```
VV                   VV    EEEEEEEEEEEEEE
 VV                 VV     EE
  VV               VV      EE
   VV             VV       EE
    VV           VV        EE
     VV         VV         EEEEEEEEEEEEEE
      VV       VV          EE
       VV     VV           EE
        VV   VV            EE
         VV VV             EE
          VVV              EE
           V               EEEEEEEEEEEEEE
```

VE v3.5 User Manual

for Mac OS X Tiger

December 2006

VE Brought to You By:

CampbellWare (www.campbellware.com)

1983, 1994, 2006

This document uses Courier, Times and Bitstream Vera Sans fonts.
If these fonts are not installed, this document may not display properly.

(This document prepared using NeoOffice 2.0)

Table of Contents

1.0 Introduction

1.1 About VE

VE (Visual Editor) is a small, fast, feature-rich text editor based on ncurses, with strong multi file editing support. VE started life in 1983 in Toronto Canada, as a text editor for the NABU 1600 small business computer. The NABU 1600 was an Intel 8086-based machine with 512Kb of RAM and a 10Mb hard drive, delivered with the QUNIX operating system.

The name "VE" was a good natured jab at the vi editor, which was delivered with the NABU 1600, but which could never be induced to run properly, necessitating the creation of VE. For several years in the mid 1980's, VE was the sole text editor for a small community of users on the NABU 1600. In 1994, the author acquired a PC (wow! VGA graphics!) and ported VE to MSDOS. Through the mid 1990's, VE was in use by a small community of MSDOS users. In 2005, the author ported VE to Linux (various distrubtions – see below) and **ncurses**, and distributed it to the Linux user community. In late 2006, the author ported VE to Mac OS X Tiger and distributed to the Mac community.

VE is provided free of charge, and is distributed under the GNU Public License v2. Use and enjoy.

1.2 VE Key Features

Key features of VE are:

- Mac OS X Terminal program
- Small, fast, tight implementation
- Compiled for PPC G5 (no yet available in Universal)
- Strong large file and multi file support
- Horizontal scrolling for files wider than display/window width
- Support for chars, words, paragraphs, lines, files
- Support for both line oriented and column oriented (rectangular) text areas
- Support for interactively defined macros

1.3 VE Has Been Verified On...

VE has been verified Mac OS X Tiger 10.4.8, running on a PowerMac G5 dual core 2.3 GHz. Note that VE v3.5g has also been extensively verified on many Linux distributions including Arch Linux 0.7.1, Arch Linux 0.7.2, SuSE Linux 9.3, SuSE Linux 9.0, various versions of Knoppix and various versions of DSL. VE v3.5g has also been back ported to MS-DOS and verified in that environment as well.

1.4 VE Website

Visit www.campbell-tx.net or www.campbellware.com for the latest information on and releases of VE.

1.5 VE Requirements

VE is an curses based text editor that be run in either a Mac OS X Terminal window or an Apple X11 xterm (or other X11 terminal emulator) window. Curses is a window and screen manipulation library for Terminal based programs. VE has no software requirements of the host computer except that curses be supported. Mac OS X Terminal fully supports curses.

VE is a full screen text editor. It requires that the Terminal or xterm that it is run in be sized for at least 80 columns and 20 rows. Anything of that size or larger (up to 512 columns wide) will support execution of VE. Anything less than this will cause VE to display an error message and exit.

VE binds several commands to keys on the number pad, such as the "/", "*", "+" and "-" keys. These bindings are only available if VE is run from within an xterm – they are not available within a Terminal windows. These bindings will work most efficiently if the TERM environment var is set to **TERM=xterm**. These key bindings are in essence keyboard shortcuts – the associated commands are all available via their "ESC-command_letter" form, making them fully available from Terminal windows.

VE makes use of color as part of its line block and column block selection process, and in its Open File and Goto File dialogs. VE also supports user selection of the screen foreground and background colors. These color related capabilities will generally require the TERM environment variable to be set to either **TERM=xterm** or **TERM=xterm-color** in order for proper color display to occur.

1.6 Installing VE

VE is a self contained executable. Installation is extremely simple. Unpack the release tarball (which you have done successfully if you are reading this) and cd into the ve-3.5x-release directory that is created ("x" is replaced by the subversion letter of the releae you have downloaded – for example, 3.5g). Simply copy the executable file "ve" to any convenient directory on your path, and you are ready to use VE.

1.7 Mac OS X VE Keyboard Setup

All VE commands are available via ESC-x, where "x" is any key. However, to be most easily used, VE automatically binds it's most commonly used commands to the function keys F1 – F12. Unfortunately, Mac OS X uses the F9-F12 keys for Dashboard and Expose. As a result, these keys are intercepted by Mac OS X and do not make it into VE itself. If you wish to use the F9-F12 keys for their VE functions, the following changes will accomplish this result.

Map the Dashboard and Expose keys to CMD-F9 through CMD-F12. This can be accomplished very easily. Go to the system Preference panel for Dashboard and Expose, and make the changes directly in the lower area of the presented preference panel.

Two other changes can enhance your use of VE under Mac OS X's Terminal program. Mac OS X intercepts the Pg Up and Pg Dn keys and uses them to scroll the Terminal buffer back and forth. With two simple changes, these keys can be passed through to VE where they accomplish the expected result of paging forward and backward through the currently presented file.

To make the changes in question, open a Terminal windows. Right click (or CTL-click) anywhere inside the Terminal window and select Window Settings from the presented context menu. In the resulting Terminal Inspector window, select Keyboard from the drop down menu at the top of the window. Scroll down to the definition for Pg Up and Pg Dn. Select Pg Dn and press the Edit button below the key list. Type in "\033[8" (do not type the quotes). Now select the Pg Up key, edit it, and enter "\033[7" (again, do not type the quotes). Finally, click the Use Settings as Defaults button. This completes the changes, which will be applied to all further Terminal windows.

1.8 Starting VE

Start VE with one of five command forms:

1. ve
   This form starts VE and opens an empty new unnamed file.

2. ve -b
   This form starts VE in directory browsing mode. VE presents its full screen directory browser and allows interactive selection of the file to edit.

3. ve filename
   This form starts VE and opens the named file for editing

3. ve filename_1 filename_2 ... filename_n
   This form starts VE and opens all files provided on the command line for editing. Wildcards may also be used, as allowed by the shell you are using. All identified files will be opened for editing, up to a maximum of 256 files. You may use the "+", "=", keypad "+" key or the Edit "+" command to cycle through the open files. The Goto File command (ESC-g or keypad "-") allows you to select a specific file at will. See the documentation for the Goto File command, below, for more details.

4. ve filename -l line_number
   This form starts VE, opens the supplied filename and moves to the indicated line number in that file. This is extremely useful for moving directly to the line number indicated in a compiler error message, or going directly to an area of a file that you are actively working at.

5. ve -h (or ve –help)
   This form starts VE and displays the help panel. When the user exits the help panel, it opens an empty new unnamed file for editing.

At startup, VE will look for a file named ~/.ve.opt. If present, VE will load this file and set its options from it's contents. This file may be created from within VE by setting program options as desired, and then using the Options menu's Save subcommand to save the current option set. VE saves the option set to the file ~/.ve.opt. This functionality is only operative if the HOME environment variable is set.

2.0 VE Quick Start Guide

The following is a quick overview of the most often used VE commands, allowing you to get quickly underway with VE.

Text Entry

Simply type the text to be entered. VE will place that text into the file. Unlike editors such as vi, VE defaults to being in text entry mode, not command mode.

Update File (ESC-U, F11)

The Update command writes the current file out to disk, and then resumes editing of it. It is equivalent to the File, Save sequence of most GUI editors. If it is a new file, VE will prompt for a file name before writing it to disk.

Find Text (ESC-f, F5), Replace Text (ESC-r, F6)

The Find command allows you to quickly locate text in a file and jump to it. The Replace command allows you to replace occurrences of specified text with different text.

Repeat Last Command (ESC-a, F1)

The Again command repeats many previous commands. This is very useful with the Find and Replace commands, to sequentially find/replace the next occurrences. Alternately, to replace all occurrences, you can enter the '*' repetition count before the replace command (ESC * ALT-r, or ESC * F6)

Undo Changes (ESC-u, CTL-z)

The Undo command will undo most changes made to a file. Note the VE does not support a Redo command.

Quit VE (ESC-q, F12)

The Quit command will present a menu allowing you to save your file and then quit VE, or discard your file and quit VE unconditionally (Save and Discard selections respectively).

3.0 VE User Interface

3.1 Entering Text

VE starts and run in text entry mode. VE is normally in INSERT mode (characters typed are inserted into the file) or in OVERWRITE mode (characters typed overwrite existing characters in the file). VE will scroll the display window to the right if the line being typed exceeds the width of the physical display or xterm window, up to the maximum supported line size (384 characters in the current version of VE).

VE displays it's current mode (Insert, Overwrite or Command) in the upper right corner of the display.

3.2 Entering VE Commands

VE commands are entered by typing ESC, an optional repetition count, and a command letter. Note that **if xterm Xresource "xterm*metaSendsEscape" is set to TRUE, the ALT key can be used as an alternate to the ESC key** if so desired. Both work equally well, although the ALT key must be pressed at the same time as the command key, whereas the ESC key can be pressed, released, and then followed by the command key. The full set of available commands is listed in the VE Help Panel, available via ESC-h or ALT-h, and in Appendix A of this manual. VE commands are case sensitive. For example, ESC-a is the "again" command, while ESC-A is the "about VE" command.

Typing ESC causes the next key typed to be interpreted as a command letter, and executed as such. Per the above, the full set of available commands is listed on the VE Help Panel and in Appendix A of this manual.

Many VE commands have additional keyboard "shortcuts", such as function keys or control codes, but **every** VE command can be executed via "ESC-command_letter", which means that VE can be used with any computer environment, irrespective of local key mappings.

VE always returns to text entry mode after executing a command. This makes VE safer for the uninitiated than a text editor like vi, where typing characters without first entering INSERT or REPLACE mode can cause catastrophic results, as each typed character is interpreted as a command and executed.

3.3 Function Keys (F1-F12)

The function keys F1-F12 are internally bound to a set of frequently used VE commands. See Appendix A for a listing of the function key bindings. These commands may be entered by simply pressing the associated function key. For example, F1 executes the “again” command. In the current release of VE, it is not possible for a user to customize the command bindings for the Function Keys. In general, environment variable TERM should be set to “xterm” or “xterm-color” for Function key bindings to work as efficiently as possible. See section 1.7 for important notes about enabling VE to receive the F9-F12 keys under Mac OS X.

3.4 Exiting From Menus, Prompts, Etc.

The [ENTER] key will terminate any menu that VE is presenting, and will complete any input that VE is prompted for. This is also true for exiting from command mode. If you have entered ESC as the start of a command, it is possible to exit without running any command by simply pressing ENTER.

3.5 Command Repetition Counts

As mentioned above, VE's general command syntax is:

ESC (optional repetition count) command_letter

Any VE command can be preceded by a repetition count. The command will be executed as many times as indicated. The special count character '*' means to repeat infinitely – this is useful with the Replace command, where it in essence means replace all occurrences, but should be used with EXTREME caution elsewhere. For example, executing a macro infinitely can have unexpected and unfortunate results depending on the content of the macro.

To enter a repetition count prior to a command, press ESC to begin command entry, and then the count itself. When the desired count has been entered (the count is echo'd on the status line and may be edited via the Backspace/Rubout key), simply enter the intended command key, and the command will be sequentially executed the number of times indicated by the repetition count that has been entered. If the command has a extra keyboard “shortcut”, such as a Function key or a CTL-code, this may be used instead of the command key if so desired.

3.6 Repeating Commands Sequentially

The VE Again command (ESC-a, F1) can be used to repeat most commands that change the file (such as delete char, delete line, replace text, line/column cut, copy, paste, etc.) and the Find Text command. The Again command repeats the command using any repetition count that was initially entered with it. The Again command can be used repeatedly to achieve as many sequential executions of a command as is desired.

3.7 Undoing Changes

The VE Undo command (ESC-u or CTL-z) provides a complete Undo capability, allowing users to undo any number of changes to a file, in the order in which they were made. In the current release of VE, the undo queue is arbitrarily limited to 128 items. After this limit is reached, as new undo actions are added, the oldest ones are discarded. Users are not able to change this limit.

Like all other VE commands, the Undo command may be used with a preceding repetition count. For example, "ESC 5 u" will undo the last five changes made. "ESC * u" will undo all changes presently queued for undo action, potentially returning the file to the state it was at when opened.

The VE Undo command fully supports the undoing of macro executions. VE supports macros, which are interactively defined collections of VE commands that accomplish some useful purpose (the creation and use of VE macros is fully documented later in this user manual). After executing a macro, a single Undo command (ESC-u) will undo all changes that have been made by that macro, irrespective of the total number of such changes.

VE's undo capability is specific to the file being edited at the time changes are made. If a user switches from one file to another (either by opening an additional file, or switching to another already open file), VE will flush the current queue of pending undo actions in order to preserve file continuity.

VE does not support a Redo capability. Once a change has been undone, it cannot be redone, except by manually entering the change again.

3.8 Mouse Support

VE provides limited mouse support when run in an xterm under Apple X11. Mouse support is not available in a Terminal window. Basic mouse support is provided during file editing, where the mouse may be used to position the cursor at a particular location on the display page. Simply move the mouse pointer to the desired cursor location and click the left mouse button. VE will move the cursor to the indicated position. This is the only mouse functionality supported during file editing. Mouse support is also available in VE's Open File Dialog and its Goto File Dialog, where the mouse may be used to select a file entry via single or double mouse clicks. See later documentation in this manual for these dialogs for more information.

