Elwin by Little Wing

 Contents for Elwin Help

Overview

Elwin is based upon the Unix VI editor and requires a bit of understanding about VI's history and modes of operation to use effectively. VI is descended from another Unix editor known as EX which was itself descended from ED (I think). Both ED and EX are line editors that compensated for the lack of a full-screen interface with sophisticated and powerful search, movement, and editing commands. Eventually Unix became blessed with the curses terminal i/o library which made it possible to write a full screen editor without dealing with the myriad terminal types that had been hooked up to Unix. VI was born. Rather than throw out the effort that had been put into ED and EX (by both the programmers and the user community who learned them), VI opted to completely internalize EX and enhance it with a full screen interface. VI also added a host of its own commands.

VI has four main modes of operation:

Insert mode which pretty much allows full screen editing of the current file. In this mode Elwin behaves like any normal Windows editor.

Command mode in which most keystrokes are interpreted as some part of a command. Within Elwin, this mode is referred to as VI mode.

Colon mode which is entered by typing a colon in command mode. Most EX commands can be entered in this mode.

EX mode in which VI behaves exactly like EX. Elwin does not support this mode at all.

Elwin has been designed not to completely internalize all VI functionality but rather to make the VI user feel comfortable in the Windows environment while taking advantage of the new interface possibilities offered by the GUI environment.

Menu Commands

VI Mode Reference

VI Mode Overview

Cool Stuff You May Have Missed

These are random odds and ends covered elsewhere, but that you may have missed and are well worth knowing.

The file name patterns used in the Open File Dialog are in your control. See Configuration and Command Line.

You can print a piece of a file by selecting the piece with the mouse and then executing File Print.

The Replace All command on the Search Replace dialog is also constrained to the current selection if there is one.

Using Ctrl-Tab and Ctrl-Shift-Tab works just like the Windows desktop. Windows are kept in the order they were most recently used. Pressing Ctrl-Tab displays the second most recently used window; if you continue to hold down the Ctrl key and press Tab again, the third most recently used will be brought to front. Continuously holding down the Ctrl key and repeatedly pressing Tab will cycle through all non-minimized windows and finally back to the beginning. When you release the Ctrl key, the currently displayed window moves to the front of the list as the most recently used. Also pressing the Shift key causes Elwin to go backwards through the list. This MRU algorithm is not used by the EX next command.

Pressing Tab when there is text selected in line mode will cause all selected lines to be indented; Shift-Tab unindents all selected lines.

Elwin maintains a session profile for each directory in which it is started. See also Configuration and Command Line.

The EX set directory command controls where temporary files are created.

Elwin responds to Drag and Drop from File Manager.

Clicking the right mouse button during a drag toggles between line mode selections and normal selections.

Registration

Elwin is distributed as shareware. It is not free or public domain. You may copy and distribute Elwin freely, but if you use it for more than 30 days you are obligated to pay a license fee of $50, or the equivalent in local currency, or anything else we deem to be worth about $50.

If you have an account on CompuServe it's very easy to register Elwin on-line. Simply GO SWREG and follow the instructions to register 5483. You will be billed on you CompuServe account and will be notified by Little Wing of your registration number usually within 24 hours by CompuServe mail.

You may also register Elwin by authorizing Little Wing to bill your Visa or Mastercard account. Just give us your account number and expiration date via mail, e-mail, or telephone. We will bill your account $50 for each license and notify you of your registration codes. Checks and P.O.'s can also be accepted. You can reach us at:

71532.403@compuserve.com

OR

Little Wing
4618 JFK Blvd., Ste. 176
N. Little Rock, AR 72116

OR

(501) 771-2408

Payment options

Especially you people outside the U.S., if you'd like to send something swell from your culture that is hard to get in Arkansas (just about everything worth having), feel free to do so. I'll give you a license for it unless it's a total crock. Allow me to suggest,

Czechs	Becherovka or Fernet
Belgians	Kriek lambic is $7.00 a bottle here!
Germans	I wonder if some bratwurst would make it?
French	I can't find decent Camembert...
Mexicans	You wouldn't believe what we pay for tequila and Kahlua
English	Maybe you'd better send money... It's a joke! I'm just kidding!

Anyway, you get the idea. Be creative!

File Menu Commands

New	Open a window for a new file.
Open...	Open a file and a new window. You can change the file open filter.
Close	Close the current window and file.
Close All	Close all windows.
Save	Save the contents of the current window. Colon mode write offers greater flexibility.
Save As...	Save the contents of the current window to the specified file.
Save All	Save the contents of all modified windows.
Refresh	Forget any changes and reload current file from disk. You can turn off confirmation.
Refresh All	Refresh all modified windows.
Info...	Display number of bytes and lines in current window.
Print...	Send the contents of the current window to the printer. If text is selected, only it will be printed.
Exit	End the Elwin session.
Most Recent Files	Open a window for the specified file.

Colon Mode
Refers to the mode in which you can execute EX commands. This mode is entered by typing a colon from VI mode. A dialog box will appear. See also EX Command Reference.

VI Mode
The mode in which most key strokes are interpreted as VI commands. Press the escape key to enter VI mode. Note: the **VI Mode Never** switch in the Preferences menu must be turned off. See also VI Command Overview.

Insert Mode
The mode in which keystrokes translate to characters inserted into the file. This is "normal" editor behavior. Insert mode can be entered most easily by typing an **i** from VI mode or by clicking the mouse in the edit window (unless Mouse Click Exits VI in the Preferences menu is not checked).

The Status Bar

Shows the Row.Column position of the caret. The top line is 1; the leftmost position is 0.

Shows whether the current window is in VI mode. It is dimmed for insert mode.

Shows whether the current file has been modified. If so an **M** is displayed; otherwise the **M** is dimmed.

