., :c1=:..., and :a1=:... codes are very non-standard. The terminfo library doesn't support them. Consequently, if you're using the terminfo library then you might as well add NO_SHIFT_FKEY to your CFLAGS setting.

Optional fields that describe character attributes

:so=:se=:start/end standout mode (We don't care about :sg#:)
:us=:ue=:start/end underlined mode
:md=:me=:start/end boldface mode
:as=:ae=:start/end alternate character set (italics)
:ug#: visible gap left by :us=:ue=:md=:me=:as=:ae=:

Optional fields that affect the cursor's shape The :cQ=: string is used by Elvis

immediately before exiting to undo the effects of the other cursor shape strings. If :cQ=: is not given, then all other cursor shape strings are ignored.

:cQ=: normal cursor
:cX=: cursor used for reading EX command
:cV=: cursor used for reading VI commands
:cI=: cursor used during VI input mode
:cR=: cursor used during VI replace mode

If the capabilities above aren't given, then Elvis will try to use the following values instead.

:ve=: normal cursor, used as :cQ=:cX=:cI=:cR=: :vs=: gaudy cursor, used as :cV=:

An example

Here's the termcap entry I use on my Minix-ST system.

```
mx|minix|minixst|ansi:\
 :is=\E[0~:co#80:li#25:bs:pt:\
 :cm=\E[%i%d;%dH:up=\E[A:do=^J:nd=\E[C:sr=\EM:\
 :cd=\E[J:ce=\E[K:cl=\E[H\E[J:\
 :al=\E[L:dl=\E[M:ic=\E[@:dc=\E[P:im=:ei=:\
 :so=\E[7m:se=\E[m:us=\E[4m:ue=\E[m:\
 :md=\E[1m:me=\E[m:as=\E[1;3m:ae=\E[m:\
 :ku=\E[A:kd=\E[B:kr=\E[C:kl=\E[D:\
 :k1=\E[1~:k2=\E[2~:k3=\E[3~:k4=\E[4~:k5=\E[5~:\
 :k6=\E[6~:k7=\E[17~:k8=\E[18~:k9=\E[19~:k0=\E[20~:
```

13. ENVIRONMENT VARIABLES

Elvis examines several environment variables when it starts up. The values of these variables are used internally for a variety of purposes.You don't need to define all of these; on most systems, Elvis only requires TERM to be defined. On Amiga-DOS, MS-DOS or TOS systems, even that is optional.

TERM, TERMCAP

> TERM tells Elvis the name of the termcap entry to use. TERMCAP may contain either the entire termcap entry, or the full pathname of the termcap file to search through. If your version of Elvis is using tinytcap instead of the full termcap library, then the value of TERMCAP can't be the name of a file; it can only be undefined, or contain the entire termcap entry. In the termcap entry, tinytcap will convert \E to an <Esc > character, but other backslash escapes (\b, \r, etc.) or carat escapes (^[, ^M, etc.) will not be converted to control characters. Instead,

you should embed the actual control character into the string.

TMP, TEMP

These only work for AmigaDOS, MS-DOS and Atari TOS.Either of these variables may be used to set the "directory" option, which controls where temporary files are stored. If you define them both, then TMP is used, and TEMP is ignored.

LINES, COLUMNS

The termcap entry for your terminal should specify the size of your screen. If you're using a windowing interface, then there is an ioctl() call which will provide the size of the window; the ioctl() values will override the values in the termcap entry. The LINES and COLUMNS environment variables (if defined) will override either of these sources. They, in turn, can be overridden by a ":set" command. Normally, the LINES and COLUMNS variables shouldn't need to be defined.

EXINIT

This variable's value may contain one or more colon-mode commands, which will be executed after all of the ".exrc" files but before interactive editing begins. To put more than one command in EXINIT, you can separate the commands with either a newline or a '|' character.

SHELL, COMSPEC

You can use COMSPEC in MS-DOS, or SHELL in any other system, to specify which shell should be used for executing commands and expanding wildcards.

HOME

This variable should give the full pathname of your home directory. Elvis needs to know the name of your home directory so it can locate the ".exrc" file there.

TAGPATH

This variable is used by the "ref" program. It contains a list of directories that might contain a relevant "tags" file. Under AmigaDOS, MS-DOS or Atari TOS, the names of the directories should be separated by semicolons (";"). Under other operating systems, the names should be separated by colons (":"). If you don't define TAGPATH, then "ref" will use a default list which includes the current directory and a few other likely places.See the definition of DEFTAGPATH at the start of ref.c for an accurate list.

14. VERSIONS

Elvis currently works under BSD UNIX, AT&T System-V UNIX, SCO XENIX, Minix, Coherent, MS-DOS, Atari TOS, OS9/68k, VAX/VMS, AmigaDos, and OS/2. This section of the manual provides special information that applies to each particular version of Elvis. For all versions except MS-DOS and VMS, the file "Makefile.mix" should be copied to "Makefile", and then edited to select the correct set of options for your system. There is more information about this embedded in the file itself.