VE defaults to providing the above mouse functionality. However, VE mouse support may be enabled or disabled on the fly, via the [Options, Rodent] (ESC-o, r) menu path. This allows mouse support to be enabled/disabled either temporarily, or as VE's default setting. The selected setting will become VE's default if VE's options are saved after changing the mouse support setting. See the Chapter on configuring VE for full information on this topic.

A major reason for temporarily disabling VE's mouse support is to enable copying and pasting of information into an open file via X-windows cut and paste operations. These operations require that X, not VE, see and interpret mouse events.

As an alternate to disabling VE's mouse support temporarily, it is usually possible (in an xterm) to simply press the SHIFT key while performing any desired X-based mouse copy/paste operation. Unless the treatment of shifted mouse clicks has been modified (for example, via window manager settings), shifted mouse clicks are not passed through to applications such as VE via the ncurses mouse interface. As a result, X will receive the mouse events, not VE. Using SHIFT plus mouse operations therefore makes X-based copy and paste to/from VE display pages possible even while VE's mouse support is enabled.

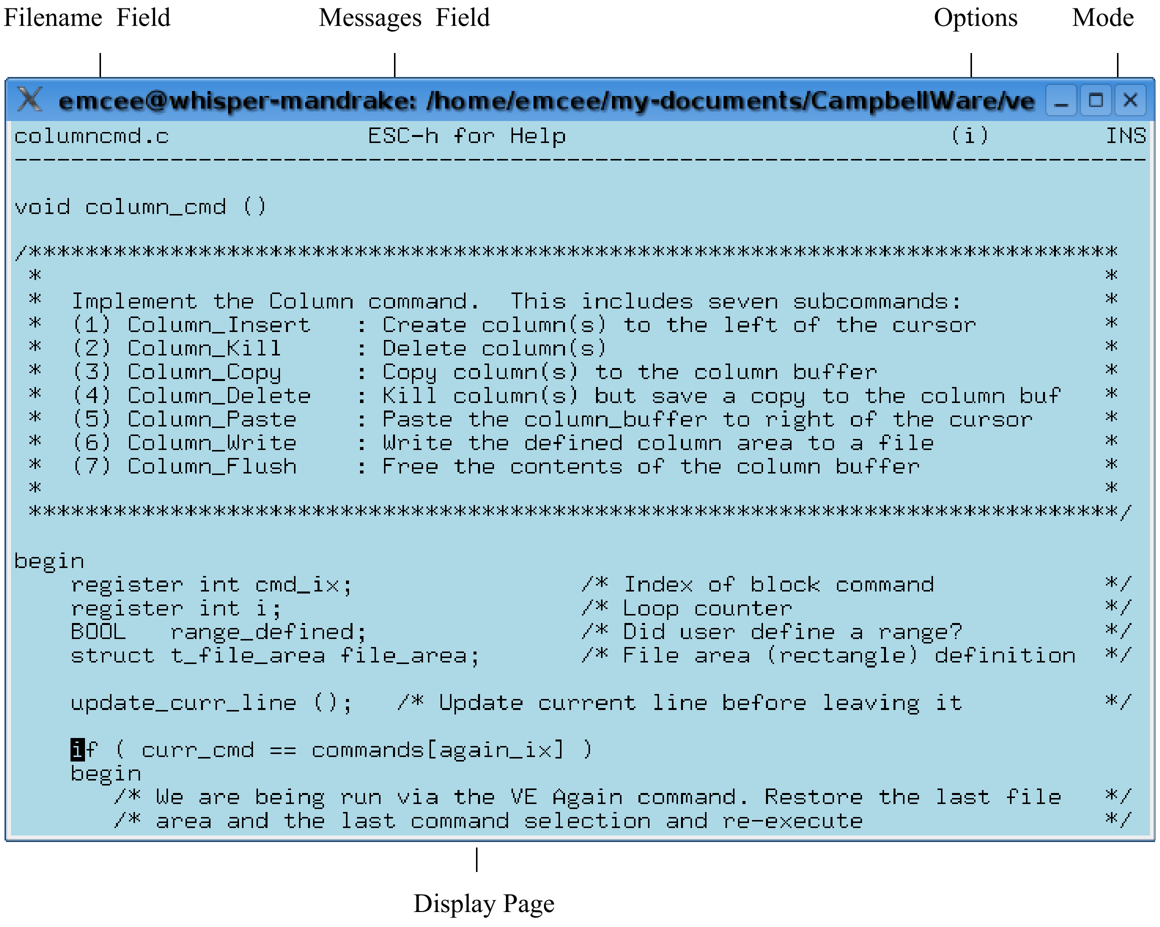
VE leaves this choice to the user. Both mechanisms (enable/disable VE mouse support, using SHIFT plus mouse opeations) are supported.

3.9 Getting Help

This document is the prime source of information about the VE editor. However, the online Help Panel, available via ESC-h, provides an overview of the available commands, although not detailed documentation on any of them. This information is presented as a full screen panel. At the bottom of the screen VE presents the usual full screen navigation guide, with ENTER as the only option. Pressing ENTER will return to editing the current file.

4.0 VE Screen Layout and Interaction Model

4.1 VE Screen Shot


Filename Field
Messages Field
Options
Mode
emcee@whisper-mandrake: /home/emcee/my-documents/CampbellWare/ve
Display Page

```
columncmd.c              ESC-h for Help                          (i)        INS
--------------------------------------------------------------------------------

void column_cmd ()

/******************************************************************************
 *                                                                            *
 *   Implement the Column command.  This includes seven subcommands:          *
 *   (1) Column_Insert   : Create column(s) to the left of the cursor         *
 *   (2) Column_Kill     : Delete column(s)                                   *
 *   (3) Column_Copy     : Copy column(s) to the column buffer                *
 *   (4) Column_Delete   : Kill column(s) but save a copy to the column buf   *
 *   (5) Column_Paste    : Paste the column_buffer to right of the cursor     *
 *   (6) Column_Write    : Write the defined column area to a file            *
 *   (7) Column_Flush    : Free the contents of the column buffer             *
 *                                                                            *
 ******************************************************************************/

begin
    register int cmd_ix;                /* Index of block command            */
    register int i;                     /* Loop counter                      */
    BOOL    range_defined;              /* Did user define a range?          */
    struct t_file_area file_area;       /* File area (rectangle) definition  */

    update_curr_line ();    /* Update current line before leaving it         */

    if ( curr_cmd == commands[again_ix] )
    begin
       /* We are being run via the VE Again command. Restore the last file   */
       /* area and the last command selection and re-execute                 */
```

(Screen shot taken from Linux VE using KDE, Plastik theme, and Default VE colors with Lightblue xterm)

4.2 VE Screen Layout Notes

The VE screen layout consists of the following logical areas:

- Display Page: this is the majority of the screen, and is the area where the file data is displayed. This page may be scrolled up, down, left and right to display all of the file content.
- Status Line: this is the top line of the screen, and is the area where VE presents information about the current file and VE's current mode.
- Status Line Filename Field: this is at the leftmost end of the status line, and displays the name of the currently editing file. If the filename is longer than a certain length, it may be truncated in this field.
- Status Line Messages Field. This is in the middle of the status line, and is the area where VE presents any messages that it wishes the user to see. The majority of VE commands use this area to present status information reflecting what they have done. For example, the Replace command reports the number of substitutions made in this area.
- Status Line Options Field. This is to the right of the Messages field. VE uses this area to display indicators reflecting the status of editor options that the user may have enabled. The available indicators are:
- “i” - indicates that auto indent is active
- “w” - indicates that text wrapping is active
- “m” - indicates that macro recording is active
- “<-” - indicates that the display page is scrolled to the right
- Status Line Mode Field. On the right hand edge of the status line, this is where VE presents its current mode, which is one of Insert (INS), Overwrite (OVR) or Command (CMD)

4.3 Status Line Command Interaction Model

All command dialogs with VE occur on the status line as well. VE will present menus and most command feedback from any menu selections made in this area. Where more detailed feedback from a command is needed, VE will temporarily deepen the status line area to three lines deep, and utilize the extra lines as needed. When the interaction is complete, VE restores the status line to its original single line depth.

Since the status line is only one line “tall”, and VE may need to present nested menus (if for example a selected command has a menu, and then the selected menu item itself has a menu), VE treats the status line like a push down stack. When a VE command needs to present a menu, it conceptually pushes the current status line onto an internal status line stack, uses the status line as it wishes, and then pops the previous status line off the internal status line stack when it is done, thus restoring the previous status line.

Since ENTER will terminate any menu that VE is presenting, to move “up” through stacked menus (i.e. To get back to the previous status line menu) simply press ENTER repeatedly until you return to the menu of interest, or back to editing the file.

4.4 Status Line Response Line Editing

Many VE commands present information on the status line and request user input. A common example of this is the Find command, which prompts the user for a string to find. Another common example is the [Jump, Line] command, which prompts for a line number to jump to. In all cases where a user is prompted to enter text, full line editing support is provided while that text is being entered. The following line edit keys are supported:

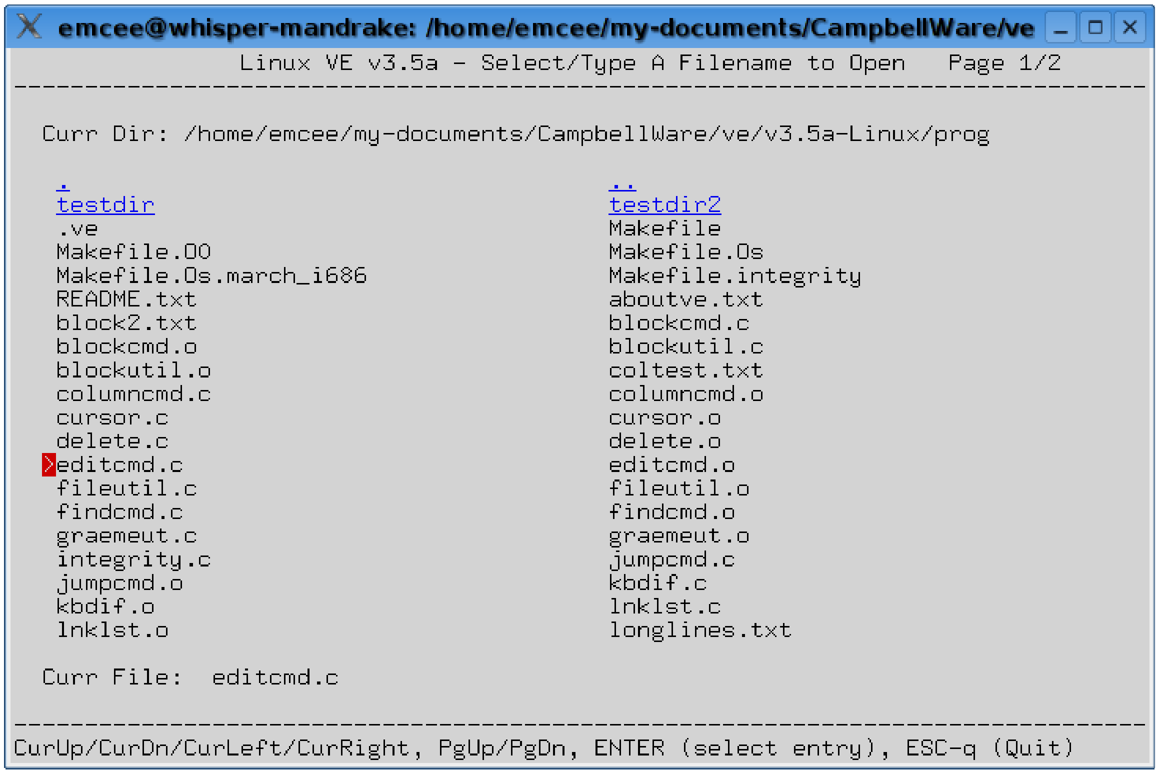
- Rubout/Backspace – erases the most recently entered character
- Delete – deletes the character under the cursor
- Home – moves to the start of entered response
- End – moves to the end of the entered response
- Cursor Left – moves the cursor left one position
- Cursor Right – moves the cursor right one position
- F10 or ESC-k – deletes all response characters to the right of the cursor
- ESC-d – deletes all response characters to the left of the cursor

F10 (or ESC-k) deserves special mention with respect to the VE Find command. A common use of the F10 or ESC-k capability occurs with this command, which remembers the last find string that was entered, and preloads the presented find string with the last one used. To enter a new string instead of reusing the previous string, the fastest approach is to enter F10 or ESC-k, which clears the previous string, and then enter the new string.

4.5 VE Open File Dialog

A number of VE commands allow users to select or specify filenames for reading, writing or opening. All such commands use a common VE Open File Dialog, which includes an interactive file system browser. This dialog allows users to select a file from the current directory, or browse the file system to a new directory and select a file from there or alternately, directly type a partial or complete filename.