Shows the current count state if any. Many VI commands accept a count as part of the command. If a count has been entered it will be displayed here. Press escape to clear the count (and all VI state). See also VI Fundamentals.

Shows the current command state if any. Several VI commands require more than one keystroke, particularly movement commands. This cell shows **Scroll** if Elwin is in a scroll state, **Mark** if waiting for a mark character, **Jump** if waiting for a mark character to jump to, **Find Char** if waiting for a character to search for on the line, **Buffer** if waiting for a buffer character, and **Replace** if waiting for a character to replace. Press escape to clear the state. See also Scrolling, Absolute Movement, and Contextual Movement.

Shows the current maipulator state if any. Shows **Delete**, **Change**, **Yank**, **Shift <**, or **Shift >** depending upon which manipulator is currently being entered. Press escape to clear the state. See also Manipulator Overview.

Shows which buffer is current. The **-** character indicates the clipboard. See also Manipulators and Paste.

Shows the time of day.

Shows the date.

Edit Menu Commands

Undo Line	Restore the current line to its initial state before changes.
Undo	Undo the last edit. Undo is itself an edit. Pressing it again undoes the undo.
Cut	Cut the current selection to the clipboard.
Copy	Copy the current selection to the clipboard.
Paste	Paste the contents of the clipboard to the current insert point.
Delete	Delete the current selection.

Search Menu Commands

Find...	Displays the find dialog. See also Regular Expressions.
Find Next	Searches for the next occurrence of the current search pattern.
Find Previous	Searches for the previous occurrence of the current search pattern.
Replace...	Displays the replace dialog. Replace All is confined to the current selection if there is one. Regular Expressions can greatly enhance Replace All.
Match Delim	Locates the delimiter which matches the one to the right of the cursor. If there is no delimiter to the right, the delimiter to the left is used. If a matching delimiter does not exist Elwin will beep. Delimiters are: **(**, **)**, **[**, **]**, **{**, **}**, **<**, and **>**.
Go To Line...	Jumps to the given line. The first line is 1. Entering 0 jumps to the last line.

Preferences Menu Commands

Auto Indent	Causes new lines to be indented with the same leading white space as the preceding line (or the following line if the **O** command is used).
Confirm On Refresh	Causes a confirmation dialog to be displayed before a refresh.
Allow Duplicates	Allows multiples windows to be opened for the same file.
Set Tab Width...	Sets the number of columns between tab stops.
Font...	Sets the font.
VI Mode Default	Causes all newly opened windows to be in VI mode.
VI Mode Never	Inhibits VI mode.
Mouse Click Exits VI	If this is checked, clicking the left mouse button will always leave Elwin in insert mode.
Retain Unix Format	DOS and Unix have different line delimiting standards. If this is checked, Elwin will retain the Unix format when saving any files that had it upon loading.

Window Menu Commands

Tile As Rows	Tiles all windows from top to bottom in a single column.
Tile As Columns	Tiles all windows from left to right in a single row.
Cascade	Cascades all open windows.
Display Window	Causes the specified window to be displayed as the top window.

Help Menu Commands

About Elwin...	Display information about Elwin.

VI Fundamentals

The Elwin VI mode borrows heavily and faithfully from the standard Unix editor. It is a common experience among Unix programmers to be initially intimidated by what seems to be a bewildering array of terse commands that turn the keyboard into a minefield; however, that VI has remained the standard among Unix professionals serves as testimony that these fears give way to a satisfied mastery of what is actually a very efficient, effective, and even elegant set of commands.

The design of the VI command set is narrowly focused on the programmer who is a touch typist and who spends as much time manipulating text as actually typing it in. A central design goal is that the fingers should not have to stray from the home keys to do any text manipulation or cursor movement. With this in mind, we can group VI commands into basically three categories: scrolling commands, movement commands, and manipulation commands. Commands from these categories are combined to produce complete command sequences that can easily carry out virtually any sort of edit that can be done with a mouse, but without moving the hands from the home keys.

There are only five manipulation commands: **d** for delete text; **c** for change text; **y** for yank text; and **>** & **<** for shift text left and right. To carry out an edit in VI mode you simply type a manipulation command and follow it with a movement command. The affected text is that between the initial cursor position and the position after the move. For example, **w** is a movement command that moves the cursor to the beginning of the next word. So to delete from the current position to the beginning of the next word, you simply type **dw**. The **L** movement command moves the cursor to the last line in the window, so to delete from the current position to the end of the window, type **dL**.

There are only five manipulators but there are (depending on how you count them) over 20 one-stroke movement commands. Plus all movement commands may be combined with a count to repeat them an arbitrary number of times, e.g. **d5w** will delete the next 5 words. Also, any manipulator can be followed by itself to imply line mode, e.g. **dd** will delete the current line; **yy** will copy the current line to the clipboard; and **<<** will shift the current line left one tab stop. These are some of the simplest examples, but you will find that once you master only a few movement commands you will begin to abandon your mouse in favor of a quick edit at the keyboard.

Finally, Elwin adds to the VI mode the feature most sorely lacking and probably most criticized about VI, a status bar. Elwin continually displays whether you are in VI mode and the state of the current command sequence. You may cancel any VI command sequence at any time by pressing **Escape**.

Where other professional editors give you a sophisticated macro capability and a large set of cumbersome commands that cannot be combined, Elwin gives you the simple syntax of **{count}<manipulator>{count}<movement>** and a concise set of instructions that alleviate the need to program your editor.

Entering and Leaving VI Mode

Pressing the **Escape** key is the only way to enter VI mode. If Elwin is in insert modeand the **VI Never** option in the **Preferences** menu is not checked, pressing the **Escape** key will put Elwin into VI mode with no state and cause the VI indicator to be displayed in the status bar. If Elwin is already in VI mode, pressing the **Escape** key causes the VI state to be cleared.