14.1 BSD UNIX

Temporary files are stored in /tmp.

You should modify /etc/rc so that the temp files are preserved when the system is rebooted. Find a line in /etc/rc which reads

ex4.3preserve /tmp

or something like that, and append the following line:

elvprsv /tmp/elv*

If you do not have permission to modify /etc/rc, don't fret. The above modification is only needed to allow you to recover your changes after a system crash. You can still run Elvis without that modification, and you can still recover your changes when Elvis crashes or when your dialup modem looses the carrier signal, or something like that. Only a system crash or power failure could hurt you.

Both Elvis and the real Vi read initialization commands from a file called ".exrc", but the commands in that file might work on one editor but not the other. For example, "set keyword prg=man" will work for Elvis, but Vi will complain because it doesn't have a "keywordprg" option. If the warning messages annoy you, then you can edit the CFLAGS setting in the Makefile and add -DEXRC=\".elvisrc\".

If you use X windows, you may wish to add "-DCS_LATIN1" to CFLAGS. This will cause the digraph table and the flipcase option to have default values that are appropriate for the LATIN-1 character set. That's the standard character set for X. The default mailer used notify users when text is preserver is "mail". You may wish to change this to "Mail" (with an uppercase 'M'). See the description of "MAILER" in the CFLAGS section of this manual.

The default keyboard macro time-out value is larger for BSD than it is for some other systems, because I've had trouble running Elvis via rlogin or Xterm. I guess it takes a while for those keystokes to squirt through the net.

14.2 System-V UNIX

Most SysV UNIX systems use terminfo instead of termcap, but the terminfo library doesn't seem to have a standard name. As shipped, Elvis' Makefile.mix is configured with "LIBS=-ltermcap".You may need to change it to "LIBS=-lterm" or "LIBS=-lterminfo" or even "LIBS=-lcurses".

The /etc/rc file (or its equivalent) should be modified as described for BSD systems, above. There's a pretty good chance that "make install" will do this for you; it knows how to create an editor recovery file in the /etc/rc2.d directory, which is where most modern SysV systems store initialization commands. You only need to do it manually for older SysV systems. The potential trouble with ".exrc" described above for BSD UNIX applies to System-V UNIX as well.

The default mailer used notify users when text is preserver is "mail". You may wish to change this to "mailx". See the description of "MAILER" in the CFLAGS section of this manual. Elvis uses control-C as the interrupt key, not Delete. This was done so that the <Del > key could be used for character deletion.

14.3 SCO Xenix

For Xenix-386, you can use the generic System-V settings. You may wish to add "-DCS_IBMPC" to CFLAGS, to have the digraph table and flipcase option start up in a mode that is appropriate for the console.Also, note that there is a separate group of settings for use with Xenix-286.

It already has "-DCS_IBMPC" in CFLAGS. Because Xenix is so similar to System-V, everything I said earlier about System-V applies to the Xenix version too, except that edit or recovery might belong in a directory called /etc/rc.d/8 instead.

14.4 Minix

There are separate settings in Makefile.mix for Minix-PC and Minix-68k.The differences between these two are that the 68k version uses ".o" for the object file extension where the PC version uses ".s", and the PC version has some extra flags in CFLAGS to reduce the size of Elvis. The PC version also uses tinytcap (instead of the full termcap) to make it smaller.

Minix-PC users should read the CFLAGS section of this manual very carefully. You have some choices to make...

The temporary files are stored in /usr/tmp.The /usr/tmp directory must exist before you run Elvis, and it must be readable & writable by everybody. We use /usr/tmp instead

of /tmp because after a system crash or power failure, you can recover the altered version of a file from the temporary file in /usr/tmp.If it was stored in /tmp, though, then it would be lost because /tmp is normally located on the RAM disk. Also, you'll need a /usr/preserve directory which is readable & writable by root; this directory is used to store text files that have been preserved after a crash. The "make install" script will create it if necessary. Elvis uses control-C as the interrupt key, not Delete.

14.5 Coherent

Elvis was ported to Coherent by Esa Ahola. Elvis is too large to run under Coherent unless you eliminate some features via the CFLAGS setting. The recommended settings, in Makefile.mix, produce a working version of Elvis which emulates Vi faithfully, but lacks most of the extensions. You should read the CFLAGS section of this manual carefully. You can probably reduce the size of Elvis by using tinytcap.c instead of -lterm. This would allow you to keep most features of Elvis, at the expense of terminal independence. (Tinytcap.c has ANSI escape sequences hard-coded into it.) To use tinytcap, just add "tinytcap.o" to the "EXTRA=" line in the Makefile, and remove "-lterm" from the "LIBS=" line.