A screen shot of the Open File Dialog appears below:



emcee@whisper-mandrake: /home/emcee/my-documents/CampbellWare/ve

```
                 Linux VE v3.5a - Select/Type A Filename to Open    Page 1/2
--------------------------------------------------------------------------------

  Curr Dir: /home/emcee/my-documents/CampbellWare/ve/v3.5a-Linux/prog

   .                                      ..
   testdir                                testdir2
   .ve                                    Makefile
   Makefile.O0                            Makefile.Os
   Makefile.Os.march_i686                 Makefile.integrity
   README.txt                             aboutve.txt
   block2.txt                             blockcmd.c
   blockcmd.o                             blockutil.c
   blockutil.o                            coltest.txt
   columncmd.c                            columncmd.o
   cursor.c                               cursor.o
   delete.c                               delete.o
  >editcmd.c                              editcmd.o
   fileutil.c                             fileutil.o
   findcmd.c                              findcmd.o
   graemeut.c                             graemeut.o
   integrity.c                            jumpcmd.c
   jumpcmd.o                              kbdif.c
   kbdif.o                                lnklst.c
   lnklst.o                               longlines.txt

  Curr File:  editcmd.c

--------------------------------------------------------------------------------
CurUp/CurDn/CurLeft/CurRight, PgUp/PgDn, ENTER (select entry), ESC-q (Quit)
```

The Open File Dialog presents a full page listing of the contents of the directory from which VE was started. Cursor left, right, up and down keys may be used to move the file selection carat to the file of interest, which may then be selected by pressing ENTER. If the directory listing exceeds one display page, the PgUp and PgDn keys may be used to move to the next/previous page of the listing as desired. As an alternate to using the PgUp or PgDn keys, the cursor up and cursor down keys may be used. When they reach the top or bottom of the displayed files, if there is another page, they will wrap to that page. Note that the cursor keys will wrap from the left to the right, and visa versa, and will wrap from the top to the bottom and visa versa. In similar fashion, the PgUp and PgDn keys will wrap from the first page to the last, and visa versa.

The Open File Dialog provides mouse support. Single clicking the mouse on a directory entry will move the selection carat to that entry. Double clicking on a directory entry will select the file that is double clicked and complete the dialog, as if the selection carat had been moved to the file of interest and ENTER had been pressed.

If the file of interest is not in the directory that is display, the Open File Dialog may be used to browse the file system until the directory of interest is reached. To do this, simply select any directory entry and press ENTER (or double click the directory entry with the mouse). This will open the selected directory and present its contents. This process can be repeated until the directory of interest is reached.

As an alternate to interactively selecting a directory and file of interest, a partial or full path name may simply be directly typed. It will be echo'd in the Curr File field, complete with full line editing support. Partial filenames are assumed be relative to the currently displayed directory, while full pathnames (those starting with "/") are assumed to be absolute paths. Note that relative pathnames of the form "../../directory_name/filename" are fully supported.

Note that if the Open File Dialog is used to select a file that VE already has open, VE will simply switch to that file, rather than opening a second copy of it.

Users may cancel out of the Open File Dialog at any time without selecting a directory and file by pressing ESC-q. F12 achieves the same result.

Note that the cursor movement and PgUp, PgDn functions may also be selected by the same ESC-command_letter and keyboard shortcuts that apply to these keys while editing files in VE. Specifically, the following alternate forms are available while in the Open File Dialog:

● Cursor Up -	CTL-u, ESC-^
● Cursor Down -	CTL-d, ESC-$
● Cursor Left -	CTL-l, ESC-,
● Cursor Right -	CTL-r, ESC-.
● Page Up -	CTL-p, ESC-B
● Page Down -	CTL-n, ESC-F

5.0 A Few Words About...

5.1 A Few Words About Tab Handling

With its default settings, VE expands Tabs into the equivalent number of spaces as a file is read in. Tabs may be optionally re-inserted as a file is written back out. In general VE views tabs simply as a way of compressing a file's size on disk. Tabs are expanded when a file is read in, and may optionally be used to compress the file's size as the file is written back out.

This behavior can be changed via the [Options, Tabs] menu path (ESC-o, t), which allows both input and output tab translation to be enabled or disabled. If Input Tab Translation is disabled, VE will represent TABs using the character '?'. Similarly, for Tab characters entered into a file as text, VE represent those characters as “?”. When Input Tab Translation is enabled, VE will expand Tabs into the appropriate number of spaces as a file is read in, and will also convert any TABs typed as text to the number of spaces needed to achieve the next tab stop.

When expanding tabs on file input, VE uses a default tab stop of 4. If this tab stop causes the visual alignment of text in a file to be incorrect, determine the correct tab stop setting for that file. Then change VE's tabstop setting via the [Options, Tabs] menu path (ESC-o, t), discard the file and re-open. Alignment should now be correct. Note that when you write the file back out, it will not contain tabs unless you enable use of tabs on output via the [Options, Tabs] menu path (ESC-o, t), which is disabled in the default options.

Finally, an application note. The Linux “make” utility REQUIRES that it's Makefile input file contains Tabs. When editing make files, you MUST disable Input Tab Translation. If this is not done, the saved Makefile will not be acceptable to the “make” utility.

5.2 A Few Words About Text File Formats

VE lives in a mixed Mac OS X, MS Windows and Linux world. Many machines that VE will run on will contain two operating systems and may mount disks from another operating system via file sharing. A well known difference between the MS Windows and the Mac OS X and Linux OS' is the format in which they store text files. Mac OS X/Unix/Linux uses a single Line Feed character (0x0A) as the separator between text lines in text files. DOS/Windows uses the combination of carriage return (0x0D) and Line Feed (0x0A). As a result, a “pure” Mac OS X/Unix/Linux editor that opens a file sourced by DOS/Windows will generally show the “extra” CR character as its ASCII equivalent ^M. Opening such a file with Midnight Commander's internal editor is an excellent example of this.

To ensure that both MS Windows and Mac OS X/Unix/Linux files can be read and written properly, VE supports both formats, determining on the fly as the file is opened which text format it is using. VE then uses the same format when writing that file out. For new files created with VE, the default format is the Mac OS X/Unix/Linux one, but this can be overridden, via the [Options, Fileformat] menu path (ESC-o, f). This command allows you to set the text file output format to either DOS or Linux. Hence, VE running from Mac OS X/Unix/Linux can be used to create new text files that will be stored on a Mac OS X/Unix/Linux-accessible DOS/Windows partition of the same machine, in a manner that will allow DOS/Windows text editors to open them normally.

The text file format of an open file may be checked at any time. When VE opens a file, it displays information about the file in its status line. This includes the number of characters and lines, and also directly shows the file format that VE has determined that the file is using. At any time during editing of a file, you may check the file format by issuing the Status command (ESC-s) or the Goto File command (ESC-g), both of which display full statistics about the current editing session, including the file format of the currently edited file.

6.0 Entering and Formatting Text

6.1 *Entering Text*

To enter text, simply begin typing. VE defaults to being in text insertion mode, either in INSERT mode (new text is added to the file) or in OVERWRITE mode (new text overwrites existing text). VE will automatically scroll the display to the right if the text being entered exceeds the width of the physical display or xterm. The current version of VE supports a maximum line size of 384 characters. This is an arbitrarily chosen number, and may easily be increased by changing a constant (MAX_LINE_SIZE) in the source module COMMON.DEF and rebuilding. See Appendix B for instructions on building VE from the supplied source code.

To change between INSERT and OVERWRITE mode, enter CTL-O or ESC-I, or press the INSERT key on the keyboard if your keyboard has such a key (Apple's standard keyboard does not). Note that the “I” in ESC-I is an upper case letter. VE displays it's INSERT/OVERWRITE mode in the upper right corner of the screen. You will see this toggle as you repeatedly press the CTL-O/ESC-I/INSERT commands.

6.2 *Formatting Text*

VE supports both on-the-fly and after-the-fact text wrapping, which can be very useful when composing text documents such as letters, program documentation etc.

6.2.1 On-The-Fly Text Wrapping

VE supports on-the-fly text wrapping, which is text wrapping that occurs as text is typed, without the need of manually pressing ENTER. Text can be entered and/or deleted continuously, and VE will format the lines and insert line breaks as needed.

To enable on-the-fly text wrapping, go to the Options menu, Wrap submenu (ESC-o, w) and select the “Enable” menu item. A “w” indicator will be displayed in the options area of the status line to indicate that on-the-fly text wrapping is active. Prior to enabling wrapping, the “Left” and “Right” menu items may be used to adjust the left and right margins within which VE will wrap text. Alternately, VE's defaults of left margin at -1 and right margin at column 72 may be used, in which case no change is needed. A left margin setting of “-1” instructs VE to determine the correct left margin to use automatically, based on the current level of indenting of the paragraph being typed in.

When On-The-Fly Text Wrapping is first enabled, VE attempts to wrap the current paragraph as starting point. Text may now be entered continuously, and VE will wrap that text as it is entered. Text entry can continue in this way until all desired text has been entered. At this point text wrapping may optionally be disabled via the [Options, Wrap, Disable] menu path (ESC-o, w, d).

While text wrapping is on, it is possible to move freely through the file and type at any place. VE will begin wrapping any line of text as soon as (a) at least one character is entered on that line, and (b) the line length exceeds the selected right margin.

6.2.2 After-The-Fact Text Wrapping

VE also supports after-the-fact text wrapping. After-the-fact text wrapping refers to entering one or more lines of text without on-the-fly text wrapping active, and then wrapping that text one or more paragraphs at a time after text entry is complete.

To do this, simply position the cursor in any paragraph to be wrapped, and enter the VE Wrap command (ESC-w). VE will reformat the paragraph to make it fit between the selected left and right margins, and will optionally right justify each line, giving the text a clean looking uniform right edge. This may be repeated on as many paragraphs as desired. Right justification is on by default, but can be disabled via the [Options, Wrap, Justify] menu path (ESC-o, w, j). This selection toggles the state of right justification between “on” and “off”.

After-The-Fact Text Wrapping supports a special feature for unique first line indentation. If the text wrapper notices that all lines of the paragraph to be wrapped have an identical indentation level except the first line, it will assume that this is intentional, and preserve the indentation level of the first line of the paragraph when wrapping. This allows for the wrapping of such things as bulleted lists and letter style paragraphs, where the first line has extra indentation vs. the remaining lines of the paragraph. As an example, consider a paragraph of the form:

```
o This is the first line of a bulleted list
  paragraph. The following
  lines will have the same
  indentation level. Only the first line is different.
```

VE's text wrapper will recognize this construct, and wrap it with the first line indented at the selected left margin, while the rest are indented relative to this, as typed.

The result of wrapping the above paragraph would be similar to the following:

```
o This is the first line of a bulleted list paragraph. The following lines
  will have the  same indentation level. Only  the first line is different.
```

Multiple paragraphs may be wrapped sequentially by repeating the Wrap command as many times as desired, either by directly re-entering the command, or using the VE “Again” command (ESC-a) or F1.

VE also supports wrapping multiple paragraphs (or even entire files) at once via the Block Wrap command. See the documentation for the Block commands, later in this document.

6.3 Breaking and Joining Lines

Any given line can be broken into two lines by placing the cursor at the point where the line break is desired, and pressing the ENTER key.

Two lines may be joined together by moving to the end of the upper of the two lines (place the cursor on the line of interest and press the END key) and pressing the DELETE key. Alternately, two lines may be joined by pressing Backspace/Rubout while in the first column of a line. That line will be joined with the line above it.

7.0 Moving Around In A File

7.1 Cursor Left, Right, Up and Down

The keyboard cursor keys do the expected things, moving the cursor around the file one row or column at a time. Some special behaviors are noted below.

If Cursor Left is pressed while on the first character of a line, VE will back up to the last character of the preceding line. The inverse is not true for Cursor Right at the end of a line, as VE allows you to cursor out beyond the end of a line and start typing. When this occurs, VE fills the intervening columns with spaces.

If Cursor Right is pressed when the cursor is at the right hand edge of the screen, VE will scroll the display window right by one column. Similarly, if Cursor Left is pressed when the cursor is at the left hand edge of the screen, and the display window has previously been scrolled right, VE will scroll the display window left by one column.

The cursor left and right, up and down commands are also available via CTL-l, CTL-r, CTL-u and CTL-d (also available via ESC-, and ESC-., ESC-^ and ESC-$, respectively).

7.2 Move to Start of Line, End of Line

The HOME and END keyboard keys move the cursor the start and end of the line the cursor is on, respectively. The HOME key will scroll the display window to the right in order to show the end of the line, if that end lies beyond the present right hand edge of the window. Similarly, the HOME key will scroll the display window left as needed in order to display the start of the line. These commands are also available as ESC-< and ESC->.

7.3 Scroll Up One Page, Down One Page

The Page Up and Page Down keys move the VE display page up one page or down one page respectively. If the top of file is encountered, the cursor is left at the very top of file; if the bottom of file is encountered, the cursor is left at the very bottom. These commands are also available as ESC-B and ESC-F respectively, and again as CTL-p (previous page) and CTL-n (next page). See Section 1.7 for information on how to make the Pg Up and Pg Dn keys available to VE under Mac OS X.

7.4 Scroll Half Page Up, Half Page Down

The VE command SHIFT+PageUp (hold down SHIFT and Page Up at the same time) will move the display page up by half a page. Correspondingly, the VE command SHIFT+Page Down (hold down SHIFT and Page Down at the same time) will move the display page down by half a page. These command are also available as ESC-- and ESC-+. See Section 1.7 for information on how to make the Pg Up and Pg Dn keys available to VE under Mac OS X.

7.5 Scroll Right or Scroll Left

The VE command SHIFT+Cursor Right (hold down SHIFT and Cursor Right at the same time) will scroll the display page a half screen to the right. Correspondingly, the command SHIFT+Cursor Left (hold down SHIFT and Cursor Left at the same time) will scroll the display page a half screen to the left (if it has been previously shifted to the right, either by text entry, Cursor Right, or Scroll Half Screen Right). The Scroll Left and Scroll Right commands are also available as ESC-(and ESC-) respectively.

The scroll indicator "<-" will appear in the status line Options area when the display page is scrolled right. It disappears when the display page is scrolled back to column zero.

7.6 Move to Next Word, Previous Word

The ESC-CursorRight (ESC and then the cursor right key) and ESC-CursorLeft (ESC and then the cursor left key) commands will move forward or backward by one word. These commands are also available via ESC-], ESC-[respectively.