The simplest way to exit VI mode is to simply click the mouse anywhere in an edit window. This causes Elwin to flush all VI state and enter insert mode which behaves exactly like any other standard Windows editor. **Note:** The **Preferences** menu controls whether a mouse click leaves VI mode.

The insert, append, replace, and open commands provide alternative methods to exit VI mode. The insert command **i** behaves exactly like clicking the mouse at the current insert point. The **I** command moves the cursor to the first non-space character of the current line and then enters insert mode. The append command **a** moves the cursor right one space and enters insert mode; **A** moves the cursor to the end of the line and enters insert mode. The open command **o** inserts a blank line after the current line, auto-indenting if the **Autoindent** option is checked, positions the cursor at the end of the new line and enters insert mode; the **O** command does the same thing except the new line is inserted above the current line. The auto-indent will always be comprised of the leading white space from the current line.

Additionally the replace command **r** will enter insert mode for a single keystroke to replace the character to the right of the cursor with the next character typed. The **R** command will put Elwin in overstrike mode in which all keystrokes will overwrite the character at the cursor.

All of these commands ignore any count.

The change manipulator **c** also causes Elwin to enter insert mode. After the change command sequence, the text to be changed will be selected and Elwin will enter insert mode. The first character typed will cause the selected text to be replaced with the character; further typing will be inserted as usual. For example, typing **2cw** in VI mode will cause the next two words to be selected; the next characters typed will replace the words with the characters typed.

Scrolling

Elwin provides a number of ways to scroll the window in either insert mode or VI mode. All work in either mode. First you may use the scroll bars which behave like any Windows program. Second you may use the **Page Up** and **Page Down** keys to scroll the window backward and forward one full screen. **Ctrl-End** positions the cursor at the end of the file. **Ctrl-Home** first positions the cursor at the upper left corner of the window; a second press positions the cursor at the beginning of the file.

In addition to these there are six standard VI commands for scrolling the window. **Ctrl-E** scrolls the window up one line causing a new line to appear at the bottom of the window; **Ctrl-Y** scrolls the window down one line. **Ctrl-F** scrolls the window forward by one screen minus two lines; **Ctrl-B** scrolls the window backward by one screen minus two lines. **Ctrl-U** scrolls the window up by half a screen; **Ctrl-D** scrolls the window down by half a screen.

All of these commands recognize a count in VI mode. For all except **Ctrl-U** and **Ctrl-D**, these simply cause the command to be repeated the number of times given. For **Ctrl-U** and **Ctrl-D** a count specifies the number of lines to scroll. By default this amount is half the lines that fit in a window. After a count has been given with **Ctrl-U** or **Ctrl-D**, it becomes the default until another count is given.

While all of these commands work in either insert or VI mode, a count can only be given in VI mode.

The last scroll command is **z** which must be followed by **.**, **-**, **Return**, or **Ctrl-M**. **z.** scrolls the current line to the middle of the window; **z-** scrolls the current line to the bottom of the window; and both **z Return** and **z Ctrl-M** scroll the current line to the top of the window. If a count is given, the line specified becomes the current line before the scroll takes place.

All of the keyboard commands ensure that the cursor remains in the window after the scroll; the scroll bar commands do not reposition the cursor.

Absolute Movement

Elwin supports the arrow keys in both VI mode and insert mode for absolute movement. In VI mode several additional keystrokes may be used that don't require movement away from the home keys. The commands **j**, **Ctrl-N**, and **Ctrl-J** all have the same effect as the down arrow; the commands **k** and **Ctrl-P** are the same as the up arrow; the commands **h**, **Ctrl-H**, and **Backspace** behave the same as the left arrow; and the commands **l** and **Spacebar** are the same as the right arrow.

The commands **+** and **Ctrl-M** moves the cursor to the first non-space character on the next line; the **-** command moves the cursor to the first non-space character on the previous line.

The **|** command moves the cursor to the column specified by the count. The **^** command moves the cursor to the first non-space character on the current line. The **0** command moves the cursor to the beginning of the line. The **$** command moves the cursor to the end of the line.

The **G** command moves to the line specified by the count. For example, **200G** moves the cursor to line 200. If no count is specified, **G** moves the cursor to the last line of the file.

The **m** command marks the current position. Follow the command with a single lower case letter to identify the mark. For example **ma** records the current position as mark **a**. The ` (back quote) command jumps to the mark specified after it. For example **`a** would jump to the position recorded under mark **a**.The **'** (single quote) character also jumps to the specified mark, but positions the cursor at the first non-space character on the line. Either jump command may be followed by itself, such as ``, to move the cursor to the previous context, which is defined to be the last position from which you executed a non-relative move. A non-relative move is basically any move for which there is no simple sequence to get back; for example **100G** may move you from line 237 to line 100. To return to line 237 simply type **''**. The one caveat to be aware of with marks is that they are implemented as a (line, column) pair. Any edits that add or remove lines also fix up the mark positions; however, edits that alter line lengths do NOT fix up the mark positions. A mark may become invalid in such instances.

Contextual Movement

The first group of contextual movement commands move the cursor by words. The three lower-case commands **w**, **e**, and **b** move the cursor to the beginning of the next word, the end of the current word, and the beginning of the current word respectively. The beginning of a word is defined to be the first non-space character following a sequence of valid C-symbol characters or a sequence of non-valid C-symbol characters. A C-symbol character is any character that may appear in a symbol as defined by the **C** language. These characters are **A-Z**, **a-z**, **0-9**, and **_**. Experiment with the **w** command to get a feel for what a word is. The three related upper case commands **W**, **E**, and **B** work the same but with a different definition of word. They define the beginning of a word to be the first non-space character following the first space character that follows the current position. **Ctrl-Right Arrow** and **Ctrl-Left Arrow** carry out the same functions as **W** and **B** respectively. These definitions, while precise, may not give you a good feel for the commands. The best way to understand them is to experiment a bit. You will find them to be quite comprehensible in no time.

