NAME

ved − visual editor (screen-oriented)

SYNOPSIS

ved [*options*] [*filename* . . .]

DESCRIPTION

Ved (visual editor) is a small and very fast screen-oriented editor that implements a user interface somewhere between **vi** and **emacs**. It is powerful and easy to learn, it has no limitations on line length and the type of characters that may appear in a file. It has almost no limitations on file size (see **NOTES** section below).

To use **ved** to edit a file for the first time, call **ved** and specify the name of the file you want to edit. If you edit a file for the second time, **ved** will place you at the same cursor position that was used the last time when editing this file. If you do not specify a file name, **ved** will assume that you want to edit the last file you edited in your current directory. The cursor position in this case again will be the same as with the last editing session for this file.

OPTIONS

−v, **−vedhelp**
An editing session is started with the on-line help. This is done in read-only mode.

−r, **−readonly**
In this mode, you may edit the content of the edit buffer but you are not allowed to write back the modified buffer by using the quit command of the editor. If you use this option more than once, the readonly value is incremented. If **−readonly** is used twice or more, you are not allowed to write the file back even in quit force mode.

b=#, **buffers=**#
By default **ved** uses 10 in-core buffers, each in-core buffer takes 8 kB (the actual values that are compiled into **ved** may be retrieved by calling **ved −help**). Files that are bigger than the total amount of in-core buffer space are edited by using the swap file of **ved**. If you specify a number that is higher than the default, you will need more space in memory but the speed will slightly increase.

−n, **−nobak**
Don't create *filename.bak* when writing back the file using the quit command of the editor. Note that if you use this option you will still break hard links. If you like to write into the original file, you should better use the 'W' command when you quit **ved**.

−e, **−edtmp**
Don't write to the file **.vedtmp**.*uid***.** (**ved** normally stores the names and cursor positions of files in **.vedtmp**.*uid*)

−raw8 Do not expand 8-bit characters. Note that **ved** will by default not expand 8-bit characters that seem to be printable. If this option is ised, the expansion of 8-bit characters is turned off unconditionally.

−dos Map new-line to carriage-return new-line and hide carriage-return characters that are directly followed by a new-line. If you type in a new line this results in inserting a carriage-return new-line pair. But as **ved** is acting binary transparent, reading in files that do not follow DOS newline conventions will not cause the content of the file to be automatically converted to use DOS new-lines.

s=*search string*
Start editing session with search to *searchstr*.

−tag First filename is really a ctag.

wrapmargin=#

Set default wrap margin to #. The wrap margin is counted from the end of a terminal line.

−R, **−Recover**

Recover Session from specified filename. **ved** generates a file *vedP.xxxx* in the */tmp* directory in which all action of the user is stored during the session. This file must be used as the only argument to **ved** when running a recover session.

THE STATUS LINE

The first (top) line of the screen is reserved for **ved** status information. The status line is divided into five fields. The fields are (from left to right):

Info/Input

The first field is used to display informational messages and to fetch additional input that is required for some commands. Each command that needs confirmation requests input from this field. The size of this field defaults to 1/4th of the length of a line. It may use more if needed.

Number

The second field contains the current multiplier for the next command. The current multiplier is reset to 1 after each command, it may be set to a different value by the user (See **NUMBERS** section below). This multiplier will be referred to as *number* in the following text. The size of this field is fixed to 1/8th of the length of a line.

Takebuf

The third field contains is the name of the actual take buffer. The name of this buffer defaults to **default** on startup. It may be changed by the user. The size of this field is fixed to 1/4th of the length of a line.

Filename

The fourth field usually displays the current filename. The size of this field defaults to 1/4th of the length of a line. It may use up more if needed and no error text is displayed. It may be hidden for some time if a long error message is displayed.

Error text

The fifth field is used to display all error messages and warnings. The size of this field defaults to 1/4th of the length of a line. If needed, it may use the space of the filename field too.

BASIC COMMAND FORMAT

Ved does not distinguish between *command mode* and *insert mode*. All non control characters are treated as input (which should be inserted into the file buffer) unless they are prefixed by some commands that need arguments.

All basic **ved** commands are control characters or escape sequences. To enter a control character, use the **control** key like the **shift** key together with a character key; to enter an escape sequence, press the <**ESC**> key followed by another key.

In the following description, ˆ represents the control key and <**ESC**> the escape key (which is identical to pressing **ˆ[**). <**ALT**> has a similar usage as <**ESC**> but different meaning in **ved** and represents the combination **ˆ]**. The combination ˆ*char* is a single keystroke, while <**ESC**>*char* or <**ALT**>*char* are two keystrokes. If the command takes an argument, the argument is terminated with a carriage return <**CR**>.

A command may be modified in four ways:

Escape commands

Typing the <**ESC**> key (or **ˆ[**) before the command will modify it to an alternate form, usually a related function.