The temporary files are stored in /tmp. Preserved files are stored in /usr/preserve. You should modify your /etc/rc file to support file preservation; add the line ... /usr/bin/elvprsv /tmp/* ... just before the first "/bin/rm" line.

14.6 Linux

The Makefile.mix file has a special section of options for Linux. There should be no surprises. Linux is mostly SysV-ish, so the SysV comments above will apply to Linux as well, except that most Linux systems still have an old-style /etc/rc file.

You should add the command

... /usr/bin/elvprsv /tmp/*

in there somewhere. On my SLS 1.02 system, I added it near the end, just before the line that runstheshellon "/etc/rc.local".

14.7 MS-DOS

Elvis was ported to MS-DOS by Guntram Blohm and Martin Patzel. Willett Kempton added support for the DEC Rainbow.

Ideally, Elvis should be compiled with Microsoft C 5.10 and the standard Microsoft Make utility, via the command "make elvis.mak". This will compile Elvis and all related utilities.

With Microsoft C 6.00, you may have trouble compiling regexp.c. If so, try compiling

it without optimization. The "Makefile.mix" file contains a set of suggested settings for compiling Elvis with Turbo-C or Borland C. (If you have Turbo-C, but not the Make utility, then you can almost use the "Elvis.prj" file to compile Elvis, but you must explicitly force Turbo-C to compile it with the "medium" memory model. Most of the related programs [ctags, ref, virec, refont, and wildcard] are only one file long, so you should have no trouble compiling them.) The "alias.c" file is meant to be compiled once into an executable named "ex.exe".You should then copy "ex.exe" to "vi.exe" and "view.exe". Elvis stores its temporary files in C:\tmp. If this is not satisfactory, then you should edit the CFLAGS line of your Makefile to change TMPDIR to something else before compiling.You can also control the name of the temp directory via an environment variable named TMP or TEMP. The directory must exist before you can run Elvis.

The **TERM** environment variable determines how Elvis will write to the screen. It can be set to any one of the following values:

pcbios Use BIOS calls on an IBM-PC clone.
rainbowUse DEC Rainbow interface.
ansi Use ANSI.SYS driver.
nansiUser faster NANSI.SYS driver.

If the **TERM** variable isn't set, then Elvis will automatically select either the "rainbow" interface (when run on a Rainbow) or "pcbios" (on an IBM clone).

You may prefer to use NANSI.SYS for speed; or you may NEED to use ANSI.SYS for a non-clone, such as a lap-top. If so, you should install one of these drivers by adding "driver = nansi.sys" (or whatever) to your CONFIG.SYS file, and then you should define TERM to be "nansi" (or whatever) by adding "set TERM=nansi" to your AUTOEXEC.BAT file. You must then reboot for these changes to take effect. After that, Elvis will notice the "TERM" setting and use the driver.

Since ".exrc" is not a valid DOS filename, the name of the initialization file has been changed to "elvis.rc". Elvis will look for an "elvis.rc" file first in your home directory. If it exists, and contains ":set exrc", then Elvis will check for another "elvis.rc" in the current directory. By default, the directory where ELVIS.EXE resides is taken to be your home directory. You can override this default by setting an environment variable named "HOME" to the full pathname of your home directory. To set "HOME", you would typically add the following line to your AUTOEXEC.BAT file:

set HOME c:\

An extra program, called "wildcard", is needed for MS-DOS. It expands wildcard characters in file names. If Elvis flashes a "Bad command or filename" message when it starts, then you've probably lost the WILDCARD.EXE program somehow.

Elvis can run under Windows, but you may have trouble with TEMP. Windows uses an environment variable called TEMP which interferes with Elvis' usage of TEMP; to

work around this, you can simply set an environment variable named TMP (with no 'E') to the name of Elvis' temporary directory. When TEMP and TMP are both set, Elvis uses TMP and ignored TEMP.

In a ":set" command, the backslash character is used to "escape" the character that follows it. To make a backslash be part of a string option's value, you must enter a double backslash. For example, to define the directory where temporary files will exist, you could add ":set dir=C:\\tmp" to your ELVIS.RC file. Just plain ":set dir=C:\tmp" (with one backslash) won't work!.

14.8 Atari TOS

Elvis was ported to Atari TOS by Guntram Blohm and Martin Patzel.It is very similar to the MS-DOS version. It has been tested with the Mark Williams C compiler and also GNU-C. The TERM environment variable is ignored; the ST port always assumes that TERM=vt52. The SHELL (not COMSPEC!) variable should be set to the name of a line-oriented shell.