7.7 Move to Next Paragraph, Previous Paragraph

The ESC-CursorDown (ESC and then the cursor down key) and ESC-CursorUp (ESC and then the cursor up key) commands move the cursor to the bottom of the current paragraph or the top of the current paragraph. Repeated use of these commands will move up and down the file a paragraph at a time. These commands are also available via ESC-} and ESC-{ respectively.

7.8 Move to Top, Middle, Bottom of Display Page

The commands ESC-H, ESC-M and ESC-L (note that these are upper case command letters) will move the cursor to the top, middle and bottom of the display page, or to the end of the displayed file, whichever occurs first on the currently displayed screen. Users familiar with the vi text editor will recognize these command letters (H, M and L) as being the same command letters used for the equivalent purpose in that venerable editor.

As an additional convenience, VE binds these three commands to the control sequences CTL-t, CTL-g, and CTL-b. The "t" and "b" are references to "top" and "bottom"; the "g" is an acknowledgment that the letters "t", "g", and "b" lie in an essentially vertical column on a standard QWERTY keyboard, with the letter "g" being in the middle between "t" and "b".

7.9 Go To Display Page Edge

VE was originally written in the mid 1980's, before the era of the ubiquitous IMB PC architecture. Most keyboards of the day were not of the 102/110 key “extended PC keyboard” type. Instead, they had the basic QWERTY keys and cursor motion arrow keys. Some had a HOME key, some did not. The computer on which VE was born, a NABU 1600 Qunix-based system, came with an ESPRIT3 serial terminal (no graphics!) with no Page Up or Page Down keys, but four cursor movement arrow keys clustered like the four points of a diamond, with a HOME key in the middle. This is nearly identical to the configuration of the keypad on most standard PC keyboards today, with the four cursor motion keys being located on the 2, 4, 6 and 8 keys, with the 5 key in the middle.

That key configuration gave rise to the Extended Cursor Motion command. When a cursor key is used, it sets a cursor direction. The Cursor Left key sets the direction to left, the Cursor Right key sets it to right and so on. When the keypad “5” key is pressed, VE performs an extended cursor movement in the currently set cursor motion direction. In the Qunix original, the four directions were used to achieve the equivalents of Page Up, Page Down, Home and End. The Linux and Mac OS X ports of VE uses this concept instead to move to a screen edge, since today's keyboards provide dedicated keys for the Page Up, Page Down, Home and End functions. Most Mac keyboards do not show the arrow symbols on the 2, 4, 5, 6 and 8 keys, but this function is implemented nonetheless. The original Qunix functionality is still available as the Extended Motion command (see below).

For the Mac OS X port of VE, the “Edge” command works per the following (note that this function is available only in an xterm window under Apple X11. It is not available in a Terminal window):

- Cursor Left and then “5” moves the cursor to the left edge of the display page. This may not be the first column of the file if the display page has been scrolled right.
- Cursor Right and then “5” moves the cursor to the right edge of the display page.
- Cursor Up and then “5” moves the cursor to the top line of the display page (the same effect is available via the Move to Top of Display Page command described earlier).
- Cursor Down and then “5” moves the cursor to the bottom line of the display page (the same effect is available via the Move to Bottom of Display Page command described earlier).

7.10 Extended Cursor Motion

The orginal Qunix Extended Cursor Motion command is still available in VE today, bound to the CTL-e key. Like the Edge command, when a cursor key is used, it sets a cursor direction. The Cursor Left key sets the direction to left, the Cursor Right key sets it to right and so on. When CTL-e is pressed, VE performs an extended cursor movement in the currently set cursor motion direction:

- Cursor Left and then CTL-e moves the cursor to column 0 of the line it is on, the logical equivalent of today's Home key.
- Cursor Right and then CTL-e moves the cursor to the end of the line it is on, the equivalent of today's End key.
- Cursor Up and then CTL-e moves up one screen in the file, the equivalent of today's Page Up key.
- Cursor Down and then CTL-e move down one screen in the file, the logical equivalent of today's Page Down key.

The Extended Cursor Motion command acquires its real utility in that it remembers the current cursor motion direction. Repeated use of the CTL-e key will continue to move in the indicated direction. Hence, Cursor Up, followed by repeated CTL-e keys will page up through the file, equivalent to repeatedly selecting the Page Up key. Hence, use of the CTL-e key allows easy paging up or down through a file.

7.11 Move Line to Top of Screen

The ESC-z or Keypad “/” commands move the line that the cursor is on to the top of the screen. The command letter 'z' is patterned after vi's 'z' command, which does the same thing.

7.12 Move to Top of File

The ESC-PageUp or ESC-\ commands will move the display window to the top of a file. The same effect can be achieved via the [Jump, Start] menu path (ESC-j, s). See Section 1.7 for information on how to make the Pg Up and Pg Dn keys available to VE under Mac OS X.

7.13 Move to Bottom of File

The ESC-PageDown or ESC-~ commands will move the display window to the bottom of a file. The same effect can be achieved via the [Jump, End] menu path (ESC-j, e). See Section 1.7 for information on how to make the Pg Up and Pg Dn keys available to VE under Mac OS X.

7.14 Jump Start, End, Line Number

The ESC-j command allows jumping to the start, to the end, or to a specificed line number of a file. This command presents a menu containing “Start”, “End” and “Line” selections. “Start” and “End” will move to the start and end of the file, respectively. The “Line” selection will prompt for a line number and then jump to that line. Line numbers in VE start at Line 1.

Note the VE also supports jumping directly to a line number upon startup, via the comand form:

ve filename -l line_number

7.15 Tags, and Jumping to Tags

VE supports a simple mechanism for tagging lines, and then jumping to a defined tag at a later time. This can be useful when it becomes necessary to check something in another part of a file being edited. In this case, the current line can be tagged, so that it may be returned to after other parts of the file have been viewed. The tagged line can be returned to at any time by simply jumping to the previously defined tag.

Defining Tags:

VE supports 10 named tags, named “0” to “9”. To tag the line the cursor is presently on, simply enter ESC-t and then a tag number, from “0” to “9”. VE will issue a message indicating that the named tag has been set. Setting a tag that has already been set silently redefines the tag – VE does not complain about overwriting existing tags. This is intended operation.

Listing Tags:

The set of defined tags for the current file can be viewed by issuing the [Tags, List] command (ESC-t, l menu path). VE will list all 10 tags, showing the assoicated line number, or “Undefined” if they have not been set.

Jumping to Tags:

VE supports three mechanisms for jumping to tags. The most obvious of these is the [Tags, Jump] command (ESC-t, j menu path) which prompts for a tag number (“0” to “9”) and then jumps to that tag. The same functionality is also available via the [Jump, Tag] command (ESC-j, t menu path). Finally, a keyboard shortcut of sorts is also supported, in the form of the (ESC-`, tagnum) menu path. This provides a three keystroke “jump to tag” capability – the previous two both require four keystrokes. The “`” key was chosen (the ` key is the unshifted version of the ~ key, on the upper left of a standard QWERTY keyboard) because it is in close physical proximity to both the ESC key and the number keys that are used to define the tags. Hence, this is a physically convenient set of keystrokes.

Tags and Multiple Files:

Tags are associated with a specific file. Each file that VE has open has its own unique tag definitions for each of the 10 available tags. You may freely move between files without worry about losing tag definitions.

Limitations of the Tags Function:

There are two salient limitations of VE's simple Tags functionality:

- Tags are dynamic associations that only last as long as VE has a given file open. They are not stored with the file. Hence, when a file is closed, all tags for that file are discarded.
- Tags are are a simple means of associating a line number with a tag symbol. The association is with the line number, not with the line itself. If a tag is defined and then the number of lines in the file is changed via text deletion, insertion, pasting, etc. the actual line that a given line number points to will be changed. Hence a jump to a tag may not achieve a jump to the expected line.

7.16 Current Line, Column Information

The ESC-i or Keypad “*” key command displays VE's current position in the file, including line, column and percentage through the file. It also displays the total number of lines in the file. A number of other VE commands, such as Page Up and Page Down, also present the same information, dynamically updating it as they move through the file.

7.17 Move to Next Open File

The ESC-n command or the Edit menu's “+” command (ESC-e, +) does not achieve motion within a given file but rather moves to the next open file in the set of files that VE presently has open. Repeated use of this comand will cycle through all available files. The Keypad '+' command is a shortcut for the [Edit, +] menu path (ESC-e, +, ENTER).

8.0 Finding and Replacing Text

VE provides a Find command for finding text and a Replace command for finding and then replacing text. These are documented below.

8.1 Finding Text

The ESC-f or F5 command command finds an occurrence of a specified string, and moves the cursor to the start of that string. When ESC-f or F5 is entered, VE will provide a “Find”: prompt, (which will be preloaded with the last find string specified if a Find command has been executed at an earlier time). The existing string may be directly re-used by simply pressing ENTER, or may be edited as desired (see the section earlier in this document on Status Line Response Line Editing for the full set of line editing features supported), or may be replaced with a new string (enter F10 or ESC-k to clear the existing string and then enter a new string). VE will search the file for the next occurrence of the specified Find string and move the cursor to the start of that occurrence.

The VE Find command supports two special search characters, allowing searches to include the beginning of a line, or strings that occur only at the beginning of a line, and a similar functionality for the end of a line, or strings that terminate at the end of a line. This is especially useful in conjunction with the Replace command (see below).

The default start-of-line character is '^' and the default end-of-line character is '$'. These defaults may be changed via the [Options, Match] menu path (ESC-o. m), using it's Start-of-line and End-of-line subcommands.

8.2 Replacing Text

The ESC-r or F6 command replaces one occurrence of a specified string with another specified string. If used with a repetition count, it will do multiple replacements sequentially. If used with the '*' repetition count, it will replace all occurrences in the file. When this command is entered, VE will provide a “Replace:” prompt, which will be preloaded with the last Replace string entered. Like the Find command, the existing string may be re-used directly by simply pressing ENTER, or may be edited as desired (see the section earlier in this document on Status Line Response Line Editing for the full set of line editing features supported), or may be replaced with a new string (enter F10 or ESC-k to clear the existing string and then enter a new string). VE will then provide a “with:” prompt, prompting for the string to use to replace the “Replace:” string with. Again, this will be preloaded with the last replacement value specified. This value may be used directly, edited to suit, or replaced, per the above. VE will then attempt the requested replacement and provide the results.

Note that the letters “ v” (a space and then lower case “v”) may be appended to the “with:” string to have VE prompt for verification of each replacement before it is made.

Note that using the replace command and specifying a Find string of '^' (start of line character) allows you to insert text at the beginning of each line. As a practical example, if you wished to temporarily comment out a number of C++ lines, you can issue the command:

ESC 8 r ^ //

This command runs the “r” command (the Replace command) 8 successive times, replacing the start of line for each of 8 lines with the C++ comment characters “//”.

8.3 Repeating Find/Replace Operations

The last find or replace operation may be repeated by simply pressing the F1 key (Again command). This can be done as many times as desired, or until no more occurrences of the search string can be found (VE will indicate this condition when it occurs).

9.0 Deleting Text

VE provides multiple mechanisms for deleting text. There are character oriented, single line oriented, line block oriented and column block (rectangle) oriented deletion tools. The line block and column block deletion tools are described later in this document with the other Line Block and Column Block manipulation commands. The remainder of the text deletion tools are described below.

9.1 Deleting Single Characters

The Keyboard Delete key (or ESC-D command) deletes the character that the cursor is presently on. Used with a repetition count, it will delete multiple characters at the same time.

9.2 Backspace/Rubout

The keyboard Backspace/Rubout key (or ESC-# command) deletes the character to the left of the cursor. Used with a repetition count, it will delete multiple characters at the same time. Pressing Backspace/Rubout on the first column of a line will move up to the previous line, deleting the implicit line end between the lines and thus join them.

9.3 Deleting Words

Pressing ESC-Delete (or the ESC-W command) deletes the word to the right of the cursor. Used with a repetition count, it will delete multiple words at the same time.

9.4 Delete to End of Line

The ESC-k (“kill”) or F10 command deletes all text to the right of the cursor, to the end of the current line. This includes the character that the cursor is on at the time the command is entered.

9.5 Delete To Start of Line

The ESC-d command deletes all text to the left of the cursor, moving the character that the cursor is presently on to the start of the line.

9.6 Delete Line

The ESC-l command (ESC, lower case letter L) or F9 deletes the line that the cursor is on. Used with a repetition count, it will delete multiple lines at the same time. The Delete Line command saves a copy of the deleted line(s) in the internal line buffer, allowing it to be pasted later via the line oriented paste command CTL-v. See the later section on “Operating on Blocks of Lines” for more information on this.

9.7 Delete Text Line Block

This command deletes blocks of text lines. See the documentation for copying, moving and deleting blocks of text lines, below.

9.8 Delete Text Column Block (Rectangle)

This command deletes rectangular areas of text spanning one or more lines and one or more columns. See the documentation for copying, moving and deleting text column blocks, below.

10.0 Operating on Blocks of Lines

VE provides a complete set of operations for manipulating blocks of text lines. Two essential mechanisms are supported, line-oriented and block-oriented. Similar functions are also available for column areas (rectangular areas of text), and are documented in the next chapter.

10.1 Line Oriented Copy, Cut and Paste

Line-oriented copy, cut and paste commands are intended for quick operations on small numbers of lines.

The following line-oriented operations are available:

- Copy Line(s) (ESC-C, CTL-c) – this command copies one or more lines to the internal line buffer. To copy just the line the cursor is on, type CTL-c. To copy up to "n" lines starting with and including the line the cursor is on, type "ESC n CTL-c".
- Cut Line(s) (ESC-X, CTL-x) – this command deletes one or more lines and copies them to the internal line buffer. To delete just the line the cursor is on, type CTL-x. To delete up to "n" lines starting with and including the line the cursor is on, type "ESC n CTL-x".
- Paste Line(s) (ESC-V, CTL-v) – this command pastes the contents of the internal line buffer into the file, ABOVE the line the cursor is presently on, except if the cursor is on the last line of the file. In this case, the contents of the internal line buffer are pasted BELOW the current line.