Alt commands

Typing the **ALT** key (**ˆ]**) before the command, inserts a mark (with a few exceptions) in the text usually at the current cursor position. You can precede any command that moves the cursor with the **ALT** key and **ved** sets the mark before moving the cursor.

Extended commands

Typing a **ˆX** before another command enters the extended mode. The extended mode just enters another level of commands.

Multiplied execution

Numbers that are entered by typing <**ESC**>#<**CR**> (# represents a positive number) will set the multiplier to that number and display it in the status line of **ved**. If a number greater than **one** is entered before a command, that command is usually repeated *number* times. Exceptions are some cursor movement commands where different functions will be executed *number* times instead of repeating the visible basic functionality of the command.

CURSOR MOVEMENT

Like every editor, **ved** has a typing point, called **cursor**, which is indicated by the (in most cases) flashing bar or box (depending on your terminal/window type) on your screen. The cursor indicates the point within the file buffer where the typed characters are going to. It is your current place in the file buffer. The commands that move the cursor within the file buffer are explained in this section.

Cursor movement command overview:

Key Label		**CTRL**	**ESC CTRL**	**ALT** set mark first	**ALT ESC** set mark first
|←	**A**	start of line	start of para	start of line	start of para
F2	**B**	go to beginning	go to bottom	go to beginning	go to bottom
→|	**E**	end of line	end of para	end of line	end of para
→	**F**	forward char	forward word	forward char	forward word
←	**H**	back char	back word	back char	back word
↓	**N**	down line	down para	down line	down para
↑	**P**	up line	up para	up line	up para

Cursor movement commands in detail:

ˆA moves the cursor to the beginning of the current line. If the cursor is already at the beginning, it will not move.

If *number* is greater than one, **ˆA** moves the cursor that number of characters from the beginning of the current line.

<**ESC**>**ˆA**

moves the cursor to the beginning of the current paragraph. If the cursor is already at the beginning, it will not move.

If *number* is greater than one, <**ESC**>**ˆA** moves the cursor that number of words from the beginning of the current paragraph.

<**ALT**>**ˆA**

inserts the mark at the current cursor position before moving the cursor to the beginning of the current line. If the cursor is already at the beginning, it will not move, but **ved** will still inserts the mark.

If *number* is greater than one, <**ALT**>**ˆA** inserts the mark, then moves the cursor that number of characters from the beginning of the current line.

<**ALT**><**ESC**>**ˆA**

inserts the mark at the current cursor position before moving the cursor to the beginning of the current paragraph. If the cursor is already at the beginning, it will not move, but **ved** will still inserts the mark.

If *number* is greater than one, <**ALT**><**ESC**>**ˆA** inserts the mark, then moves the cursor that number of words from the beginning of the current paragraph.

ˆB
moves the cursor to the beginning of the file. If the cursor is already at the beginning of the file, it will not move.

If *number* is greater than one, **ˆB** moves the cursor that number of lines from the beginning of the file.

<**ESC**>**ˆB**
moves the cursor to the end of the file. If the cursor is already at the end of the file, it will not move.

If *number* is greater than one, <**ESC**>**ˆB** moves the cursor that number of lines from the end of the file.

<**ALT**>**ˆB**
inserts the mark at the current cursor position before moving the cursor to the beginning of the file. If the cursor is already at the beginning of the file, it will not move, but **ved** still inserts the mark.

If *number* is greater than one, <**ALT**>**ˆB** inserts the mark, then moves the cursor that number of lines from the beginning of the file.

<**ALT**><**ESC**>**ˆB**
inserts the mark at the current cursor position before moving the cursor to the end of the file. If the cursor is already at the end of the file, it will not move, but **ved** still inserts the mark.

If *number* is greater than one, <**ALT**><**ESC**>**ˆB** inserts the mark, then moves the cursor that number of lines from the end of the file.

ˆE
moves the cursor to the end of the current line. If the cursor is already at the end of the line, it will not move. The last character on the line is the invisible ASCII newline character NL; when the cursor is at the end of the line, it is not on a visible character.

If *number* is greater than one, **ˆE** moves the cursor that number of characters from the end of the current line.

<**ESC**>**ˆE**
moves the cursor to the end of the current paragraph. If the cursor is already at the end, it will not move.

If *number* is greater than one, <**ESC**>**ˆE** moves the cursor that number of words from the end of the current paragraph.

<**ALT**>**ˆE**
inserts the mark at the current cursor position before moving the cursor to the end of the current line. If the cursor is already at the end of the line, it will not move, but **ved** still inserts the mark.

If *number* is greater than one, <**ALT**>**ˆE** inserts the mark, then moves the cursor that number of characters from the end of the current line.

<**ALT**><**ESC**>**ˆE**
inserts the mark at the current cursor position before moving the cursor to the end of the current paragraph. If the cursor is already at the end of the paragraph, it will not move, but **ved** still inserts the mark.