A simple shell in included with Elvis.Its source is in "shell.c", and the name of the executable is "shell.ttp". The file "profile.sh" should contain a set of instructions to be executed when the shell first starts up. An example of this file is included, but you will almost certainly want to edit it right away to match your configuration. (If you already have a command-line shell, then you'll probably want to continue using it. The shell that comes with Elvis is very limited.) Currently, character attributes cannot be displayed on the screen.

Elvis runs under MiNT (a free multi-tasking extension to TOS) but it can be a CPU hog because of the way that Elvis reads from the keyboard with timeout. Also, Elvis doesn't use any of the special features of MiNT. I have received a set of patches that optimize Elvis for MiNT, but they arrived too late to integrate into this release.

14.9 OS9/68k

Elvis was ported to OS9/68k by Peter Reinig. The Makefile is currently configured to install Elvis and the related programs in /dd/usr/cmds If this this is unacceptable, then you should change the BIN setting to some other directory. Similarly, it expects the source code to reside in /dd/usr/src/elvis; the ODIR setting is used to control this.

Temporary files are stored in the /dd/tmp directory. Your /dd/startup file may need to be modified to prevent it from deleting Elvis' temporary files; make /dd/startup run the elvprsv program before it wipes out /dd/tmp.

The program in alias.c is linked repeatedly to produce the "vi", "view", and "input" aliases for Elvis. Sadly, the "ex" alias is impossible to implement under OS9 because the shell has a built-in command by that name. For some purposes, you must give

`make' the "-b" option. Specifically, you need this for "make -b clean" and "make -b install".

14.10 VAX/VMS

John Campbell ported Elvis to VAX/VMS. A heavily laden VAX can take half an hour to compile Elvis. This is normal. Don't panic.

While running, Elvis will create temporary files in SYS$SCRATCH. Enter SHOW LOGICAL SYS$SCRATCH to see what actual directory you are using. Many sites have SYS$SCRATCH equivalenced to SYS$LOGIN.The Elvis temporary files look like the following on VMS while Elvis is running:

ELV_1123A.1;1 ELV_1123A.2;1 SO070202.;1

Also, filtering commands (like !!dir and !}fmt) should work on VMS.This assumes, however, that you can create temporary mailboxes and that your mailbox quota (a sysgen parameter) is at least 256 bytes for a single write to the mailbox. This is the default sysgen parameter, so there should be few people who experience filter problems.

Additionally, an attempt was made to support the standard terminals on VMS: "vt52", "vt100", "vt200", "vt300", "vt101", "vt102".Non-standard terminals could be supported by setting your terminal type to UNKNOWN (by entering SET TERM/UNKNOWN) and defining the logical name ELVIS_TERM. Whatever ELVIS_TERM translates to, however, will have to be included in tinytcap.c.

Note hat the upper/lowercase distinctions are significant, and that DCL will upshift characters that are not quoted strings, so enter DEFINE ELVIS_TERM "hp2621a". As distributed, it would probably not be a good idea to have more than the standard terminals in tinytcap.c (else it wouldn't be tiny, would it?). Changes here, of course, would require a recompilation to take effect. If you have a version of the "termcap" library and database on your system, then you may wish to replace tinytcap with the real termcap.

14.11 AmigaDOS

Mike Rieser and Dale Rahn ported Elvis to AmigaDOS.

The port was done using Manx Aztec C version 5.2b.Elvis uses about as much space as it can and still be small code and data. Elvis should also compile under DICE, though there may be a little trouble with signed versus unsigned chars.

The port has been done so the same binary will run under both versions of AmigaDOS. Under AmigaDOS 2.04, Elvis supports all the documented features. It also uses an external program ref to do tag lookup. So, the accompanying programs: ref and ctags

are recommended. Under AmigaDOS 1.2/1.3 Elvis works, buts lacks the more advanced features.

For the port to AmigaDOS 2.04, we tried to use as many Native AmigaDOS calls as we could. This should increase Elvis's chances at being compiled with other compilers.DICE seems to have a different default char type. You may need to use the UCHAR() macro in tio.c. To test it, try the :map command; if it looks right, things are cool.

For the port to AmigaDOS 1.3, we tried to make sure the program was at least usable. Many features are missing, most notably running commands in subshells. Also, what we could getworking, we used Aztec functions to support them, so this part is little more compiler dependent.

Aztec is compatible with the SAS libcall #pragma. I personally prefer using the includes that come from Commodore over the ones supplied with Aztec, but for people with a straight Aztec installation, I went with the default names for the Aztec pragmas.

One include you'll need is <sys/types.h >. It's a common include when porting software just make yourself one. It's a two line file that saves a lot of hassle especially in the Elvis source. So, make a directory where your includes are located called `sys' and in a file below that type:

```
/* sys/types.h */
#include <exec/types.h >
```

When setting environment variables (either local or global) for variables that specify a directory, make sure the variable ends in `:' or `/'. This saved from having to change much of the way Elvis works. The default temporary directory (if TEMP and TMP aren't specified) is "T:". The default if HOME directory (if no HOME environment variable is set) is "S:".