Any one of these commands can be preceded by a repetition count to operate on multiple lines at once. As an example:

ESC 5 CTL-c

will copy 5 lines into the internal line buffer.

The remainder of this chapter deals with VE operations oriented towards blocks of text lines (referred to as a "line block").

10.2 Defining a Line Block

After entering ESC-b or the F2 key, VE brings up the Block menu, and presents a visual marker on the screen, at the start of the line the cursor is presently on. Note that line blocks are complete lines – the cursor is moved to the start of the line to emphasize this. To perform the equivalent copy, cut and paste operations on partial lines use the Column Block commands (documented in the next chapter, below).

The visual marker is a right carat symbol, “>”. To define the block of interest, simply use cursor motion commands to move the marker. As cursor motion commands are used to move away from the first line of the block, a second marker will appear. All lines between the two markers (including the lines that the markers are on) define the current block. The “cursor up/down”, “page up/down”, “jump to line” and “move to start/end of file” commands may be used in any combination to position the second marker, thus defining a line block.

At any time during the definition of a line block, it is possible to “escape” out of the Block command entirely by simply pressing ENTER.

10.3 Line Block Copy, Cut, Paste, Wrap, Read, Write

Line-oriented text block copy, kill, cut and paste operations are intended for convenient manipulation of larger blocks of text lines. A block of text lines is defined as one or more full lines. VE operations are provided to visually define a line block (see above), copy it to the internal line block buffer, kill it, delete it (also copies it to the line block buffer), append it to previously copied lines in the line block buffer, paste the contents of the line block buffer, and write the line block buffer out to a file. An operation is also provided to read in an external file, make it's contents the defined line block and paste those contents into the file. VE provides a line block operation to perform after-the-fact text wrapping on a whole line block at once. Finally, VE provides and operation to exchange the line block buffer and the column block buffer, allowing line block operations to be performed on column blocks and visa versa.

The following line block-oriented operations are available. Simply type the first letter of the command to execute it:

- Copy Block (“c” subcommand of Block Menu) – this command copies the selected block into the internal block buffer. This command can be used in conjunction with the Paste Block command (below) to copy a block of text from one part of a file to another. Simply copy the block using the Copy Block command, then move to the intended target area, and use the Paste Block command (below) to paste a copy into the file.
- Append Block (“a” subcommand of Block Menu) – this command is similar to the Copy Block command (above), but has the added attribute that the defined line block is **appended** to the current contents of the internal line block buffer, as opposed to replacing any current contents, which is the behavior of the Copy Block command. The Append Block command allows the accumulation of sections of text into the block buffer before pasting them elsewhere or writing them out to a file.
- Kill Block (“k” subcommand of Block Menu) – this command deletes the selected line block from the file without saving a copy of it in the internal line block buffer.
- Delete Block (“d” subcommand of Block Menu) – this command deletes the selected line block from the file and copies it to the internal line block buffer. This command can be used in conjunction with the Paste Block command (below) to move a block of text lines from one part of a file to another. Simply delete the line block using the Delete Block command, then move to the intended target area, and use the Paste Block command (below) to paste a copy into the file at the new location, thus moving the line block.

- Paste Block (“p” subcommand of Block Menu) – this command pastes the contents of the internal line block buffer into the file, ABOVE the line the cursor is presently on. An exception to this occurs if the cursor is on the last line of the file. In this case, VE will paste the line block buffer contents in BELOW the line the cursor is on.
- Wrap Block (“w” subcommand of Block Menu) - this command performs after-the-fact text wrapping on the defined line block. The entire block will be wrapped to fit between the defined left and right margins. Note that this command is paragraph-oriented. If the defined line block does not enclose complete paragraphs, this command will find the paragraph start nearest to the start of the block, and the paragraph end nearest to the end of the block, and wrap that complete range.
- Read Block (“r” subcommand of Block Menu) – this command presents the VE Open File Dialog to allow selection or entry of a filename, reads that file into the internal line block buffer, and then pastes the result into the currently editing file ABOVE the line the cursor is presently on. In essence, this command provides a way of inserting an external file into the file currently being edited. Like the Paste command, an exception occurs if the cursor is on the last line of the file – in this case the paste will occur BELOW the line the cursor is on.
- Write Block (Save As) (“s” subcommand of the Block Menu) – this command writes the currently defined line block OR the contents of the internal line block buffer out to a separate file. This command presents the VE Open File Dialog to prompt for a filename, and then writes the currently defined line block or the contents of the line block buffer out to the indicated file. The filename may be directly typed, or users may select an existing file to overwrite. VE will prompt before overwriting any existing file. The block may be written to directories other than the current directory by either typing the full pathname to the new directory as part of the filename, or using the Open File Dialog to navigate to the directory of interest before typing in a filename.

  This command incorporates a useful subtlety, allowing a block of text lines to be written out to a file without disturbing the current contents of the internal line block buffer. If a block is defined using the Block Command (ESC-b, F2), and then the Write Block subcommand is used, the currently defined line block will be written out to the requested file, but the present contents of the internal line block buffer will not be effected. Alternately, if a block is defined using the Block Command (ESC-b, F2) and copied to the internal line block buffer, and then at a later time, the [Block Write] subcommand is used, it will instead write out the contents of the line Block Buffer. The “useful subtlety” here is that areas of your file can be written out to separate files without impacting the contents of the internal line block buffer if so desired.

  Flush Block (“f” subcommand of the Block Menu) – this command does as its name suggests, emptying (flushing) the contents of the internal line block buffer. This may be useful when editing an extremely large number of extremely large files, or when a lot of text lines have been placed into the internal block buffer. This command frees the associated memory, thus assisting in low memory situations.

  As may be anticipated, this subcommand traces its heritage back to VE's initial QUNIX implementation, on the 8086-based NABU-1600, with only 512K of memory. Conservation of memory was somewhat important in that environment, giving rise to this command. It is not anticipated that this command will be greatly needed in today's gigabyte computers.

10.4 Repeating Line Block Commands

Like most VE commands, the Block command (ESC-b or F2) can be used with repetition counts to achieve multiple executions with a single user command. Alternately, the VE Again command (ESC-a or F1) can be used to repeat the last executed Block command as many times as desired. When used with a repetition count, only the initial repetition will intereact with the user to define the line block. All subsequent executions will use the block defined in the first repetition. The same is true of repetitions accomplished via the VE Again command. Executing Again (ESC-a or F1) after a Block command will re-use the last Line Block definition without requiring the user to define the same block again for each repetition.

10.5 VE Paste Command

As an added convenience, VE makes the Block Paste command available as a unique command, Paste (ESC-p). This command simply runs the Paste Block command, but does not require the user to re-enter the Block command (ESC-b or F2) first. This minimizes the number of keystrokes needed for this very common operation (pasting copied lines). The VE Paste command, like the [Block, Paste] command, can be used with repeitition counts, or with the VE Again command. For example:

ESC 6 p

will paste 6 successive copies of the contents of the internal line block buffer into the file.

11.0 Operating On Column Areas (Rectangles)

In addition to support for manipulation of blocks of text lines as a single entity, VE also provides the powerful capability to manipulate ranges of text columns as a single entity. This entity is referred to throughout this documentation as a “column area”, or a “rectangle”.

A “column area” is a rectangular text area consisting of one or more columns spanning one or more lines. A column area can be as small as a single character on a single line, or as large as all of the characters on all of the lines of the file. Using column areas, VE users can cut and paste single words, partial lines of text, whole columns of data (for example in column oriented output of a data reporting tool) or whole files. The ability to operate on column area is an extremely flexible and powerful capability within VE.

Column areas always act on the existing lines of a given file – they can alter existing lines, but they cannot be used to create new lines. In essence, VE's column area operations allow users to insert, copy, cut and paste horizontally, while VE's line block operations allow users to insert, copy, cut and paste vertically. To bridge the gap between these, VE also provide the ability to exchange the line block buffer and the column area buffer, allowing column area operations to be performed on line blocks, and visa versa.

11.1 Defining a Column Area (Rectangle)

After entering either ESC-c or F3, VE will execute the Column command, which will present the Column menu, and place a visual marker onto the screen at the current cursor position. Defining a column area is a two step process. First, a line range is defined, and then a column range is defined, thus specifying a rectangular area within the file that will be the target for a column operation.

The definition of a column area is similar in operation to the definition of a line block. Column Area definition is carried out interactively by moving visual markers on the display page to define first the line range and then the column range.

For defining the line range, the visual marker is the same right carat symbol (“>”) used in line block definition. To define the line range of interest, simply use cursor motion commands to move the marker. As cursor motion commands are used to move away from the first line of the range, a second marker will appear. All lines between the two markers (including the lines that the markers are on) define the line range of the column block. The “cursor up/down”, “page up/down”, “jump to line” and “move to start/end of file” commands may be used in any combination to position the second marker, thus defining the line range for the column area.

When the line range has been satisfactorily defined, use of either the cursor right or cursor left keys will begin column range definition. VE will return the cursor to the initial position it was at when the Column command was executed (on the line that is at the top of the defined line range), and replace the displayed line markers with column markers. If the defined line range is only a single line, only one column marker will be displayed. The column marker is conceptually a “down carat” symbol pointing to the column of interest. Since no such standard text symbol exists, VE uses the letter “capital v” - “V”.

To define the column range of interest, cursor motion commands may now be used to move the markers. As cursor motion commands are used to move away from the first column of the range, a second set of markers will appear. All columns between the two markers on each line (including the columns that the markers are on) define the column range of the column area. All text enclosed within the four displayed markers is included in the defined column area. Note that if the defined line range for the column area is only a single line, just two markers will appear. The “cursor left/right”, “Home/END”, and “Word Left/Right” commands may be used in any combination to position the second set of column markers, thus defining the column range.

When the column range has been satisfactorily defined, a full rectangle has been specified. To terminate column area definition and execute a command on the defined column block, simply enter the first letter of any selection from the Column command's menu (which has been continuously presented on the status line during column block definition). VE will remove all displayed markers and execute the entered command.

The method for defining a column area may be abbreviated where this makes sense. For example, if the area of interest is a single word or phrase on the current line, it is possible to go directly to column range definition without defining a line range. To do this, simply enter a cursor left/right key as the first keystroke after the Column command presents the line range definition marker. VE will immediately move to column definition, changing the line marker to a column marker. Similarly, if the area of interest is only one column wide, simply enter the first letter of any of the Column command's menu items immediately after defining the line range and pressing cursor /eft/right to enter column range definition.

In its simplest form, the definition of a column area can consist of one cursor left/right key, followed by a Column command. This defines and operates on a single character column area!

Finally, at any time during the definition of a column area, it is possible to “escape” out of the Column command entirely by simply pressing ENTER.

11.2 Column Area Insert, Copy, Cut, Paste, Move, Read, Write

VE provides a full set of Column commands to operate upon a defined column area (rectangular area of text). Commands are provided to insert a blank column area (useful for indenting lines of text), kill a column area (conceptually the inverse of the above), copy a column area to the internal column area buffer, delete a column area (also copies it to the internal column area buffer), paste a column area from the internal column area buffer, read a column area in from an external file into the column area buffer and then paste it, write the defined column area out to an external file, empty the internal column area buffer and finally, exchange the column area and line block buffers. Each of these capabilities is detailed below.

- Insert Columns ("i" subcommand of Column menu) – this command inserts a number of blank columns into the file starting at the cursor position at time of Column command execution. The number of columns inserted is determined by the width of the defined column range. This command is extremely useful for indenting blocks of source code, etc.

  Per the above information on abbreviation of column area definition, this command may be entered without defining a column range, which results in a single column being inserted. This command can also be used without even the initial line range definition, resulting in a single space being inserted at the current cursor position.

- Kill Columns ("k" subcommand of Column menu) – this command is the philosophical inverse of the Insert command. It deletes the defined column area without saving a copy of it to the internal column area buffer. Like Insert, this command is also extremely useful for changing the indentation of blocks of source code, etc.

  Per the above information on abbreviation of column area definition, this command may be entered without defining a column range, which results in a single column being deleted. Like the Insert command, this command can also be used without even the initial line range definition, resulting in a single character being deleted at the current cursor position.

- Delete Columns ("d subcommand of Column menu) – this command deletes the defined column area in the same manner as the Kill command, but saves a copy of it in the internal column area buffer prior to doing so. Any previous contents in the column area buffer are discarded and replaced with the new column area being copied. Coupled with the Paste Columns command (see below) this is command can be used to move column areas of text to different locations in the file.

  Per the above information on abbreviation of column area definition, this command may be entered without defining a column range, which results in a single column being deleted. Similar to the Kill command, this command can be used without even the initial line range definition, resulting in a single character being deleted at the current cursor position.

- Copy Columns ("c" subcommand of Column menu) – this command creates a copy of the defined column area in the internal column area buffer. Any previous contents in the column area buffer are discarded and replaced with the new column area being copied.

  Per the above information on abbreviation of column area definition, this command may be entered without defining a column range, which results in a single column being copied. Like most other Column commands, this command can be used without even the initial line range definition, resulting in a single character being copied at the current cursor position.

- Paste Columns ("p" subcommand of Column menu) – this command pastes the current contents of the internal column area buffer into the file, starting at the current cursor position. The contents are pasted to the right of the current cursor position.