If *number* is greater than one, <**ALT**><**ESC**>**ˆE** inserts the mark, then moves the cursor that number of words from the end of the current paragraph.

ˆF
moves the cursor one character position to the right.

If *number* is greater than one, **ˆF** moves the cursor that number of characters to the right.

Depending on the cursor position, with or without *number* being greater than one, **ˆF** could move the cursor down to the next line. If the cursor is at the end of the line, the cursor will move to the first position of the next line.

<**ESC**>**ˆF**
moves the cursor forward to the first letter of the next word.

If *number* is greater than one, <**ESC**>**ˆF** moves the cursor forward that number of words.

Depending on the current cursor position and the value of *number*, <**ESC**>**ˆF** might move the cursor down one or more lines.

<**ALT**>**ˆF**
inserts the mark at the current cursor position before moving the cursor one character to the right.

If *number* is greater than one, <**ALT**>**ˆF** inserts the mark, then moves the cursor that number of characters to the right.

<**ALT**><**ESC**>**ˆF**
inserts the mark at the current cursor position before moving the cursor to the first letter of the next word.

If *number* is greater than one, <**ALT**><**ESC**>**ˆF** inserts the mark, then moves the cursor that number of words to the right.

ˆH moves the cursor one character position to the left.

If *number* is greater than one, **ˆH** moves the cursor that number of characters to the left.

Depending on the cursor position, **ˆH**, with or without *number* being greater than one, could move the cursor up to the previous line. If the cursor is at the beginning of the line when you press **ˆH**, the cursor will move to the last position of the previous line.

<**ESC**>**ˆH**
moves the cursor to the first letter of the word just prior to its current position.

If *number* is greater than one, <**ESC**>**ˆH** moves the cursor that number of words backwards. Depending on the current cursor position and the value of *number*, the cursor may move up one or more lines.

<**ALT**>**ˆH**
inserts the mark at the current cursor position before moving the cursor one character position to the left.

If *number* is greater than one, <**ALT**>**ˆH** inserts the mark, then moves the cursor that number of characters to the left.

<**ALT**><**ESC**>**ˆH**
inserts the mark at the current cursor position before moving the cursor to the first letter of the previous word.

If *number* is greater than one, <**ALT**><**ESC**>**ˆH** inserts the mark, then moves the cursor that number of words to the left.

ˆN moves the cursor down one line while maintaining its horizontal position. If the cursor is on the last character of the last line when you press **ˆN**, the cursor will not move and you will hear a beep.

If *number* is greater than one, **ˆN** moves the cursor down that number of lines. If *number* is greater than the number lines remaining in the file, the cursor will stop moving at the end of the file.

<**ESC**>**ˆN**

moves the cursor to the beginning of the next paragraph.

If *number* is greater than one, <**ESC**>**ˆN** moves the cursor down that number of paragraphs.

<**ALT**>**ˆN**

inserts the mark at the current cursor position before moving the cursor down one line. Note that the cursor maintains its horizontal position on the line; i.e., if the cursor was on the fifth character of the line when you pressed <**ALT**>**ˆN**, it will move to the fifth character of the line below.

If *number* is greater than one, <**ALT**>**ˆN** inserts the mark, then moves the cursor down that number of lines.

<**ALT**><**ESC**>**ˆN**

inserts the mark at the current cursor position before moving the cursor to the beginning of the next paragraph.

If *number* is greater than one, <**ALT**><**ESC**>**ˆN** first inserts the mark, then moves the cursor down that number of paragraphs.

ˆP moves the cursor up one line while maintaining its horizontal position. If the cursor is on the first line of the file when you press **ˆP**, the cursor will not move and you will hear a beep.

If *number* is greater than one, **ˆP** moves the cursor up that number of lines. If *number* is greater than the number of lines between the cursor and the beginning of the file, the cursor will stop moving at the beginning of the file.

<**ESC**>**ˆP**

moves the cursor to the beginning of the previous paragraph.

If *number* is greater than one, <**ESC**>**ˆP** moves the cursor that number of paragraphs prior to the cursor's position.

<**ALT**>**ˆP**

inserts the mark at the current cursor position before moving the cursor up one line. Note that the cursor maintains its horizontal position on the line; i.e., if the cursor was on the fifth character of the line when you pressed <**ALT**>**ˆP**, it will move to the fifth character of the line above.

If *number* is greater than one, <**ALT**>**ˆP** inserts the mark, then moves the cursor up that number of lines.

<**ALT**><**ESC**>**ˆP**

inserts the mark at the current cursor position before moving the cursor to the beginning of the previous paragraph.

If *number* is greater than one, <**ALT**><**ESC**>**ˆP** first inserts the mark, then moves the cursor up that number of paragraphs prior to its current position.

ADDITION/DELETION

All additions are always made at the place where the visible cursor is.