To avoid conflict with other uses, Elvis uses elvis.rc instead of .exrc or where it looks for macros.

14.12 OS/2 2.x

Elvis was ported to OS/2 by Kai Uwe Rommel. Greg Roelofs fixed some generic bugs and added various tweaks and VIO features not supported by OS/2's ANSI emulation. Elvis was ported using the emx port of the GNU C compiler ("emx+gcc"); other OS/2 compilers (including the gcc/2 port) will probably not work due to their lack of termcap support. If you use emx 0.8f or earlier, you will need to change the -Zmtd option to -Zmt in Makefile.mix, as noted in the comment there.

The port is derived from the MS-DOS port, so most of the MS-DOS comments should still be valid. In particular, the default pathnames for the temp directory and

preservation directory are the same, and wildcard.exe is used. (emx's _wildcard() function is [optionally] used in most places, but ex.c still calls the "standard" wildcard() function.)

Note that the executables are dynamically linked, so you'll need EMX.DLL and EMXLIBC.DLL somewhere in your LIBPATH.These are included with executables-only distributions and, of course, with emx itself. You'll also need a termcap.dat file in a location pointed at by the TERMCAP variable; again, one is supplied. Set the TERM variable equal to one of the listed termcap entries such as "ansi" or "pcbios".

Finally, note that the VIO features mentioned above are not enabled by default. Without them, Elvis uses only termcap codes for screen updates and can therefore be used remotely (e.g., in a telnet session). If you're working at the OS/2 system console, however, the VIO features can be enabled via a special Elvis variable, "viomode" (abbreviated "vm"). Although it can be used interactivelyto enable most of the new features (smooth backscrolling, cursor shape in insert mode, shorter beeps), enabling both the auto-detection of ANSI mode (plus setting it, if necessary) and the restoration of screen colors requires the EX-INIT variable to be set. (By the time Elvis is running and the interactive command is given, it's too late for such initializations.)For example, add "set EXINIT=set viomode" to your config.sys file; this will take effect at the next OS/2 reboot.

14.13 Other Systems

For SunOS and Solaris 1.x, use the BSD settings; for Solaris 2.x, use the SysV settings. Earlier versions of Elvis didn't link correctly due to a quirk in Sun's version of the "make" utility, but this version of Elvis has a work-around for that quirk so you should have no trouble at all. For AIX, use the SysV settings in Makefile.mix, with the changes suggested by comments there.

For other UNIXoid systems, I suggest you start with the Minix-68k settings and then grow from that.Minix is a nice starting point because it is a clone of Version 7 UNIX, which was the last common ancestor of BSD UNIX and SysV UNIX. Any operating system which claims any UNIX compatibility whatsoever will therefore support V7/Minix code. You may need to fiddle with #include directives or something, though. Minix-68k is a better starting point than Minix-PC because the PC compiler has some severe quirks.

If you're thinking of porting Elvis to some non-UNIX system, I suggest you begin by studying the "INTERNALS" section of this manual.

15. QUESTIONS & ANSWERS

1) How can I make Elvis run faster under DOS?

There are several things you can do. The first thing to do is get a good screen driver such as NANSI.SYS.This can speed up screen redrawing by as much as a factor of eight! The DOS-specific part of section 12 tells you how to do this. You might also consider reducing the size of the blocks that Elvis uses. You'll need to recompile Elvis to do this.The default BLKSIZE is 2048 bytes for the DOS version of Elvis, which means that for each keystroke that you insert, Elvis must shift an average of about 1000 bytes. That's a lot to ask from a little old 5MHz 8088.

A BLKSIZE of 512 bytes might be more appropriate. A "write-back" disk cache can help. DOS 5.0 and Windows 3.0 come with one of these, called SMARTDRV.EXE.I suggest you add "SMARTDRV C+" to your AUTOEXEC.BAT file. If you're really desperate for more speed, you might want to make Elvis store its temporary files on a RAM disk. However, this limits the size of the file you can edit, and it eliminates any chance you may have had to recover your work after a power failure or system crash, but it might be worth it; you decide.To do this, add ":set dir=R:\" (or whatever your RAM disk's name is) to the elvis.rc file. Next, consider turning off the "sync" option.When the sync option is turned on, Elvis will close the temporary file and reopen it after every change, in order to force DOS to update the file's directory entry. If you put ":set nosync" into the elvis.rc file, then Elvis will only close the file when you start editing a different text file, or when you're exiting Elvis.Consequently, there is no chance that you'll be able to recover your changes after a power failure... so if you're going to this, then you might as well store the temp files on the RAM disk, too.

2) Why isn't Elvis reading my .exrc (or ELVIS.RC) file?