  Per the above information on abbreviation of column area definition, this command may be entered without defining a line and or column range. In this case, VE will attempt to paste the current contents of the internal column area buffer into the file with the first column of the first line of that buffer being placed at the current cursor position. If the column area in the column buffer will not fit at the current cursor position (for example, there are not enough lines between the current cursor position and the end of file) VE will indicate this condition and not do the paste.

  Use of the Paste Columns command without an initial line and column range definition is in fact the intended and most convenient way to use this command. To copy a column area, the Copy Columns command is used, which requires a full column area definition. Then the cursor is moved to the intended target area, the Column command is issued again, and "p' (Paste Columns) is selected from the Column menu. This pastes the previously copied column area at the new location.

- Read Column Area From File ("r" subcommand of Column menu) – this command presents the VE Open File Dialog to prompt for an external file name and then reads the contents of the selected file into the internal column area buffer. Any previous contents of that buffer are discarded and replaced with the new content. The contents of the column area buffer are then pasted into the file at the current cursor position. Conceptually and in implementation, this is simply a file read, followed by a Paste Columns command. After the filename prompt, its behavior is identical to the Paste Columns command.

- <u>SaveAs – Write Defined Column Area to File</u> ("s" subcommand of Column menu) – this command writes the defined column area out to an external file, prompting for the file's name via the VE Open File Dialog. The file's name may be directly typed, or an existing file may be selected to be overwritten. VE will prompt before overwriting any existing file. The defined Column Area may be written to a directory other than the current directory by either typing in the full pathname of the intended directory as part of the filename, or using the Open File Dialog to navigate to the directory of interest before entering the filename. Like most other column commands, this command can be used without either the column range definition or even the line range definition, resulting in a single column or a single character being written out to file, respectively.

  Like its SaveAs counterpart in the Line Block commands, this command has a subtle twist in operation which can be exploited as desired. If used with a column area definition, this command will write the defined area out to file. If used without a column area definition, this command will write the current contents of the internal column area buffer out to file. Hence, it is possible to save defined column areas to file without impacting the current contents of the internal column area buffer.
- <u>eXchange Column/Line Buffers</u> ("x" subcommand of Column Menu) – this command exchanges the contents of the internal column area buffer and the internal line block buffer. This allows saved column areas to be treated as full lines, and full lines to be treated as column areas, as desired. As an example of potential usage, column areas can be saved or deleted from one area of a file and pasted into another as full lines, achieving the ability to move text both horizontally and vertically. Many creative uses of the eXchange command are possible.
- <u>Empty Column Area Buffer</u> ("e" subcommand of Column Menu) – this command is the logical equivalent of the Flush command in the line Block menu. It empties the current contents of the internal column area buffer and frees all associated memory.

11.3 Repeating Column Area Commands

Like most VE commands, the Column command (ESC-c or F3) can be used with repetition counts to achieve multiple executions with a single user command. Alternately, the VE Again command (ESC-a or F1) can be used to repeat the last executed Column command as many times as desired. When used with a repetition count, only the initial repetition will intereact with the user to define the column area. All subsequent executions will use the column area defined in the first repetition. The same is true of repetitions accomplished via the VE Again command. Executing Again (ESC-a or F1) after a Column command will re-use the last Column Area definition without requiring the user to define the same column area again for each repetition.

12.0 Saving Your Work

VE provides three commands for saving your work. The first is the Update command; the second and third are subcommands of the Quit Command. These allow you to write your file out to another filename or to save your work and then optionally quit the VE editor.

12.1 Updating Your File As You Work

The ESC-u or F11 command (the Update command) writes the current contents of the file out to disk and then returns to editing the file. This provides a convenient way to checkpoint your work as you go. This is equivalent to the [Edit, Save] dialog on most GUI editors.

The Update command can be used even if the current file has not yet been named (is a new unnamed file). An attempt to Update such a file will cause VE to prompt for a filename. The file will then be written out under that name, which will then become the file's current name.

12.2 Saving Your Work To Another File

The “write” subcommand of the Quit command (ESC-q/F12, w) writes the current file out to another filename. This subcommand presents the VE Open File Dialog to prompt for a filename. The intended file name may be directly typed, or an existing file may be selected to be overwritten. VE will prompt before overwriting any existing file. The file may be written to a directory other than the current directory by either typing in the full pathname of the intended directory as part of the filename, or using the Open File Dialog to navigate to the directory of interest before entering the filename. Once a directory and file have been selected, this command writes the current file to that filename. This subcommand does not change the name of the current file, but merely writes a copy under a different name.

12.3 Saving Your Work and Exiting

The “save” subcommand of the Quit command (ESC-q/F12, s) updates the current file to disk and then exits VE (or switches to another open file, if more than one file is open).

13.0 Editing Multiple Files

13.1 Multiple File Support Overview

VE incorporates strong support for editing large numbers of files (up to 256) simultaneously. VE was conceived from "day 1" as a multi-file editor and this functionality is fully integrated throughout the program.

VE can manage up to 256 open files simultaneously. VE designates one of the open files as the "current file" and displays that file on the screen. All other open files are open and available, but cannot be seen unless they are made the current file. This can be accomplished by any of a number of means, the simplest of which is the Goto File command (ESC-g).

There is no need to update/save a file before switching to another open file. Moving to another file simply changes which of the files is displayed on the screen. It does not impact any other aspect of the file. VE allows users to switch freely between all available files, editing them all at the same time. Think of the display window simply as a resource that can be attached to any open file and you have a fairly accurate view of how VE treats the association between the display window and the available file set.

13.2 Opening Multiple Files From the Command Line

VE can be started from the command line with multiple files, and after it is running, files can be freely added, discarded, and replaced.

To open multiple files from the command line at start up, simply put the files of interest on the command line that launches VE, per the below example:

ve file_1 file_2 file_3 ... file_n

Wildcards can be used if they are supported by the shell in use. For example:

ve *.c

will open all C source files in the current working directory.

13.3 Opening Additional Files from Within VE

Once VE is running, additional files may be added using the Edit command (ESC-e, F8). The File and New subcommands of this menu will add an additional file to those already opened, and make the new file the current file (i.e. display it on screen).

The File subcommand ("f" subcommand of the Edit menu) presents the VE Open File Dialog to prompt for a filename and attempts to open the indicated file. If the file exists, it is opened and made the current file. If it does not exist, an error message to that effect is posted to the messages area of the status line.

The New subcommand ("n" subcommand of the Edit Menu) opens a new unnamed file and makes it the current file (i.e. displays it on screen).

13.4 Discarding Files from Within VE

While VE is running, any currently open file may be discarded by using the [Quit, Discard] command path (ESC-q, d). This subcommand will prompt for confirmation if the file has been changed since it was opened or last saved. The Discard subcommand discards the current file and either moves to another file or presents a menu of available files and prompts for a file to move to. If only one other file is open, Discard simply moves to it. If more than one file is open, Discard will display the menu of presently open files and prompt for the file to move to.

13.5 Moving Between Open Files Sequentially

Entry of ESC-n or the Keypad "+" key (when run from within an xterm) causes VE to cycle forward to the next open file, making it the current file and displaying it on screen. Repeated use of any of these commands will cause VE to cycle through all open files, wrapping around to the start when the last open file is reached.

This functionality is also available from the Edit command's "+" menu selection. Note that as a convenience, this command supports use of the "=" command as well, as this is the unshifted version of the "+" key.

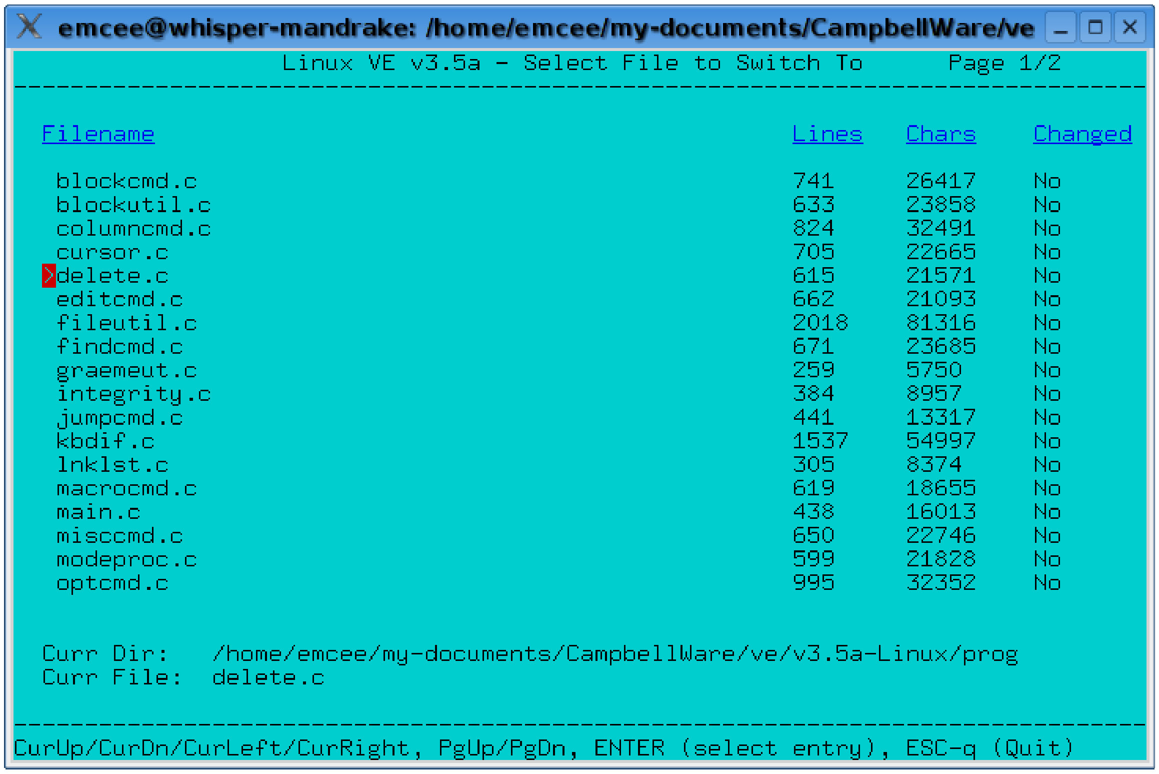
13.6 Moving Between Open Files via Goto File Dialog

The Goto File command (ESC-g) and the Keypad "-" key both cause VE to present its Goto File Dialog, an interactive full screen menu of all available files which allows users to select the next open file to move to.

This Dialog presents a number of information elements regarding each file:

- The file's name – the name of the file, potentially shortened to fit the available space
- The file's size in lines and chars
- The file's modification status – has it been changed since it was opened or last saved

A screen shot of the Goto File Dialog appears below:



emcee@whisper-mandrake: /home/emcee/my-documents/CampbellWare/ve

```
                   Linux VE v3.5a - Select File to Switch To      Page 1/2
--------------------------------------------------------------------------------

  Filename                                                  Lines    Chars    Changed

   blockcmd.c                                               741      26417    No
   blockutil.c                                              633      23858    No
   columncmd.c                                              824      32491    No
   cursor.c                                                 705      22665    No
  >delete.c                                                 615      21571    No
   editcmd.c                                                662      21093    No
   fileutil.c                                               2018     81316    No
   findcmd.c                                                671      23685    No
   graemeut.c                                               259      5750     No
   integrity.c                                              384      8957     No
   jumpcmd.c                                                441      13317    No
   kbdif.c                                                  1537     54997    No
   lnklst.c                                                 305      8374     No
   macrocmd.c                                               619      18655    No
   main.c                                                   438      16013    No
   misccmd.c                                                650      22746    No
   modeproc.c                                               599      21828    No
   optcmd.c                                                 995      32352    No


  Curr Dir:   /home/emcee/my-documents/CampbellWare/ve/v3.5a-Linux/prog
  Curr File:  delete.c

--------------------------------------------------------------------------------
CurUp/CurDn/CurLeft/CurRight, PgUp/PgDn, ENTER (select entry), ESC-q (Quit)
```

Cursor up and down keys may be used to move the file selection carat to the file of interest, which may then be selected by pressing ENTER. If the listing of currently open files exceeds one display page, the PgUp and PgDn keys may be used to move to the next/previous page of the listing as desired. As an alternate to using the PgUp or PgDn keys, the cursor up and cursor down keys may be used. When they reach the top or bottom of the displayed files, if there is another page, they will wrap to that page. In similar fashion, the PgUp and PgDn keys will wrap from the first page to the last, and visa versa.

The Goto File Dialog provides mouse support. Single clicking the mouse on a displayed entry will move the selection carat to that entry. Double clicking on a displayed entry will select the file that is double clicked and complete the dialog, as if the selection carat had been moved to the file of interest and ENTER had been pressed.

As an alternate to interactively selecting a file of interest, the file's may simply be directly typed. It will be echo'd in the Curr File field, complete with full line editing support.

Users may cancel out of the Goto File Dialog at any time without selecting a file by pressing ESC-q. F12 achieves the same result.