All deletions are saved in two buffers: the **forward delete buffer** and the **backwards delete buffer**. The characters in the backward delete buffer are saved in reverse order to allow to reinsert them in the correct order. As long as you don't move the cursor and start deleting new text, the content of both, the forward and the backwards delete buffer may be reinserted with the **undo delete** command. This allows you to move text by deleting it and reinserting it after you moved the cursor to another position in the file

Addition/Deletion command overview:

Key Label		**CTRL**	**ESC CTRL**	**ALT** set mark first	**ALT ESC** set mark first
F3	**D**	delete char	delete words	˜	˜
TAB	**I**	tab	˜	tab	˜
NL	**J**	insert new line	˜	jump to mark	switch mark and cursor
F4	**K**	kill line	kill para	kill selection	˜
CR	**M**	insert new line	change modes	sets mark	˜
F5	**O**	open line	˜	open line	˜
	_	undo delete	˜	clear mark	˜
DEL	**DEL**	erase char	erase word	˜	˜

Addition/Deletion commands in detail:

ˆD erases the character at the cursor position.

If *number* is greater than one, **ˆD** erases that number of characters to the right of the cursor.

<**ESC**>**ˆD**
erases the next word. A word is defined from the current cursor position to the next blank or carriage return.

If *number* is greater than one, <**ESC**>**ˆD** erases that number of words to the right of the cursor.

ˆI inserts a **TAB** character.

If *number* is greater than one, **ˆI** inserts that number of **TAB** characters.

<**ALT**>**ˆI**
inserts the mark at the current cursor position before inserting a **TAB** character.

If *number* is greater than one, <**ALT**>**ˆI** inserts the mark, then inserts that number of **TAB** characters.

ˆJ inserts a **new line** at the current cursor position and moves the cursor to the beginning of the next line.

If *number* is greater than one, **ˆJ** inserts that number of new lines.

ˆK erases the current line. If the cursor is located at the beginning of the line, the entire line is erased, including the carriage return at the end of the line. If the cursor is located between the beginning and end of the line, everything right of the cursor is erased, including the carriage return at the end of the line. You can use **ˆK** to erase blank lines.

If *number* is greater than one, **ˆK** erases that number of lines from the current cursor position.

<**ESC**>**ˆK**
erases from the current cursor position to the end of the paragraph. The end of a paragraph is indicated by two successive carriage returns--one at the end of the last line and one on the blank line that follows the paragraph.

If *number* is greater than one, <**ESC**>**ˆK** erases that number of paragraphs.

<**ALT**>**ˆK**
erases text between the mark and the current cursor position. For instance, if the mark is in the middle of the file and the cursor is at the end, the last half of the file will be erased.

ˆM inserts a **new line** at the current cursor position and moves the cursor to the beginning of the next line.

If *number* is greater than one, **ˆM** inserts that number of new lines.

ˆO inserts a **new line** at the current cursor position, but does not move the cursor. Any text to the right of the cursor moves down to start a new line.

If *number* is greater than one, **ˆO** inserts that number of new lines without moving the cursor.

<ALT>ˆO
inserts a new line without moving the cursor. It inserts the mark at the beginning of the next (new) line.

If *number* is greater than one, **<ALT>ˆO** inserts that number of new lines without moving the cursor. It inserts the mark after the last new line.

ˆDELETE
erases the character to the left of the cursor. This acts the same as using the DELETE key alone.

If *number* is greater than one, **ˆDELETE** erases that number of characters to the left of the cursor.

<ESC>ˆDELETE
erases the word to the left of the cursor. This acts the same as using ˆDELETE.

If *number* is greater than one, **<ESC>ˆDELETE** erases that number of words to the left of the cursor.

ˆ_ reinserts the last deletion. The content of the **forward** and **backward** delete buffer is cleared after reinserting the deleted text.

<ALT>ˆ_
erases the mark.

CUT/PASTE

These commands deal with the ’take’ buffers. You can have up to 32 take buffers, each named as you choose when you use it for the first time. The name appears on the status (top) line of the terminal screen following **\:**. The default take buffer is "default".

Cut/Paste command overview:

Key Label		**CTRL**	**ESC CTRL**	**ALT** set mark first	**ALT ESC** set mark first
	T	save line into paste buf	save para into paste buf	save selection into paste buf	˜ into paste buf
F7	**V**	paste	paste and clear paste buf	replace select with paste buf	replace select clear paste buf
	Y	save char into paste buf	save word into paste buf	˜	˜
F8	\	chng paste buf	clr paste buf	˜	˜

Cut/Paste commands in detail:

ˆT removes a line from the file and puts it in the current take buffer. The line is defined from the current cursor position to the next carriage return.

If *number* is greater than one, **ˆT** removes that number of lines from the file and puts them in the current take buffer.

<ESC>ˆT

removes a paragraph from the file and puts it in the current take buffer. A paragraph is defined from the current cursor position to two successive carriage returns--one at the end of the last line and one on the blank line that follows the paragraph.