The two commands **[** and **]** move the cursor to the previous or next section respectively. A section is defined to be a line that begins with a **{**. Both of these commands will use a count if given. The **%** will find the character that matches the one near the cursor; characters that may be matched are **[{(<**and **]})>**.

The commands **H**, **M**, and **L** move the cursor to the first non-space character of the top line, the middle line, and the last line in the window respectively.

The commands **f** and **F** move the cursor to the next occurrence of the given character on the line. For example, **fa** will move the cursor forward to the next **a** on the line. **F** works the same way but searches backward. Both commands leave the cursor just beyond the character in the direction of the search, i.e. **f** leaves the cursor to the right of the hit and **F** leaves the cursor to the left of the hit. The commands **t** and **T** work similarly but leave the cursor just before the hit, i.e. **t** leaves the cursor to the left of the hit and **T** leaves the cursor to the right of the hit. Finally, the commands **;** and **,** repeat the previous find or to command, **;** in the same direction, **,** in the opposite direction. All of these commands may be used with a count. None of them will move cursor from the current line. These commands are particularly useful in combination with the **c** or change manipulator.

The last group invokes the regular expression search function. The **/** command causes a small dialog to appear into which you may type a regular expression or use the combo box to select an old search string; you may use the arrow keys to scroll through old search strings without dropping the combo box. Upon pressing **Return** Elwin will search forward in the file for the first match of the pattern. The **?** command does the same but searches backward. The **n** command repeats the previous search in the same direction. The **N** command repeats the previous search in the opposite direction. None of these commands use a count.

Manipulators

A manipulator is a command that works with a block of text. A manipulator must always be followed by a movement command. The block of text is defined to be from the cursor position where the command was begun to the position after the movement command. For example **d** is the delete manipulator, and **$** moves the cursor to the end of the line; so **d$** will delete the text from the current cursor position to the end of the line. Any movement command may follow a manipulator. If the movement command uses a count, it may be given either before or after the manipulator; for example, **100yG** and **y100G** will both yank (copy) the text from the current cursor position to line 100 into the clipboard. Most movement commands work exactly as they would if you were not using a manipulator although a few are treated as special cases to make their behavior more "reasonable" with a manipulator. For the most part, you probably won't even notice these special cases because they exist only to make Elwin "do the right thing". Also a few movement commands cause the manipulator to work in "line mode", i.e. complete lines will be manipulated rather than just the block of text. These line mode movement commands are: **j**, **+**, **k**, **-**, **G**, **'**, **`**, **[**, **]**, **H**, **M**, **L**, and any synonyms for these.

The five manipulators are **y**, **d**, **c**, **<**, and **>**, which correspond to yank, delete, change, left shift, and right shift. Yank copies the selected text to the clipboard. Delete cuts the selected text to the clipboard. Change selects the text and replaces it with the next characters typed. The shift commands move the selected lines left or right by one tab stop; these two always work in line mode.

Instead of following a manipulator with a movement command, you may simply repeat it to imply line mode. For example **yy** will yank the current line to the clipboard and **dd** will delete the current line. The behavior may be modified with a count. For example **5yy** will yank the current line and the next 4 lines to the clipboard.

By default, the yank and delete manipulators place the affected text in the clipboard; however, Elwin maintains 26 named buffers which may also receive text from these manipulators. You may specify a buffer by using the " command followed by a single lower case letter. The current buffer is always displayed in the status bar. The Windows clipboard may be selected by entering "". If you specifiy a buffer with an upper case letter, the selected text will be appended to the current contents of the buffer instead of replacing them. Text may be retrieved from a buffer by using the **p** or paste command.

In addition to these buffers, Elwin also maintains nine automatic delete buffers named 1-9 and specified like any other buffer. Any type of deleted text is placed in buffer 1. The next time a delete occurs, the contents of each buffer are shifted to the next higher numbered buffer. When text is shifted from buffer 9 it is permanently lost. These buffers can only be specified for a paste operation.

Manipulator Abbreviations

There are a few abbreviations for manipulators with an implicit block to make common edits easier. All of them may be modified with a count. They are **C** and **D** which change or delete to the end of the line respectively; they are the same as **c$** and **d$**; **Y** is the same as **yy** and will yank the current line; **s** is the same as **cl** which will replace the current character with the next characters typed; **S** is the same as **cc** which will replace the entire line with the next characters typed; **x** is the same as **dl** and will delete the character to the right of the cursor; and **X** is the same as **dh** and will delete the character to the left of the cursor.

The **x** and **s** commands behave subtly different from their non-abbreviated forms. If the cursor is at the end of the line, these two commands will affect the character to the left of the cursor rather than just beep at you.

Undo, Redo, Paste, Join, Toggle Case, etc.

The two paste commands **p** and **P** insert text from the current buffer at the cursor. If the text in the buffer was put there by a line mode manipulator it will be pasted in line mode; **p** will insert below the current line and **P** will insert above the current line. If the text in the buffer was put there by a non-line mode manipulator, it will be inserted exactly at the insertion point; in this case the only difference between **p** and **P** is the final position of the cursor.

The two undo commands **u** and **U** undo the last edit and undo all edits to the current line respectively. The last edit is defined to be the last manipulator command (not including yank), the last join, toggle case, or replace, the last continuous piece of text typed, or the last EX command. If the **u** command is repeated it undoes the last undo.