Note that the cursor movement and PgUp, PgDn functions may also be selected by the same ESC-command_letter and keyboard shortcuts that apply to these keys while editing files in VE. Specifically, the following alternate forms are available while in the Goto File Dialog:

● Cursor Up -	CTL-u, ESC-^
● Cursor Down -	CTL-d, ESC-$
● Page Up -	CTL-p, ESC-B
● Page Down -	CTL-n, ESC-F

14.0 Exiting from VE

VE's Quit command (ESC-q, F12) provides a complete set of options for exiting VE, with or without saving the currently open file(s). The following Quit subcommands are available (in the Quit menu, simply type the first letter of the command):

- All/Absolute ("a" subcommand of the Quit menu) – this subcommand will unconditionally exit VE, even if one or more of the currently open files has been modified since opened or saved last. If none of the open files has been changed since it was opened, VE simply quits with no further dialog. If one or more of the open files has been changed since it was opened, VE will prompt for confirmation, indicating the number of changed files. If confirmation is provided (type "y" to the prompt), VE discards all open files and quits.
- Discard/Quit ("d" and "q" subcommands of the Quit menu) – this subcommand discards the current file, and either moves to another open file, or quits VE if no other files are open. The Discard/Quit subcommand will prompt for confirmation if the current file has been modified since it was opened or since the last time it was saved. If only one other file is open, VE will simply move to that file. If multiple files are open, VE will present the Goto File Dialog to prompt the next file to move to.
- Replace ("r" subcommand of the Quit menu) – this subcommand provides a convenient mechanism for chaining through a number of files, one by one. The Replace subcommand discards the current file, presents the VE Open File Dialog to prompts for a new filename, and then opens the selected file. In essence, it replaces the current file with a new file. If no file is selected (user cancels out of Open File Dialog) VE simply quits.
- Save ("s" subcommand of the Quit menu) – this subcommand saves the current file to disk and then either quits VE if no other files are open, or presents the VE Goto File Dialog to prompt for the next file to move to. Like the Discard command, if only one other file is open, VE will simply move to it. If more than one file is open, VE will present the Goto File Dialog to prompt for the file to move to.
- Write ("w" subcommand of the Quit menu) – this subcommand writes the current file to another filename. It presents the VE Open File Dialog to prompt for a filename and then writes the current file to that filename. Files may be written to directories other than the current directory by either typing in the complete pathname (directory plus filename) of the intended file or using the Open File Dialog to navigate to the directory of interest before entering the filename. Of course, the option also exists to select an existing files to be overwritten. VE will prompt before overwriting any existing file. Note that the Write subcommand does not change the name of the current file, but merely writes a copy under a different name.
- Update ("u" subcommand of the Quit menu) – this subcommand saves the currently editing file to disk and then returns to editing that file. As such, it does not quit VE at all, but is providing in the Quit menu as a convenience.

15.0 VE Macros

15.1 VE Macros Overview

VE supports the definition and execution of keyboard macros. This is a capability that allows VE to learn keystrokes as they are entered and then play them back on command, as if they were being directly typed. This allows you to automate complicated sequences of commands, repeating them at will. Once defined, macros can be saved as macro files and read back in to VE as needed, allowing them to be saved between editing sessions.

VE's macro capabilities are accessed via the obviously named Macro command (ESC-m). This command presents a submenu that contains commands to create macros, execute macros, save macros, load macros and list the currently available macros. These menu items are detailed below.

15.2 Creating a Macro

Macros are created via the [Macro, Create] menu path (ESC-m, c). VE will prompt for a macro name, and will then enter macro definition mode. This is indicated by the “m” showing in the options area of the status line. Any desired name may be used for a macro – VE has no predefined internally used names. Names may range from 1 character long to 20 characters long.

In macro definition mode, VE learns all keystrokes as entered, recording them into the macro being created. When all of the desired keystrokes for the macro have been entered, the Macro definition is completed by returning to the Macro command and using the Stop menu item (ESC-m, s). The [Macro, Stop] command keys are not recorded with and do not form part of the macro.

15.3 Executing a Macro

To execute a macro, position the cursor at the desired starting point for the the macro and enter the [Macro Xecute] command (ESC-m, x). VE will prompt for a macro name, and then execute that macro.

15.4 Repeated Macro Execution

Macros may be repeatedly executed, via either repetition counts or via the Again command (F1 or ESC-a). To facilitate this, VE makes the [Macro, Xecute] command available as the free standing command ESC-x. This allows its use with repetition counts, and its repeated execution via the Again command. For example, “ESC 5 x” will prompt once for a macro name, and then execute that macro five times. “ESC-x” will prompt for and run a macro. The Again command will repeat that macro as often as wished, without reprompting for the macro name, including and repetition count that was part of the original macro Xecute command.

15.5 Listing the Available Macros

The [Macro, List] command (ESC-m, l) will list all of the available macros in a full screen panel. At the bottom of the screen VE presents the usual full screen navigation guide, with ENTER as the only option. Pressing ENTER will return to editing the current file.

15.6 Deleting Macros

Any macros may be deleted with the [Macro, Delete] command (ESC-m, d). VE will prompt for the macro name and then delete the macro.

15.7 Saving Macros to Disk

Macros may be saved to disk files, for later reloading. Use the [Macro, Write] command to achieve this (ESC-m, w). VE will prompt for a file name and write the macro out to the indicated file. VE macro files are not text files and are not human readable or editable.

15.8 Loading Macros from Disk

Macros may be reloaded from disk files via the [Macro, Read] command (ESC-m, r). When entered, VE will prompt for a filename, and attempt to read a macro from that file. If the file exists, VE will load the macro and make it available for execution.

16.0 Miscellaneous Commands

16.1 Redrawing The Screen

In the unlikely event that VE should encounter a bug that corrupts the screen image, that image may be redrawn via the View command (ESC-v).

16.2 A Look At VE's Internal Status

For the curious, VE provides the Status command (ESC-s), which provides selected details on the current internal status of VE. This command lists multiple information points:

- Current filename
- Current file size, number of lines
- Current file format
- Line Buffer size, number of lines
- Line Block Buffer size, number of lines
- Column Area Buffer size, number of lines
- Total amount of memory in use for files and buffers
- Status of the Undo System
- Number of files currently open
- Number of macros defined

This information is presented as a full screen panel. At the bottom of the screen the usual full screen navigation guide is presented, with ENTER as the only option. Pressing ENTER will return to editing the current file.

16.3 About VE

The About VE command provides some brief information on the origins and history of the VE text editor. The same information is available in the opening pages of this manual.

This information is presented as a full screen panel. At the bottom of the screen the usual full screen navigation guide is presented, with ENTER as the only option. Pressing ENTER will return to editing the current file.

17.0 Configuring VE

17.1 Configuring Screen Colors

VE defaults to using the colors of the console or xterm from which it was started. However, these colors may be changed via the [Options, Colors] menu path (ESC-o, c).

The [Options, Colors] command presents a submenu which allows you to set the foreground (text) and background colors. As you change these colors, they become the “working set” of colors, but have not yet been applied to the display. The Apply subcommand applies the colors to the display, making them the actual screen colors. As with all VE options, the selected colors are saved to the file ~/.ve.opt when the [Options, Save] command is entered (ESC-o, s). Subsequent starts of VE will load this file and change the colors to those specified in the it. Hence, to “permanently” change the colors used by your version of VE, change the colors to suit your taste, and then save your options. Henceforth, VE will use the new colors.

The [Options, Colors] menu selection presents the following choices – simply type the first letter of the command to select it:

- Palette – selecting Palette displays the available colors on the status line. The special color “Default” represents the colors of the console or xterm from which VE was started.
- Displayed – selecting Displayed shows the names of the curretly displayed foreground and background colors.
- Working – selecting Working shows the names of the current working set of foreground and background colors.
- Foreground – selecting Foreground prompts you to select a new foreground color for the working set from the set of colors that are available (see Palette). The current working foreground color is included as part of the prompt. To simply keep the current foreground color, press ENTER. To change it, enter an available color name and press ENTER. The new color becomes the working foreground color, but is not yet applied to the display.
- Background – like Foreground, but manipulates the working background color.
- Apply – Selecting Apply applies the current working colors to the display, making them the displayed colors.

17.2 Configuring Auto Indent

With its default settings, VE auto indents. This means that when ENTER is pressed at the end of a line of text, VE will automatically indent the next line to the same level of indentation as the line just typed. This behavior may be enabled or disabled via the [Options, Indent] menu path (ESC-o, i). Here, it is possible to select “Enable” or “Disable” to control this function.

17.3 Configuring Tabs

Tabs may be configured via the [Options, Tabs] menu path (ALT-o, t). The Tabs submenu supports enabling and disabling of input and output tab translation, and also allows the tabstop value that will be used to be modified.

Input tab translation will expand all tabs encountered as a file is read in, expanding them to spaces according to the current tab stop. The default value for Tabstop is 4. This can be changed via the [Options, Tabs, Tabstop] menu path (ESC-o, t, t). Input tab translation is ON by default.

Output tab translation compacts all possible white space into tabs before writing the file, again using the current tab stop. In keeping with the general idea of compacting the size of the file, output tab translation also strips any trailing white space from the end of lines, to minimize the size of the file.

Tab translations only effect the file on input and output. They have no impact on the in-RAM version of the file. Turning input translation on after loading a file containing tabs without input translation on will not expand the tabs on the fly. Instead, they will be represent in the file by the special tab character '?'. Similarly, turning output tab translation on and writing the file out does not compact the currently editing version of the file, but only the written version.

17.4 Configuring Find/Replace's Start/End Match Characters

As outlined earlier in this document, VE supports two special characters that match the start of a line and the end of a line. These characters can be used in Find/Replace strings to match the start or end of a line, or words that occur at the start or end of a line. With its default settings, VE uses the “^” character to match the start of a line, and the “$” to match the end of a line. These characters may be changed with the [Options, Match] menu path (ESC-o, m). Here, using the Start-of-Line and End-of-Line menu selections (type “s” or “e”), it is possible to change the character used.

17.5 Configuring Text Wrapping

See the earlier section of this document on Formatting Text for full details regarding the use and configuration of text wrapping.

17.6 Configuring Text File Format

As outlined earlier in this document, VE fully supports both Mac OS X/Unix/Linux and DOS/Win formatted text files (Mac OS X/Unix/Linux uses just the Line Feed character as a line end; DOS/Win uses both Carriage Return and Line Feed). By default, VE determines the text format of a file as it is read in, and uses the same format when it is written out.

The format used to write any given file out can be modified while the file is open by using the [Options, Fileformat] menu path (ESC-o, f). Here it is possible to select “unix” or “dos” (type “u” or “d”). The selected format will be used to write out the current file. However, this is not a global setting. VE's default behaviour is always to write any given file in the format it was read in. If the current file format is changed with the [Options, Fileformat] menu path, that change is only applicable to the file VE is presenting at the time. All other files will continue to be written in the format that they were read in.

17.7 Configuring Mouse Support

With its default settings, mouse support is enabled in VE when running in an xterm under Apple X11. Mouse support is not available in Terminal windows. This means that it is possible to position the cursor to any location on the display window by moving the mouse pointer to that location and clicking the left mouse button, and to use the mouse within the Open File and Goto File dialogs to select displayed files. This behavior may be enabled or disabled via the [Options, Rodent] menu path (ESC-o, r). Here, it is possible to select “Enable” or “Disable” to control this function.

A major reason for temporarily disabling VE's mouse support is to enable copying and pasting of information into an open file via X-windows copy and paste operations. These operations require that X, not VE, see and interpret mouse events. Per the earlier section on Mouse Support, it is possible to perform X-based copy and paste operation while VE's mouse support is enabled by simply pressing SHIFT at the same time as the desired mouse operations. This is known to work in an xterm, but may be less reliable in a console.

If the VE options are saved after the mouse support setting is changed, that setting will become VE's default. See the section below on saving VE's configuration.

17.8 Saving The VE Configuration

The VE options configuration may be saved at any time via the [Options, Save] menu path (ESC-o, s). This writes the current options to the file “.ve.opt” in the current HOME directory. When VE starts up, it looks for the .ve.opt file. If found, it will load its options from there. If not found, it uses its internal defaults. Hence, VE can be “permanently” customized by setting options as desired, and then saving those options via the [Options, Save] menu path. VE will use those options on all subsequent starts. This functionality is only active if the HOME environment variable is set.

17.9 Loading The VE Configuration

VE will autoload its options from the file ~/.ve.opt each time it starts. Hence, if options have been previously saved via the Option command's Save menu item, those options will be automatically restored each time you start VE. This functionality is only active if the HOME environment variable is set.

18.0 Miscellany

18.1 Repetition Counts Default Commands to Unmodified Form

The meaning of several special keyboard keys can be modified by pressing ESC key and then the special keyboard key. An example of such keys are the cursor left and cursor right keys. Entered by themselves, they move one character to the left or the right. Preceded by ESC, they are modified to move one **word** to the left or right.

While use of a preceding ESC or ALT to modify the meaning of a command key is a simple and easily understand notation, it creates a logical dilemma when a repetition count is entered, since repetition counts must be preceded by an ESC to move into Command mode first. So, if “ESC 5 cursor_right” is entered, does it mean “move 5 chars to the right” or “move 5 words to the right”? VE resolves this dilemma by assuming that the ESC is logically part of the repetition count, not part of the command, and thus defaults the command to its unmodified form. In the example above, VE defaults the command to “move 5 chars” not “move 5 words”.

As a practical consequence of this, it is not possible to enter a repetition count with any of the modified commands available via ESC/ALT and such special keyboard keys. To enter a repetition count with such a command, it is necessary to use the “ESC-letter” form of the command, rather than the special keyboard key form. In the example above, to move to the right five words, it is necessary to enter “ESC 5]”.

The keyboard keys in this category include:

• Cursor Left:	char left or word left
• Cursor Right:	char right or word right
• Cursor Up:	line up or paragraph up
• Cursor Down:	line down or paragraph down
• Page Up:	page up or move to top of file
• Page Down:	page down or move to end of file
• Delete:	delete char or delete word

18.2 The “*” Repetition Count Isn't Actually Infinity

Previously in this document, it has been noted that to replace all occurrences of something using the replace command, a repetition count of “*” should be used. This remains true. However, it is worth noting that “*” is actually translated by VE into the “very large number” 32,000 (near the safe limit of an unsigned 16 bit integer, the lowest common denominator for integer sizes). If a “*” repetition count is used, and there are more than 32,000 occurrences to replace, it may be necessary to repeat the replace operation (use the Again command (ESC-a or F1).