If *number* is greater than one, **<ESC>ˆT** removes that number of paragraphs from the file and puts them in the current take buffer.

<ALT>ˆT
removes the selection (what lies between the mark and the cursor) and puts it in the current take buffer.

ˆV pastes the contents of the current take buffer into the file at the current cursor position. The take buffer will still contain a copy of the text.

<ESC>ˆV
pastes and clears the contents of the current take buffer into the file at the current cursor position. The take buffer will be empty.

<ALT>ˆV
replaces the selection (what lies between the mark and the cursor) with whatever is in the current take buffer. The old content of the selection is lost. The take buffer will still contain a copy of the text.

<ALT><ESC>ˆV
replaces the selection (what lies between the mark and the cursor) with whatever is in the current take buffer and clears the take buffer. The old content of the selection is lost. The take buffer will be empty.

ˆY removes a character from the file and puts it in the current take buffer. The character at the current cursor position is the one taken.

If *number* is greater than one, **ˆY** takes that number of characters and puts them in the current buffer.

<ESC>ˆY
removes a word from the file and puts it in the current take buffer. A word consists of any characters from the current cursor position up to the next blank or carriage return.

If *number* is greater than one, <ESC>ˆY takes that number of words and puts them in the current buffer.

ˆ\ changes the current take buffer.

On the information line you will see the **Take Buffer:** prompt. Type a take buffer name and press the RETURN key.

If you type a new name, a new take buffer is created. If you type a name you have already used during the current editing session, you can reuse that take buffer. The take buffer name is always displayed on the information line.

<ESC>ˆ
clears any text out of the current take buffer. On the information line you will see the Buffer cleared message.

SEARCH

Search command overview:

Key Label		**CTRL**	**ESC CTRL**	**ALT** set mark first	**ALT ESC** set mark first
F9	**R**	forward search	reverse search	forward search	reverse search

F10	**Z**	re-search	reverse re-srch	re-srch	reverse re-srch

Search commands in detail:

ˆR searches forward in a file for the specified text string.

On the information line you will see the **+Search:** prompt. Type the string you want to find. If the string exists between the cursor's current position and the end of the file, **ved** moves the cursor to that location. If **ved** can not find the specified string, the cursor does not move and you will hear a beep.

If *number* is greater than one, **ˆR** finds that numbered occurrence of the string.

<**ESC**>**ˆR**

searches backwards in a file for the specified text string.

On the information line you will see the −**Search:** prompt. Type the string you want to find. If the string exists between the cursor's current position and the beginning of the file, **ved** moves the cursor to that location. If **ved** can not find the string, the cursor does not move and you will hear a beep.

If *number* is greater than one, <**ESC**>**ˆR** finds that numbered occurrence of the string.

<**ALT**>**ˆR**

searches forward in a file for the specified text string.

On the information line you will see the **+Search:** prompt. Type the string you want to find. When (and if) found, the mark is inserted at the beginning of the string and the cursor moves to the end of the string; the string is then considered selected. If **ved** can not find the string, the cursor does not move, no mark is inserted, and you will hear a beep.

If *number* is greater than one, <**ALT**>**ˆR** finds that numbered occurrence of the string.

<**ALT**><**ESC**>**ˆR**

searches backwards in a file for the specified text string.

On the information line you will see the −**Search:** prompt. Type the string you want to find. When found, the mark is inserted at the beginning of the search string and the cursor moves to the end of the string; the string is then considered selected. If **ved** can not find the string, the cursor does not move, no mark is inserted, and you will hear a beep.

If *number* is greater than one, <**ALT**><**ESC**>**ˆR** finds that numbered occurrence of the string.

ˆZ re-searches for the last text string you asked to for with the **ˆR**, <**ESC**>**ˆR**, <**ALT**>**ˆR**, or <**ALT**><**ESC**>**ˆR** command. If you already found one occurrence of the text string, **ˆZ** will find the next occurrence. If **ved** can not find another occurrence, the cursor does not move and you will hear a beep.

If *number* is greater than one, instance, **ˆZ** finds that numbered occurrence of the string.

<**ESC**>**ˆZ**

reverses direction and re-searches for the last text string you asked for with the **ˆR**, <**ESC**>**ˆR**, <**ALT**>**ˆR**, or <**ALT**><**ESC**>**ˆR** command. If you already found one occurrence of the text string, <**ESC**>**ˆZ** will find the next, or previous, occurrence, depending on which your direction. For instance, if the original command was **ˆR**, <**ESC**>**ˆZ** will find the previous occurrence. If **ved** can not find another occurrence, the cursor does not move and you will hear a beep.

If *number* is greater than one, instance, <**ESC**>**ˆZ** finds that numbered occurrence of the string.