The redo command **.** is perhaps the most useful command in the entire Elwin command set. It causes the last edit as defined above to be repeated at the current cursor position. This can be very useful when combined with the change manipulator. A count may be specified with redo; if no count is given the count originally given is used.

As a special case, if the paste command is repeated with redo and the specified buffer was a delete buffer, the buffer number will be incremented for each redo.

The toggle case command **~** simply changes the case of the character to the right of the cursor if it is a letter and advances the cursor. It may be used with a count.

The join command **J** by default concatenates the current line and the following line. All white space between the two lines is removed and a single space is inserted. If a count *n* is given, it causes *n* lines to be joined.

The write and exit command **ZZ** will write all unsaved changes and exit Elwin.

Manipulator Menu Commands

The manipulator menu provides all of the functionality of the Elwin manipulators from the mouse instead of from the keyboard. Pressing the right mouse button causes it to appear.

Close	Closes the manipulator menu.
Delete	Deletes the selected text and places it in the current buffer. A line mode delete may be specified by pressing the right mouse button during the selection.
Yank	Yanks the selected text to the current buffer. A line mode yank may be specified by pressing the right mouse button during the selection.
Put	Pastes the text from the current buffer. Equivalent to the **p** command.
Undo	Undoes the the last edit. Equivalent to the **u** command.
Shift <<	Shifts the selected text left one tab stop. Equivalent to the **<** manipulator.
Shift >>	Shifts the selected text right one tab stop. Equivalent to the **>** manipulator.
Set Buffer	Displays the buffer selection submenu which allows the current buffer to be specified.

Scrolling Reference

Ctrl-E	Scroll <count> lines onto the bottom of the window.
Ctrl-Y	Scroll <count> lines onto the top of the window.
Ctrl-D	Scroll <count> lines onto the bottom of the window.
Ctrl-U	Scroll <count> lines onto the top of the window. If no count is given, the count last given is used. If no count has been given, half the window is scrolled.
Ctrl-F	Scroll <count> pages forward.
Ctrl-B	Scroll <count> pages backward. A page is one window minus two lines.
z	Must be followed by **Return**, **Ctrl-M**, **.**, or **-**. **Return** and **Ctrl-M** scroll the current line to the top of the window. **.** scrolls the current line to the middle of the window. **-** scrolls the current line to the bottom of the window. A count jumps to the specified absolute line and then positions as above.

See also Scrolling Overview.

Insert Reference

a	Move cursor right and enter insert mode.
A	Move cursor to end of line and enter insert mode.
i	Enter insert mode.
I	Move cursor to the first non-space character and enter insert mode.
o	Open a new line below the current line.
O	Open a new line above the current line.
r	Replace the character to the right of the cursor with the next one typed.
R	Enter overstrike mode in which all keystrokes will replace the character under the cursor.

See also Entering and Leaving VI Mode.

Absolute Movement Reference

j, Down Arrow, Ctrl-N, Ctrl-J
Move the cursor <count> lines down.

k, Up Arrow, Ctrl-P
Move the cursor <count> lines up.

h, Left Arrow, Ctrl-H, Backspace
Move the cursor <count> characters left.

l, Right Arrow, Spacebar
Move the cursor <count> characters right.

+, Ctrl-M Move the cursor <count> lines down to the first non-space character on the line.

- Move the cursor <count> lines up to the first non-space character on the line.

| Move the cursor to the column specified by the count.

^ Move the cursor to the first non-space character on the current line.

0 Move the cursor to the first character on the current line.

$ Move the cursor to the end of the line.

G Move the cursor to the absolute line specified by the count. If no count is given, move the cursor to the last line of the file.

m Record the current position under the mark letter following the **m** command.

**`** Jump to the position specified by the mark letter following the ` command.

' Jump to the line specified by the mark letter following the ' command. If no mark is specified, jump to the previous context.

See also Absolute Movement Overview.

Contextual Movement Reference

w	Move the cursor to the beginning of the next word.
e	Move the cursor forward to the end of a word.
b	Move the cursor backward to the beginning of the previous word. The beginning of a word is defined to be the first non-space character following a sequence of valid C-symbol characters or a sequence of non-valid C-symbol characters. A C-symbol character is any character that may appear in a symbol as defined by the **C** language. These characters are **A-Z**, **a-z**, **0-9**, and **_**.
W	Move the cursor to the beginning of the next full word.
E	Move the cursor forward to the end of a full word.
B	Move the cursor backward to the beginning of the previous full word. The beginning of a full word is defined to be the first non-space character following the first non-space character.
[	Move the cursor to the previous section.
]	Move the cursor to the next section. A section is marked by a line that begins with a { character.
%	Move the cursor to the character that matches the delimiter near the cursor. Delimiters are [{(< and]})>.
H	Move the cursor to the first non-space character of the top line in the window.
M	Move the cursor to the first non-space character of the middle line in the window.
L	Move the cursor to the first non-space character of the bottom line in the window.
f*c*	Move the cursor right past the <count>'th occurrence of *c*.
F*c*	Move the cursor left past the <count>'th occurrence of *c*.
t*c*	Move the cursor right to the <count>'th occurrence of *c*.
T*c*	Move the cursor left to the <count>'th occurrence of *c*.
;	Repeat the last **f**, **F**, **t**, or **T** command.
,	Repeat the last **f**, **F**, **t**, or **T** command in the opposite direction.
/regexp	Search forward for the given regular expression.
?*regexp*	Search backward for the given regular expression.
n	Repeat the last **/**, or **?** command.
N	Repeat the last **/**, or **?** command in the opposite direction.

See also Contextual Movement Overview.

Manipulator Reference

d	Delete the specified text.
c	Change the specified text.
y	Yank the specified text.
<	Shift the specified text left.
>	Shift the specified text right.