For security reasons, Elvis doesn't normally look for a .exrc file in the current directory. A mean-spirited person could lay a trap by placing a dangerous command in a .exrc file in some public directory.

If you know for a fact that no such person has ever used your computer, then you can make elvis process the .exrc file from the current directory by adding "set exrc" to either your EXINIT environment variable or to the .exrc file in your home directory.

3) What is my home directory?

On UNIX systems, each user is given a private "home" directory. Among other uses, this is used for storing personal configuration files for use with various programs. UNIX stores the pathname of your home directory in an environment variable called "HOME". DOS doesn't have UNIX-style home directories, but if you explicitly set HOME to the name of some directory, then Elvis will still look for its ELVIS.RC file there. If HOME is unset, then elvis will look for ELVIS.RC in the same directory where ELVIS.EXE was found.

4) Where's the *Esc* key on a DEC keyboard?

I don't know. Maybe the *F11* key? Maybe Control-3? You could always use ":map!" to

make some other key act like the *Esc* key. If all else fails, try *Control-[*..

5) Is there a way to show which keys do what?

Yes.The command ":map" will show what each key does in command mode, and ":map!" (with an exclamation mark) shows what each key does in input mode. The table is divided into three columns: the key's label, the characters that it sends, and the characters that Elvis pretends you typed.

6) How can I make Elvis display long lines like the real vi?

You can't yet. The next version of Elvis should support this, though.

7) I can't recover my text [under MS-DOS or Atari TOS]. According to the directory listing, the temporary file is 0 bytes long. What went wrong?

MS-DOS and TOS only update a file's directory entry when the file is closed. If the system crashes while the file is still open, then the file's length is stored as 0 bytes.The ":set sync" option is supposed to prevent this; you probably turned it off in the interest of speed, right?

Under MS-DOS [I don't know about TOS], you should delete the empty temporary file, and then run CHKDSK/F. This might find the data that belonged in the empty file, and place it in a new file with a name like "000001.CHK" -- something like that. You can then try to extract the text from that temporary file by giving the command "elvprsv -R 000001.chk".If you're lucky, then this might recover your text.

8) What is the most current version of Elvis?

Each version of Elvis that is released to the public has a version number of the form "number point number". As I write this, the most current version of Elvis is 1.8. The intermediate steps between one release and the next are labeled with the next version number, with a letter appended. For example, after 1.4 was released, I started working on 1.5a.I am currently working on 2.0a. When Elvis reaches a stable state, I'll call it 2.0 and release it. Sometimes a beta-test version of Elvis will be available via anonymous FTP from m2xenix.psg.com, in the directory "pub/elvis/beta".

9) I only got executables, but now I want the source code. Where can I get it?

If you have access to the Internet, then you should be able to fetch it from one of the public archives such as plains.nodak.edu. It is accessible via anonymous FTP, or via an email servernamed"archive-server@plains.nodak.edu". Elvis is located in the directory

"/pub/Minix/all.contrib".

It is also available from the C Users' Group, in volume #365. As I write this, they are asking $4 per disk plus $3.50 per order in the US, and elvis requires three disks; this is

subject to change. Their phone number is (913) 841-1631, and their address is:

The C Users' Group
1601 W. 23rd Street, #200
Lawrence KS 66046-2743

10) Is this shareware, or public domain, or what?

It is not public domain; it is copyrighted by me, Steve Kirkendall. However, this particular version is freely redistributable, in either source form or executable form. (I would prefer that you give copies away for free, complete with the full source code... but I'm not going to force you.)

It is not shareware; you aren't expected to send me anything. You can use it without guilt.

It is not "copylefted." I hold a copyright, but currently I have not added any of the usual restrictions that you would find on copylefted software. If people start doing really obnoxious things to Elvis, then I will start adding restrictions to subsequent versions, but earlier versions won't be affected. (So far, everybody has been pretty good about this so no restrictions have been necessary.)

11) Can I reuse parts of your source code?

Yes.Please be careful, though, to make sure that the code really is mine. Some of the code was contributed by other people, and I don't have the authority to give you permission to use it. The author's name can be found near the top of each source file. If it says "Steve Kirkendall" then you may use it; otherwise, you'd better contact the author first.

Please don't remove my name from the source code. If you modify the source, please make a note of that fact in a comment near the top of the source code. And, finally, please mention my name in your documentation.

12) Can Elvis work with non-ASCII files?

Elvis is 8-bit clean. This means that Elvis will allow you to edit files that use a European extended ASCII character set. However, some terminals are not 8-bit clean; they treat characters in the range 0x80-0x9f as control characters. Elvis expects all characters above 0x7f to be treated as normal displayable characters, so on these terminals Elvis may produce a scrambled display.