<**ALT**>**ˆZ**

re-searches for the last text string you asked for with the **ˆR**, <**ESC**>**ˆR**, <**ALT**>**ˆR**, or <**ALT**><**ESC**>**ˆR** command. If you already found one occurrence of the text string, <**ALT**>**ˆZ** will find the next occurrence, set the mark at the beginning of it, and move the cursor to the end of it. This occurrence of the string is then considered selected. If **ved** can not find another occurrence, the cursor does not move, no mark is inserted, and you will hear a beep.

If *number* is greater than one, instance, <**ALT**>**ˆZ** finds that numbered occurrence of the search string, with the numbering beginning from the current cursor position. For instance, if *number* is 4, the next three occurrences are skipped and the mark is inserted at the beginning of the fourth occurrence of the string. This happens regardless of how many occurrences of the string have already been located, unless there are not that many occurrences of the string in the file.

<**ALT**><**ESC**>**ˆZ**

reverses direction and re-searches for the last text string you asked for with the **ˆR**, <**ESC**>**ˆR**, <**ALT**>**ˆR**, or <**ALT**><**ESC**>**ˆR** command. If you already found one occurrence of the text string, <**ALT**><**ESC**>**ˆZ** will find the next, or previous, occurrence, depending on your direction. For instance, if the original command was **ˆR**, <**ALT**><**ESC**>**ˆZ** will find the previous occurrence, set the mark at the beginning of it, and move the cursor to the end of it. This occurrence of the search string is then considered selected.

If *number* is greater than one, instance, <**ALT**><**ESC**>**ˆZ** finds that numbered occurrence of the search string, with the numbering beginning from the current cursor position.

EXITING/PUTTING

ˆC The QUIT command ends the editing session.

On the information line you will see the **QUITTING. PUT EDITS? (Y/W/N/F/!)** prompt.

Type **Y** if you want to save the changes made during this editing session by creating a backup file and creating a new file that holds the modified content. The backup process is done the following way:

The current backup file (if there is one) (called *filename*.bak) is deleted, the current file is renamed to *filename*.bak, and a version of the file containing the latest changes becomes *filename*. If there is no *filename*.bak, *filename* becomes *filename*.bak.

Type **W** if you want to save the changes made during this editing session by directly writing the new content into the old file.

Type **F** if you want to **forcibly** save the changes made during this editing session. This command works similar to the **Y** command, but **ved** will ignore the state of the **−readonly** flag and the access rights of the file. The latter will allow you to write into files without write permissions if you are super user. For execptions on this rule, see description of the **−readonly flag.**

Type **!** if you want to **forcibly** save the changes made during this editing session by directly writing the new content into the old file.

Type **N** if you want to end the session but don't want to save the changes.

If you don't want to end the session, type anything except **Y**, **W**, **F**, **!** or **N**.

If the file has been locked before by another edit session or if the file has been modified more recently than the copy that has been used for the current session, you will not be allowed to save the file. Use the force commands (**F** or **!**) or write back using different file name.

<**ESC**>**ˆC**

The BACKUP command backs up the current file.

On the information line you will see the **BACKUP?(Y/W/N/F/!)** prompt.

If you enter one of the expected answers (**Y**, **W**, **!** or **N**) The same actions as with the QUIT command are performed except that the controls stays inside **ved** after the backup command has completed.

ˆXˆH calls on-line help.

On the information line you will see the **HELP(Y?)** prompt. To continue editing (and abort this command), type **N**. To see the help file, type **Y**.

ˆXˆZ suspends the current editing session.

On the information line you will see the **SUSPENDING. PUT EDITS? (Y/W/N/F/!)** prompt.

Type **Y** if you want the file to be backuped and updated before suspending **ved**. See the **QUIT** command for a description of the backup process.

Type **W** if you want the the file to be updated in place before suspending **ved**.

Type **F** if you want file to be **forcibly** updated before suspending **ved**. This command works similar to the **Y** command, but **ved** will ignore the state of the −**readonly** flag and the access rights of the file. The latter will allow you to write into files without write permissions if you are super user. For execptions on this rule, see description of the −**readonly flag.**

Type **!** if you want file to be **forcibly** updated in place before suspending **ved**.

Type **N** if you want to suspend **ved** but don't want to save the changes before.

If you don't want to suspend **ved**, type anything except **Y**, **W**, **F**, **!** or **N**.

If the file has been locked before by another edit session or if the file has been modified more recently than the copy that has been used for the current session, you will not be allowed to save the file. Use the force commands (**F** or **!**) or write back using different file name.

When you are ready to continue editing, use the **fg** or **resume** command of your shell. You will return to the editing session. The cursor will be in the same place it was when you suspended the session, regardless of whether or not you saved your edits.

FILE COMMANDS

ˆG lets you insert another file at the current cursor position.

On the information line you will see the **Get from:** prompt. Type a file name and press the RETURN key.