See also Manipulator Overview.

Manipulator Abbreviation Reference

C	Change to the end of line - **c$**.
D	Delete to the end of line - **d$**.
s	Change next character - **cl**.
S	Change line - **cc**.
x	Delete next character - **dl**.
X	Delete previous character - **dh**.
Y	Yank line - **yy**.

See also Manipulator Abbreviation Overview.

Miscellaneous Reference

p	Put the contents of the current buffer after the current position.
P	Put the contents of the current buffer before the current position.
"*c*	Select buffer *c* as the current buffer. *c* may be a-z, A-Z, or 1-9. A-Z causes text put in buffer to be appended to the current contents.
u	Undo the last edit.
U	Undo all edits to current line.
.	Redo the last edit.
~	Toggle the case of the character to the right of the cursor.
J	Join <count> lines together.
ZZ	Save all unsaved changes and exit.

See also Undo, Redo, Paste, Join, Toggle Case, etc. Overview.

Commands Outside VI Mode

Ctrl-Left Arrow
Move left one word.
Ctrl-Right Arrow
Move right one word.

Ctrl-Up Arrow
Scroll window one line up.
Ctrl-Down Arrow
Scroll window one line down.

Home Moves to first non-white space character on line. Second press moves to column 0.
Ctrl-Home Moves to upper left corner of window. Second press moves to first character in file.

End Moves to end of line.
Ctrl-End Moves to last character in file.

Keypad - Deletes current line. Can be used with a count in VI mode.
Keypad * Copies current line. Can be used with a count in VI mode.
Keypad + Pastes from current buffer. Same as VI **p**.

Tab Shifts the selected text one tab stop right in line mode.
Shift-Tab Shifts the selected text one tab stop left in line mode.

Insert Toggles overstrike mode and insert mode.

Regular Expression Reference

The following sequences have special meaning in a regular expression:

- **^** Matches the beginning of a line and is only special if it is the first character in a regular expression
- **$** Matches the end of a line and is only special if it is the last character in a regular expression
- **\<** Matches the beginning of a word
- **\>** Matches the end of a word
  These two can be used together to avoid getting hits on words that are parts of other words, e.g. **\<the\>** will match **the** but will not match toge**the**r.
- **.** Matches any single character
- **[** *character set* **]**
  A set of characters in square brackets **[ ]** matches any single character from the set. The **-** character can be used to denote a range of characters, e.g. **[a-z]** matches any lower-case letter. If the first character in the brackets is **^** then the expression matches any single character **not** contained in the set.
- **\{** *n* **\}**
  Designates closure of *n* occurrences and must follow an expression *c* that matches a single character. It modifies the previous expression so that the two together only match *n* occurences of *c*. For example **a\{5\}** matches 5 a's. The expression **\<[a-zA-Z]\{4\}\>** matches four letter words.
- **\{** *n, m* **\}**
  Designates closure of *n* to *m* occurrences on the preceding single-character expression. If *m* is omitted, it is taken to mean infinity.
- * Designates closure of 0 to infinity occurrences on the preceding single-character expression.
- **\+** Designates closure of 1 to infinity occurrences on the preceding single-character expression.
- **\?** Designates closure of 0 or 1 occurrence on the preceding single-character expression.
- **\(** *subexpression* **\)**
  Matches the *subexpression*. This is used to refer to pieces of a matched string in a replacement string.
- **\t** Matches a tab

The following sequences have special meaning in a replace expression:

- **&** Replaced by the entire matched text
- **~** Replaced by the entire previous replacement text
- *n* Replaced by the *n*th subexpression from the matched text. Consider the expression
  m_\([a-zA-Z]*\)
  and the replace expression
  M_\1
  This would replace words like m_blah and m_foo with M_blah and M_foo.
  See **\(** *subexpression* **\)** above.
- **\u** Converts the first character of any subsequent *n* to upper case
- **\l** Converts the first character of any subsequent *n* to lower case
- **\U** Converts all the characters of any subsequent *n* to upper case
- **\L** Converts all the characters of any subsequent *n* to lower case
- **\E** Turns off **\U** and **\L**

EX Command Reference

EX commands may be issued from VI mode by typing the **:** character. This will cause the EX command dialog to be displayed.

EX commands are of the general form

[*linespec, ...***] [***command***] [!] [***parameters***]**

Each command uses these parts differently. Some commands provide defaults for parts, ignore parts, or consider it an error to include or omit a part. These details are explained seperately with each command.

Each command may be preceded by zero, one, or two line specifiers depending upon the command. A line specifier is an expression made of the following elements:

.	The current line, i.e. the line with the caret
n	A number indicating an absolute line number; the first line in the file is 1
$	The last line in the file
'*m*	The line referenced by mark *m*
/pat/	The first line after the caret position containing *pat*
?*pat***?**	The first line preceding the caret position containing *pat*

The above values may be modified by

+*n*	Indicates *n* lines down
-*n*	Indicates *n* lines up

Each command below indicates the number of line specifiers it takes and the defaults it supplies.