Elvis can't edit binary files because it can't handle the NUL character, and because of line-length limitations. Elvis has also modified to work with 16-bit character sets, but that modification is not part of the standard Elvis distribution.Yongguang Zhang (ygz@cs.purdue.edu) has created a Chinese version of Elvis that uses 16-bit characters and runs under cxterm (Chinese X-term) on X-windows systems. Junichiro Itoh

(itojun@foretune.co.jp) has modified Elvis to edit Japanese text under MS-DOS.

+command or -c command

If you use the +command parameter, then after the first file is loaded command is executed as an EX command. A typical example would be "elvis +237 foo", which would cause elvis to start editing foo and then move directly to line 237. The "-c command" variant was added for UNIX SysV compatibility.

FILES

/tmp/elv*

During editing, elvis stores text in a temporary file.For UNIX, this file will usually be stored in the /tmp directory, and the first three characters will be "elv". For other systems, the temporary files may be stored someplace else; see the version-specific section of the documentation.

tags

This is the database used by the :tags command and the -t option.It is usually created by the ctags(1) program.

.exrc or elvis.rc

On UNIX-like systems, a file called ".exrc" in your home directory is executed as a series of ex commands.A file by the same name may be executed in the current directory, too. On non-UNIX systems, ".exrc" is usually an invalid file name; there, the initialization file is called "elvis.rc" instead.

ENVIRONMENT

TERM

> This is the name of your terminal's entry in the termcap or terminfo database. The list of legal values varies from one system to another.

TERMCAP

> Optional. If your system uses termcap, and the TERMCAP variable is unset, then will ready our terminal's definition from /etc/termcap. If TERMCAP is set to the full pathname of a file (starting with a '/') then will look in the named file

instead of /etc/termcap. If TERMCAP is set to a value which doesn't start with a '/', then its value is assumed to be the full termcap entry for your terminal.

TERMINFO

Optional.If your system uses terminfo, and the TERMINFO variable is unset, then will read your terminal's definition from the database in the /usr/lib/terminfo database. If TERMINFO is set, then its value is used as the database name to use instead of /usr/lib/terminfo.

LINES, COLUMNS

Optional. These variables, if set, will override thescreensizevalues given in the termcap/terminfo for your terminal. On windowing systems such as X, has other ways of determining the screen size, so you should probably leave these variables unset.

EXINIT

Optional. This variable can hold EX commands which will be executed before any .exrc files.

SHELL

Optional. The SHELL variable sets the default value for the "shell" option, which determines which shell program is used to perform wildcard expansion in file names, and also which is used to execute filters or external programs. The default value on UNIX systems is "/bin/sh".

Note: Under MS-DOS, this variable is called COMSPEC instead of SHELL.

HOME

This variable should be set to the name of your home directory. Looks for its initialization file there; if HOME is unset then the initialization file will not be executed.

TAGPATH

Optional.This variable is used by the "ref" program, which is invoked by the shift-K, control-], and :tag commands. See "ref" for more information.

TMP, TEMP

These optional environment variables are only used in non-UNIX versions of . They allow you to supply a directory name to be used for storing temporary files.

BUGS

There is no LISP support.Certain other features are missing, too. Auto-indent mode is not quite compatible with the real vi. Among other things, 0^D and ^^D don't do what you might expect. Long lines are displayed differently. The real vi wraps long lines onto multiple rows of the screen, but elvis scrolls sideways.

ELVPRSV

Preserve the the modified version of a file after a crash.

SYNOPSIS

elvprsv ["-why elvis died"] /tmp/filename... elvprsv -R /tmp/filename...

DESCRIPTION

elvprsv preserves your edited text after elvis dies.The text can be recovered later, via the elvprsv program. For UNIX-like systems, you should never need to run this program from the command line. It is run automatically when elvis is about to die, and it should be run (via /etc/rc) when the computer is booted. THAT'S ALL! For non-UNIX systems such as MS-DOS or VMS, you can either use elvprsv the same way as under UNIX systems (by running it from your AUTOEXEC.BAT file), or you can run it separately with the "-R" flag to recover the files in one step.

If you're editing a file when elvis dies (due to a bug, system crash, power failure, etc.) then elvprsv will preserve the most recent version of your text. The preserved text is stored in a special directory; it does NOT overwrite your text file automatically. (If the preservation directory hasn't been set up correctly, then elvprsv will simply send you a mail message descrining how to manually run elvprsv.)

elvprsv will send mail to any user whose work it preserves, if your operating system normally supports mail.

FILES

/tmp/elv*

The temporary file that elvis was using when it died.

/usr/preserve/p*

The text that is preserved by elvprsv.

/usr/preserve/Index

A text file which lists the names of all preserved files, and the names of the /usr/preserve/p* files which contain their preserved text.

BUGS

Due to the permissions on the /usr/preserve directory, on UNIX systems elvprsv must be run as superuser. This is accomplished by making the elvprsv executable be owned by "root" and turning on its "set user id" bit.