You can insert files from the current working directory or from other directories (as long as you use a valid path name). The path name is subject of filename globbing by your shell (e.g. wild cards are working as long as they expand to exactly one filename).

<ESC>ˆG

exits the current file and enters another file.

On the information line you will see the **Change to:** prompt. Type the name of the file you wish to edit and press the RETURN key.

If the old file was modified, you will see the **CHANGING TO:** *filename*. **PUT EDITS?(Y/W/N/F/!)** prompt; if you want to save and backup the edits in the current file.

Type **Y** if you want the file to be backuped and updated before changing the file. See the **QUIT** command for a description of the backup process.

Type **W** if you want the the file to be updated in place before changing the file.

Type **F** if you want file to be **forcibly** updated before changing the file. This command works similar to the **Y** command, but **ved** will ignore the state of the −**readonly** flag and the access rights of the file. The latter will allow you to write into files without write permissions if you are super user. For execptions on this rule, see description of the −**readonly flag.**

Type **!** if you want file to be **forcibly** updated in place before changing the file.

Type **N** if you want to change the file but don't want to save the changes before.

If you don’t want to change the edited file, type anything except **Y**, **W**, **F**, **!** or **N**.

If the file has been locked before by another edit session or if the file has been modified more recently than the copy that has been used for the current session, you will not be allowed to save the file. Use the force commands (**F** or **!**) or write back using different file name.

ˆW writes a copy of the file you are editing to the file you specify.

On the information line you will see the **Write to:** prompt. Type a file name and press the RETURN key. If the file exists, its contents are overwritten. If the file does not exist, it is created.

<**ESC**>**ˆW**
writes the contents of the current take buffer to the file you specify.

On the information line you will see the \ **to file:** prompt. Type a file name and press the RETURN key. If the file exists, its contents are overwritten. If the file does not exist, it is created. The take buffer contents will remain intact.

<**ALT**>**ˆW**
writes a copy of the selection (what lies between the mark and the cursor) to the file you specify.

On the information line you will see the **Sel to:** prompt. Type a file name and press the RETURN key. If the file exists, its contents are overwritten. If the file does not exist, it is created.

NUMBERS

<**ESC**>#<**CR**>
To enter a multiplier that only works for the next command, type <**ESC**>, enter a number and then press the RETURN key. If you entered a number by mistake, simply enter <**ESC**>**1**<**CR**> to reset the number to it’s default value.

ˆU multiplies *number* by the multiplier. The default multiplier is 4. Each time you press **ˆU** *number* changes to the next multiple of 4 (16, 16, 64, and so on).

<**ESC**>**ˆU**
changes the multiplier.

The default multiplier is 4. If you want to multiply *number* by a different value, use <**ESC**>**ˆU**. On the information line you will see the **Mult =** prompt. Type a number and press the RETURN key.

MISCELLANEOUS

<**ALT**>**ˆG**
lets you edit your editing macro file called **.vedmac** (which is located in your *$HOME* directory). It is the same as if you used **ˆXˆX** to execute a new **ved** and answered **N** to the *put* question. When you are through editing **.vedmac**, press ˆC to continue editing the file you were in when you issued the <**ALT**>**ˆG** command; you will be asked whether or not you want to put the edits from **.vedmac** before you return to the original file.

<**ALT**>**ˆJ**
moves the cursor to the mark.

The current cursor position determines whether this is a forward or backward movement. For instance, if the mark is in the middle of the file and the cursor is at the beginning, its a forward movement; if the cursor is at the end, its a backward movement.

<**ALT**><**ESC**>**ˆJ**
switches the cursor and the mark.

ˆL redisplays the current screenful of text. This is useful after a message has been sent to your terminal which may have overwritten some of the text.

<ESC>ˆL
adjusts the text displayed on the terminal screen so that the line the cursor is on becomes the optimum line on the screen. This usually is the center line of the display but it may be changed using the <**ESC**>**:set optline** # <**CR**> command. This may involve displaying different text, depending on where the cursor is when you use this command.

<ALT>ˆL
Adjusts the current Line of the Cursor to be the first Line on Screen.

<ESC>ˆM
changes modes.

On the information line you will see the **Mode?(ORV8)** prompt. Type

D to enable or disable **DOS** mode.

V for visible mode.

O for overstrike mode.

8 to enable or disable **raw8** mode.

R to reset back to default mode.

<**CR**> to list the current mode settings.

If you type anything else, you will abort the command.

<ALT>ˆM
sets a mark at the current cursor position.

<ESC>n
displays the next Page of file.

<ESC>p
displays the previous Page of file.

<ALT><ESC>n
Sets a Mark, then displays the next Page of file.

<ALT><ESC>p
Sets a Mark, then displays the previous Page of file.

ˆQ unfreezes the screen which was frozen by pressing **ˆS**.

ˆS freezes cursor and text movement on the screen. Use **ˆQ** to unfreeze it.