The symbol **%** may be used as shorthand for the line specifiers **., $**, i.e. the entire file. A line specifier with no command will move the cursor to the given line.

a[bbreviate] *lhs rhs*
[.,.] co[py] *linespec*
e[dit] [!] [*file*]
[1,$] g[lobal] [!] */pat/command*
[.]k *x*
[.,.] m[ove] [*linespec*]
[.,.] p[rint]
q[uit] [!]
se[t] [*parameters*]
[.,.] t *linespec*
u[ndo]
[1,$]w[rite] [!] [*file*]
x[it]
[.,.]<
"
cd *path*
[.,.] d[elete] [*buffer*]
f[ile] [*file*]
[.,.+1] j[oin]
[.] ma[rk] *x*
n[ext]
[.] pu[t] [*buffer*]
[.] r[ead] [*file*]
[.,.]s[ubstitute] [*/pat/repl/*] [*options*]
una[bbreviate] *lhs*
[1,$]v */pat/command*
[1,$]wq [!] [*file*]
[.,.] y[ank] [*buffer*]
[.,.]>
[.,.]& [*options*]

a[bbreviate] *lhs rhs*

Defines an abbreviation *lhs* for *rhs*. Whenever *lhs* is typed as input followed by whitespace, it will be converted to *rhs*. If **abbreviate** is entered without parameters, a list of current abbreviations will be displayed. See also set remap.

cd *path*
Change the current working directory to *path*.

[.,.] **co[py]** [*dest*]

Copy the given range of lines after the line specified by *dest*. If the destination line specifier is 0 (the default), the lines will be inserted at the beginning of the file.

[.,.] **d[elete]** [*buffer*]

Delete the given range of lines. The deleted text is placed in the specified buffer if given. See also manipulators.

e[dit] [!] [*file*]

Open the given file. If the file is already open, its window will be brought to the front. If no file is specified, the current file will be refreshed from disk. If ! is specified no prompting will occur if the file has been modified.

f[ile] [*file*]

Display information about the current file. If a file is specified, the current window will be renamed to the given name.

[1,$] **g[lobal]** [!] / *pat* / *command*

Every line in the range is checked for *pat* in order from top to bottom. If *pat* is found on a line, then the line is made current and *command* is executed. If the **!** is specified *command* will be executed upon lines that do not contain the pattern. The entire global command is considered to be a single undo sequence. The character **?** may be used as a pattern delimiter instead of **/** for convenience, e.g. if your pattern contains slashes. The **v** command is the same as using **!**.

[.,.+1] **j[oin]**

Concatenates all the lines in the range into a single line. All white space between lines will be replaced by a single space.

[.] **ma[rk]** *x*

Set mark *x* to the given line. The **k** command is a synonym.

[.,.] **m[ove]** [*linespec*]
Move the given range of lines after the line specified by *linespec*. If *linespec* is 0 (the default), the text will be moved to the top of the file.

n[ext] *path*
Switch to the next window in the window stack.

[.,.] **p[rint]**
Print the given range of lines.

[.] **pu[t]** [*buffer*]

The contents of the specified buffer are pasted below the specified line. If the specified line is 0, the text is pasted at the beginning of the file. If no buffer is specified, the clipboard is used. Buffers are specified with a single letter, i.e. don't use the "*x* syntax.

q[uit] [!]
Quit Elwin immediately without saving modified files.

[.] **r[ead]** [*file*]

Read the contents of the given file and insert them after the given line. If the specified line is 0, the text is inserted at the beginning of the file. If no file is specified, the current file name is used.

se[t] [*parameters*]

If no parameters are given, a list of all option settings is displayed. Otherwise the parameters are used to set the value of the given options. There are three types of options: string, integer, and boolean. String and integer options are set using the syntax *option*=*value*. The value may be enclosed in double quotes for string options to accommodate spaces. Boolean options are set on simply by listing their name and are set off by prefixing them with **no**, e.g. **set autoindent** will turn on the auto indent feature while **set noautoindent** will turn it off. The available options are listed below.

[no]**autoindent**	Controls whether new lines are automatically indented with the same white space as the previous line.
directory	Specifies the directory in which temporary files will be created. This option has no effect on temporary files already in existence. This is a persistent setting and is saved with the profile information. The default is the value of the TEMP environment variable.
[no]**ignorecase**	Controls whether pattern matching is carried out case sensitive.
[no]**inputmode**	Controls whether the initial state of new windows is VI mode or normal input mode. This option is also persistent and saved to the profile. It is identical to VI Default.
[no]**list**	When set, tabs are displayed visibly and line ends are marked. This option is not yet implemented.
[no]**magic**	Controls whether search patterns are interpreted as regular expressions. This option is also persistent and saved to the profile.
[no]**number**	Causes lines to displayed numbered. This option is not yet implemented.
[no]**readonly**	Controls whether the file is read only. This option is not yet implemented.
[no]**remap**	Controls whether abbreviations are rescanned after expansion. Having this on can result in nasty looping.
report	This sets the minimum number of lines that will cause a message to be displayed if modified by a single command. It is not yet implemented.
scroll	This determines the number of lines that will be scrolled by Ctrl-D and Ctrl-U commands.
shell	Maybe someday it will be meaningful to shell out of Windows...
sidescroll	The number of columns scrolled by pressing the horizontal scroll buttons. The default is 8.
tabstop	The number of columns between tab stops. The default is 8. This parameter is automatically saved to the profile.
taglength	Not yet implemented.
wrapscan	Not yet implemented.

[.,.] **s[ubstitute]** [/ *pat* / *repl* /] [*options*]

For each line in the range Elwin replaces the first occurrence of the given pattern with the given replacement string. By default Elwin replaces only the first occurrence of the pattern on a line. An *option* of **g** causes Elwin to replace all occurrences of the pattern on each line. An *option* of **c** causes Elwin to prompt for confirmation before replacing each match. If no pattern and replacement is specified Elwin uses the most recently specified pair. The entire set of substitutions is considered to be a single undo sequence. See also Regular Expressions.

una[bbreviate] *lhs*
Removes any abbreviation defined for *lhs*.

u[ndo]
Causes the last edit sequence to be backed out.

[1,$] **w[rite]** [!] [*file*]

Causes the given range of lines to be written out to the given file. If no file is specified the current file is written. By default the file will not be written unless it has been modified. The **!** parameter causes the file to be written regardless of whether it has been modified.