ELVREC

Recover the modified version of a file after a crash

SYNOPSIS

Elvrec [preservedfile [newfile]]

DESCRIPTION

If you're editing a file when elvis dies, the system crashes, or power fails, the most recent version of your text will be preserved. The preserved text is stored in a special directory; it does NOT overwrite your text file automatically.

The elvrec program locates the preserved version of a given file, and writes it over the top of your text file -- or to a new file, if you prefer.The recovered file will have nearly all of your changes. To see a list of all recoverable files, run elvrec with no arguments.

(Note: if you haven't set up a directory for file preservation, then elvis' you'll have to manually run the elvprsv program instead of elvrec.)

FILES

/usr/preserve/p*

The text that was preserved when elvis died.

/usr/preserve/Index

A text file which lists the names of all preserved files, and the names of the /usr/preserve/p* files which contain their preserved text.

BUGS

elvrec is very picky about filenames. You must tell it to recover the file using exactly the same pathname as when you were editing it. The simplest way to do this is to go into the same directory that you were editing, and invoke elvrec with the same filename as elvis. If that doesn't work, then try running elvrec with no arguments, to see exactly which pathname it is using for the desired file. Due to the permissions on the /usr/preserve directory, on UNIX systems elvrec must be run as superuser. This is accomplished by making the elvrec executable be owned by "root" and setting its "set user id" bit. If you're editing a nameless buffer when elvis dies, then elvrec will pretend that the file was named "foo".

FMT

Adjust line-length for paragraphs of text

SYNOPSIS

fmt [-width] [files]...

DESCRIPTION

fmt is a simple text formatter. It inserts or deletes newlines, as necessary, to make all lines in a paragraph be approximately the same width. It preserves indentation and word spacing. The default line width is 72 characters. You can override this with the -width flag. If you don't name any files on the command line, then fmt will read from stdin.

It is typically used from within vi to adjust the line breaks in a single paragraph. To do this, move the cursor to the top of the paragraph, type "!}fmt", and hit [Return]

REF

Display a C function header

SYNOPSIS

ref [-t] [-x] [-c class]... [-f file]... tag

DESCRIPTION

ref quickly locates and displays the header of a function. To do this, ref looks in the "tags" file for the line that describes the function, and then scans the source file for the

function. When it locates the function, it displays an introductory comment (if there is one), the function's declaration, and the declarations of all arguments.

SEARCH METHOD

ref uses a fairly sophisticated tag look-up algorithm. If you supply a filename via -f file, then elvis first scans the tags file for a static tag from that file. This search is limited to the tags file in the current directory.

If you supply a classname via -c class, then elvis searches for a tag from that class. This search is not limited to the current directory; You can supply a list of directories in the environment variable TAGPATH, and ref will search through the "tags" file in each directory until it finds a tag in the desired class. If that fails, ref will then try to look up an ordinary global tag. This search checks all of the directories listed in TAGPATH, too. If the tag being sought doesn't contain any colons, and you haven't given a -x flag, then any static tags in a tags file will be treated as global tags.

If you've given the -t flag, then ref will simply output the tag line that it found, and then exit. Without -t, though, ref will search for the tag line. It will try to open the source file, which should be in the same directory as the tags file where the tag was discovered. If the source file doesn't exist, or is unreadable, then ref will try to open a file called "refs" in that directory.

Either way, ref will try to locate the tag, and display whatever it finds.

INTERACTION WITH ELVIS

ref is used by elvis' shift-K command. If the cursor is located on a word such as "splat", in the file "foo.c", then elvis will invoke ref with the command "ref -f foo.c splat".

If elvis has been compiled with the -DEXTERNAL_TAGS flag, then elvis will use ref to scan the tags files.This is slower than the built-in tag searching, but it allows elvis to access the more sophisticated tag lookup provided by ref. Other than that, external tags should act exactly
like internal tags.

OPTIONS

-t

Output tag info, instead of the function header.

-f file

The tag might be a static function in file.You can use several -f flags to have ref consider static tags from more than one file.

-c class

The tag might be a member of class class. You can use several -c flags to have ref consider tags from more than one class.

FILES

tagsList of function names and their locations, generated by ctags. refs Function headers extracted from source files (optional).

ENVIRONMENT

TAGPATH

List of directories to be searched.The elements in the list are separated by either semicolons (for MS-DOS, Atari TOS, and AmigaDos), or by colons (every other operating system). For each operating system, ref has a built-in default which is probably adequate.

NOTES

You might want to generate a "tags" file the directory that contains the source code for standard C library on your system.If licensing restrictions prevent you from making the library source readable by everybody, then you can have ctags generate a "refs" file, and make "refs" readable by everybody. If your system doesn't come with the library source code, then perhaps you can produce something workable from the lint libraries.