18.3 VE File Writes Overwrite “filename~”

To protect users from potential file content loss, whenever VE writes out a file, it first writes to a temporary file, and then if that write succeeds, deletes the original and renames the temporary to the original's filename. This approach protects users in the unlikely event that a bug in VE's file writing code results in loss of file contents.

VE constructs the temporary file name by overwriting the last character of the current file's name with the character “~”. A consequence of this is that there is a pre-existing file whose name matches the name of the file being written, but with “~” as the last character, that file will be effectively deleted (it is overwritten with the contents of the current file being written out, and then renamed to the name of that file). This is intended operation and not a bug!

Users of VE are advised to avoid use of the character “~” as the last character of filenames in directories where they may be using VE!

19.0 Known Issues

There are several known issues with the current implementation of VE. Please do not report these as bugs. The issues are:

19.1 Open/Goto File Dialogs and Macros

The present issue of VE does not support the use of the Open or Goto File Dialogs from within a macro. There is no explicit interlock to stop the use of these dialogs, however, and attempts to use them from within a macro will fail in all sorts of horrible and messy ways! CampbellWare is aware of this issue and working on a solution. There always has to be something for a next release!

19.2 Key Mappings

There does not appear to be standard key mappings for the Number Pad keys that VE assigns some commands to. In particular, the Number Pad /, *, - and + keys may produce significantly different key codes depending on the terminal emulator that is running (xterm is only ONE of many terminal emulators that can run under Mac OS X's X11). As a result, these key bindings may not work under all emulators. They are known to work properly under the default xterm provided by Apple. Try them for your favorite emulator – if they do not work, the same capabilities are available via their “ESC-letter” form, or ESC-z, ESC-i, ESC-g and ESC-n respectively.

19.3 Telnet and Key Mappings

When used over a slow Telnet connection, the curses mechanism for detecting extended keyboard commands (timing the interval between the leading ESC and the keyboard code that follows) may break down erratically. Some extended keyboard keys are reported correctly and some are not. The keys in this class include such things as the cursor up/down/left/right keys, the Page Up, Page Down keys, the Home and End keys, and so on. If you observe that these keys are not working during a Telnet session, please use the ESC-letter version of the command. Note that many of the most useful of these keys (cursor up, down, left and right, and page up and down) have been assigned CTL-letter keyboard shortcuts to facilitate usage over Telnet. See Appendix A below for the full list of such shortcuts.

19.4 Auto Indent and On-the-fly Text Wrapping

These two options are mutually exclusive in VE. If Auto Indent is enabled via the Options command, On-the-fly Text Wrapping is automatically disabled. The inverse is true as well: enabling On-the-fly Text Wrapping disables Auto Indent.

19.5 On-the-fly Text Wrapping and Justify/Delete

On-the-fly Text Wrapping can appear to get “stuck” when deleting or rubbing out spaces, if the Right Justify option has been set. This is because in some situations, VE's right justify function may immediately replace the space just deleted, resulting in no change to the file. When this occurs, repeated delete/rubouts of spaces under the cursor will appear to not occur, and the text wrapping function will seem to be stuck. There are two simple solutions to this problem:

- Cursor over to a non space character and carry on
- Disable the Right Justify functionality

19.6 Support for Text Files with Long Lines (>384 chars)

The current edition of VE supports line sizes up to 384 chars. This is an arbitrary size selected by the author as being sufficient for the majority of text files. As has been mentioned earlier in this document, this number can be easily increased by changing the constant MAX_LINE_SIZE in module COMMON.DEF and rebuilding VE from source. Assuming this is not done, however, it is important to understand how VE handles files whose lines exceed 384 characters.

As files with lines longer than MAX_LINE_SIZE are read, the lines longer than this value are broken into multiple separate lines, each one of which is no longer than MAX_LINE_SIZE characters, including the trailing line end characters (LF for Linux formatted files, and CR, LF for DOS formatted files). This allows the files to be viewed, even if they cannot be actively edited. It is imperative that no attempt to be made to write the file back out. VE **will** write the file back out, with the lines broken as described above. This will destroy the integrity of the original file.

20.0 Bug Reports, Donations, Gratuitous Praise

Please send bug reports, donations, gratuitous praise and any other form of communication wished with the author of VE by emailing:

webmaster@campbell-tx.net

CampbellWare will make every effort to respond to all such emails.

21.0 Appendix A – VE Keyboard Commands

21.1 Function Keys

F1	Again command	ESC-a or F1
F2	Block command	ESC-b or F2
F3	Column command	ESC-c or F3
F4	Jump to Line command	ESC-j or F4
F5	Find command	ESC-f or F5
F6	Replace command	ESC-r or F6
F7	Paste command	ESC-p or F7
F8	Edit command	ESC-e or F8
F9	Delete Line command	ESC-l or F9
F10	Delete Right command	ESC-d or F10
F11	Update File command	ESC-u or F11
F12	Quit command	ESC-q or F12

21.2 CTL Keys

CTL-t	Cursor TopOfPage
CTL-g	Cursor MiddleOfPage
CTL-b	Cursor BottomOfPage
CTL-u	Cursor Up
CTL-d	Cursor Down
CTL-l	Cursor Left
CTL-r	Cursor Right
CTL-n	Page Down (Next Page)
CTL-p	Page Up (Previous Page)
CTL-c	CopyLine
CTL-v	Paste Line
CTL-x	Cut (copy & cut) Line
CTL-o	Toggle INSERT/OVERWRITE mode
CTL-z	Undo

21.3 ESC Commands

'I'	INSERT/OVERWRITE toggle	ESC-I or CTL-o or Insert Key
'a'	AGAIN command	ESC-a or F1
'b'	BLOCK command	ESC-b or F2
'c'	COLUMN command	ESC-c or F3
'd'	DELETE LEFT command	ESC-d
'e'	EDIT command	ESC-e or F8
'f'	FIND command	ESC-f or F5
'g'	GOTO FILE command	ESC-g or Keypad '-'
'h'	HELP command	ESC-h
'i'	INFO command	ESC-i or Keypad '*'
'j'	JUMP command	ESC-j or F4
'k'	DELETE RIGHT command	ESC-k or F10
'l'	Delete Line command	ESC-l or F9
'm'	MACRO command	ESC-m
'n'	NEXT FILE command	ESC-n or Keypad '+'
'o'	OPTION command	ESC-o
'p'	GET(paste) command	ESC-p or F7
'q'	QUIT command	ESC-q or F12
'r'	SUBSTITUTE command	ESC-r or F6
's'	STATUS command	ESC-s
't'	TAGS command	ESC-t
'u'	Undo command	ESC-u or CTL-z
'U'	Update command	ESC-u or F11
'v'	VIEW command	ESC-v
'w'	WRAP Paragraph command	ESC-w
'x'	XECUTE command	ESC-x
'z'	Select Line command	ESC-p or Keypd "/"

','	cursor left	ESC-, or <- Key or CTL-l
'.'	cursor right	ESC-. or -> Key or CTL-r
'^'	cursor up	ESC-^ or Key or CTL-u
'$'	cursor down	ESC-$ or Key or CTL-d
';'	Extended Motion command	ESC-; or CTL-e
':'	Go To Display Page Edge	ESC-: or Keypad 5
']'	Word Right command	ESC-] or ESC- ->
'['	Word Left command	ESC-[or ESC- <-
'{'	Paragraph Up	ESC-{
'}'	Paragraph Down	ESC-}
'-'	Scroll Half Screen Up command	ESC--
'+'	Scroll Half Screen Down command	ESC-+
')'	Scroll Right command	ESC-)
'('	Scroll Left command	ESC-(
'#'	RUBOUT command	ESC-# or RUBOUT Key
'D'	DELETE CHAR command	ESC-D or Delete Key
'W'	DELETE WORD command	ESC-W or ESC-Delete
'<'	Start of line	ESC-< or HOME Key
'>'	End of line	ESC-> or END Key
'B'	Page Up	ESC-B or PgUp Key or CTL-p
'\'	Top of File	ESC-(or ESC-PgUp
'F'	Page Down	ESC-F or PgDn Key or CTL-n
'~'	Bottom of File	ESC-) or ESC-PgDn
'C'	Copy Line Del Buffer	ESC-C or CTL-c
'X'	Cut Line Del Buffer	ESC-X or CTL-x
'V'	Paste Line Del Buffer	ESC-V or CTL-v
'H'	Cursor TopOfPage	ESC-H or CTL-t
'M'	Cursor MiddleOfPage	ESC-M or CTL-g
'L'	Cursor BottomOfPage	ESC-L or CTL-b
'`'	JumpToTag command	ESC-`
'A'	About VE	ESC-A

22.0 Appendix B – Building VE From Source

22.1 Where To Get the Source

The VE 3.5 distribution tarball contains a PPC G5 executable, a README file, this manual, and a complete source tarball. To build VE from source, unpack the source tarball to any convenient place. This will create a top level directory called “ve”, a subtending directory called ve3.5x-src (where “x” is the specific subversion letter of the release you have downloaded), and a set of directories underneath that. You are now ready to build VE.

22.2 Prerequisites

VE makes no assumptions about software loaded on your machine other than that the GNU C++ development tools (C compiler, linker, etc.) are available, and that files stdio.h, stdlib.h and curses.h are available in the current include path. The GNU tools and support files are provided on your Apple OS X install disks, and must be installed if you wish to build VE from source.

22.3 Building VE

To build VE from source, perform the following steps:

- cd to the “prog” directory under the ve3.5x-src directory. You **MUST** be in the prog directory to build VE successfully. Most VE source files contain #include statements that use filenames that are relative to the “prog” directory.
- If you have built VE previously, enter “make clean” to remove all previous object files.
- Enter “make”.
- Copy the newly created ve executable (filename “ve”) to anywhere convenient on your path. Alternately, you may su to root and enter “make install”.
- Congratulations! You have built and installed VE from source.

22.4 Some Important Files and Constants

There should be very little need to build VE from source, unless you wish to rebuild it for an Intel Mac or are increasing the supported line size or terminal size. The following files will be of interest for such tasks (filenames are relative to the “prog” directory):

- File “Makefile” - this file contains the instructions which allow the make utility to build VE. If you are changing to a non PPC architecture, you will need to edit Makefile to specify compiler and or other tool options needed to target the build. This User Manual does not offer instruction on building for non PPC architectures.
- File ../def/common.def – this file defines common contants for all of VE. Almost all VE source files #include ../def/common.def and so it is essential to do a “make clean” after changing anything in common.def before rebuilding. Odd behaviour may result if this warning is not heeded!

  A key constant defined here is MAX_LINE_SIZE. This defines the length in characters of the maximum size line that VE will support. In the distribution version, this is set to 384 characters. Increase this to any size needed. Note that several internal data structures are sized using this setting, and so increasing it will increase the amount of data space required by VE. Note also that constant VIRTUAL_SCREEN_SIZE in module **termif.c** must **always** be larger than MAX_LINE_SIZE. See the notes on termif.c below.

  Another important constant defined here is MAX_OPEN_FILES. This defines the maximum number of filest that VE may have open at any one time. This is set to 256 in the current version. This may be **reduced** at will, but it must **not** be increased beyond 256.

- File termif.c – this file defines the interface between VE and the physical terminal/xterm. In the Linux port, this is implemented via ncurses calls.

  A key constant defined here is VIRTUAL_SCREEN_SIZE. This defines the maximum left/right scrolling range for VE. In the current version, this is set to 512 columns. It is **essential** that VIRTUAL_SCREEN_SIZE be larger than MAX_LINE_SIZE. It must also be larger than the width of the physical display. If you have increased MAX_LINE_SIZE to a value greater than VIRTUAL_SCREEN_SIZE, you must increase the value of VIRTUAL_SCREEN_SIZE such that it becomes larger than the new value of MAX_LINE_SIZE.

23.0 Appendix C – A Brief VE Release History

23.1 Release History

The following is a brief VE release history:

- v1.x – 1983. Initial version of VE, released on Qunix (a unix lookalike) for the NABU 1600 small business computer. Limited basic text editing functionality.
- v2.x – 1994. Release of VE for MS-DOS and the x86 PC. Port of v1.x from Qunix to MS-DOS, addition of word and paragraph support and both on-the-fly and after-the-fact text wrapping.
- v3.0 – 2004 – Initial release of VE for Linux and the x86 architecture. SIGNIFICANT number of new features added, including Help and About commands, support for Tags, paging, configurable screen colors, etc. The initial version of the Goto File Dialog debut'd in this release. Numerous bugs fixes were also included.
- v3.1 – 2004 - Added Global Undo
- v3.2 – 2004 - Added long line support via Scroll Left/Right
- v3.3 – 2005 - Added Column/Rectangle support
- v3.4 – 2005 - Added Mouse support
- v3.5 – 2005 - Added Open File Dialog, rewrite of Goto File Dialog, mouse support for both of these dialogs, global line editing for responses to all VE prompts.
- v3.5 – 2006 – Ported to Mac OS X and released.

23.2 New in Release 3.5

The following content is new in release 3.5 (Jul 2005):

- Open File Dialog, an interactive file selection and file system browsing
- Goto File Dialog re-implementation of Goto File Dialog to simplify, share code base with Open File Dialog
- Mouse Support for Open/Goto File Dialogs
- New startup option "-b", which starts VE in directory browsing mode, allowing interactive selection of the file to initially edit
- Global Line Editing for all responses to VE prompts
- [Quit, All] Enhancement to only prompt if one or more files have changed
- Replace Command use of second line to present "with:" prompt
- Significant number of text field overflow vulnerabilities eliminated
- Miscellaneous bug fixes

End of Document