[1,$] **wq** [!] [*file*]
Same as write then quit.

[.,.] **y[ank]** [*buffer*]

Causes the given range of lines to be copied to the given buffer. If no buffer is specified, the currently selected buffer is used. The buffer is designated by a single character, i.e. the "*x* syntax is not used.

[.,.] **<**

The given range of lines is shifted left.

[.,.] **>**

The given range of lines is shifted right.

" *comment*
    Commands beginning with a " are ignored.

[.,.] **&** [*options*]

Repeats the last substitute command applying any specified *options*.

x[it]

Immediately exits Elwin saving any modified files without prompting.

Configuration And Command Line

If the Elwin command line is empty it automatically starts in session mode. Elwin maintains a session for each directory in which it is initially opened. This makes it easy for you to have many projects going. The idea is that you will have an Elwin icon in Program Manager for each project you have. You set the properties for the icon so that the starting directory is the same as the project. Elwin does the rest and opens the files being edited the last time it was shut down in that directory. Putting "-d" on the command line defeats session mode.

Note that Elwin maintains session information in Elwin.ini which may become cluttered if you change or move project directories often. Feel free to clean it out occasionally if you wish. The format is simple.

If any files are specified on the command line, Elwin does not use session mode and does not save any session information when shut down.

The **File Open** dialog filters may be changed by adding to Elwin.ini in the [Elwin] section an entry for **FileOpenFilter**. The format for this entry consists of pairs of strings. In each pair, the first string will be displayed in the "List Files of Type:" listbox, and the second string will be used as the filter for the "File Name:" edit control. Every string, including the last, must be followed by a **|** character. For example, the default filter string entry would look as follows:

```
[Elwin]
FileOpenFilter=C++ Files (*.cpp;*.h)|*.cpp;*.h|C Files (*.c;*.h)|*.c;*.h|All files (*.*)|*.*|
```

If you primarily edit, say, C and Pascal files you may wish to use:

```
[Elwin]
FileOpenFilter=C Files (*.c;*.h)|*.c;*.h|Pascal Files (*.pas)|*.pas|All Files (*.*)|*.*|
```

The only other configuration detail of note is the use of disk caching. Elwin stores all text in temporary files in the directory used by Windows, typically specified by the TEMP environment variable. Disk cache software such as SmartDrive can dramatically improve Elwin performance. The use of a RamDrive utility for your TEMP directory is also highly recommended. See also EX command set directory.

Differences From VI

Several elements of the VI command set and behavior have been altered or removed for the Elwin implementation usually due to the inherent differences between the text-based interface of VI and the GUI interface of Elwin.

[[and **]]** have been changed to **[** and **]**.

The find and to set of commands, **f** and **t**, et al., behave slightly differently to accommodate a cursor that rests between characters rather than under them. The functional meaning has been preserved.

Buffer selection with " is more or less permanent whereas in VI it only affects the current manipulator. The unnamed buffer is the clipboard; choosing Cut or Paste from the Edit menu always uses it. The command "" also resets to it.

Shifting works with tab width not shift width.In non-linemode **p** and **P** do essentially the same thing; they differ only in where they leave the cursor. This is because of the logical difference between a cursor that is under a char and a caret between two chars. Alas, **xp** no longer twiddles chars, although this may be added as a special case in a future release...

J, the join command, always replaces white space with 1 space.

The commands **Ctrl-L**, **Ctrl-^**, **Ctrl-Z** have been ommitted.

No special keystrokes except accelerators and cursor movement have been implemented in insert mode.

Macro support has not been implemented. Some sort of macro support will probably be introduced in a future release.

Matching delimiter logic **%** is different if the caret is not sitting on a delimiter. VI searches ahead for first delimiter it finds. Elwin just fails.

Tips For The New User

Spend some time memorizing the **hjkl** movement keys. You will be surprised at how natural these become after you have them memorized, and you won't have to move to the arrow keys to get around. Older Unix hacks probably acquired their **hjkl** skills playing Rogue...

Similarly, the **Ctrl-F**, **Ctrl-B**, and **Ctrl-Y**, **Ctrl-E** keys can become very useful with a bit of practice. Note that all of these keystrokes allow you to plant the little finger of your left hand on the **Ctrl** key and then to use remaining fingers to do the scrolling without moving from the home keys. The orginal VT100 keyboard placed the **Ctrl** key directly left of the **A** key (where it belongs, frankly) which makes these combinations less of a reach. These commands work in either insert mode or VI mode.

Use the **.** redo command. It automatically turns any command sequence into a single-stroke macro. For the experienced user this can become the most used Elwin command. When using the change manipulator **c** it is a good idea to always press **Escape** after typing the changed text so the complete change will be available for the redo command.

Use **Ctrl-Tab** to cycle through non-iconized windows.

Quick block moves and copies can be accomplished using the **y** and **d** manipulators. If you can count the number of lines you need to copy or move, say 5 or fewer, you can type **5yy** or **5dd** to yank or delete the lines, move to where you want them, and type **p** to put them. If you have too many lines to easily count in a glance, use a mark and jump sequence. At one end of the block type **ma**; move to the other end of the block and type **y'a** or **d'a** to get the block; move to where you want it, and type **p** to put it.

Use the **''** command (two single-quotes) to jump back to the previous context. Any time you execute a move command for which there is no simple reciprocal command to return, the starting position is remembered as the "previous context". Typing the jump command **'** twice rather than specifying a mark letter jumps the caret to the previous context.

Technical Support

Technical support may be obtained from Little Wing by calling (501) 771-2408 Monday-Friday between the hours 9 AM and 5 PM Central Time or by sending e-mail to 71532,403 on CompuServe or 71532.403@compuserve.com on Internet.