ˆXˆX lets you execute a *shell* command without terminating the editing session.

Control temporarily returns to the Command Interpreter while the command is executed, then you can continue editing. For instance, if you want to read another file into the one you are editing but can not remember the file name, you can execute an ls command to find its name. When you type **ˆXˆX**, you will see the **Execute:** prompt on the information line. Type the command line and press the RETURN key. If the file was modified before you called **ˆXˆX**, you will see the **EXECUTING. PUT EDITS? (Y/W/N/F/!)** prompt.

Type **Y** if you want the file to be backuped and updated before executing the command. See the **QUIT** command for a description of the backup process.

Type **W** if you want the the file to be updated in place before executing the command.

Type **F** if you want file to be **forcibly** updated before executing the command. This command works similar to the **Y** command, but **ved** will ignore the state of the

−**readonly** flag and the access rights of the file. The latter will allow you to write into files without write permissions if you are super user. For execptions on this rule, see description of the −**readonly flag.**

Type **!** if you want file to be **forcibly** updated in place before executing the command.

Type **N** if you want to execute the command but don't want to save the changes before.

If the file has been locked before by another edit session or if the file has been modified more recently than the copy that has been used for the current session, you will not be allowed to save the file. Use the force commands (**F** or **!**) or write back using different file name.

In all cases, after returning from the command, the cursor will be on the same place as it was before typing **ˆXˆX**.

Note that if the *shell* command you want to execute requires the current file as an argument, you might need to update it. For instance, if you the file you are editing contains formatting commands that you want to test, you will need to update the file so that the version of the file you format and view is the most up-to-date one.

To abort the command while you can see the **Execute:** prompt, do not type a command; simply press the RETURN key. If you had started to type a command, use the DELETE key to erase it; the field following the prompt must be blank. To abort the command while you can see the **EXECUTING. PUT EDITS? (Y/W/N/F/!)** prompt, type anything except **Y**, **W**, **F**, **! N**.

<**ESC**>**ˆX**

executes the contents of the current take buffer.

You can type a *shell* command line in your file and then save it in a take buffer (using the **ˆT**, <**ESC**>**ˆT**, **ˆY** or <**ESC**>**ˆY** commands).

When you type <ESC>ˆX, you will see the **EXECUTING. PUT EDITS? (Y/W/N/F/!)** prompt on the information line.

Type **Y** if you want the file to be backuped and updated before executing the command. See the **QUIT** command for a description of the backup process.

Type **W** if you want the the file to be updated in place before executing the command.

Type **F** if you want file to be **forcibly** updated before executing the command. This command works similar to the **Y** command, but **ved** will ignore the state of the −**readonly** flag and the access rights of the file. The latter will allow you to write into files without write permissions if you are super user. For execptions on this rule, see description of the −**readonly flag.**

Type **!** if you want file to be **forcibly** updated in place before executing the command.

Type **N** if you want to execute the command but don't want to save the changes before.

If you don't want to execute the command, type anything except **Y**, **W**, **F**, **!** or **N**.

If the file has been locked before by another edit session or if the file has been modified more recently than the copy that has been used for the current session, you will not be allowed to save the file. Use the force commands (**F** or **!**) or write back using different file name.

<**ALT**>**ˆX**

executes the selection (what lies between the mark and the cursor).

You can type a *shell* command line in your file and select it.

When you type <**ALT**>**ˆX**, you will see the **EXECUTING. PUT EDITS? (Y/W/N/F/!)** prompt on the information line.

Type **Y** if you want the file to be backuped and updated before executing the command. See the **QUIT** command for a description of the backup process.

Type **W** if you want the the file to be updated in place before executing the command.

Type **F** if you want file to be **forcibly** updated before executing the command. This command works similar to the **Y** command, but **ved** will ignore the state of the **−readonly** flag and the access rights of the file. The latter will allow you to write into files without write permissions if you are super user. For execptions on this rule, see description of the **−readonly flag.**

Type **!** if you want file to be **forcibly** updated in place before executing the command.

Type **N** if you want to execute the command but don't want to save the changes before.

If you don't want to execute the command, type anything except **Y**, **W**, **F**, **!** or **N**.

If the file has been locked before by another edit session or if the file has been modified more recently than the copy that has been used for the current session, you will not be allowed to save the file. Use the force commands (**F** or **!**) or write back using different file name.

ˆ@ indicates the current line number. This number tells you how many lines exist between the beginning of the file and the current cursor position.

On the information line you will see **+Line:** *333* where *333* is the current line number.

<**ESC**>ˆ@
indicates how many lines exist between the current cursor position and the end of the file.

On the information line you will see **−Line:** *22* where *22* is the number of lines remaining in the file.

<**ALT**>ˆ@
indicates what line the mark is on, in relation to the beginning of the file.

On the information line you will see **+Line:** *45* where *45* is the positive line number on which the mark is located.

If there is no mark, you will see **No mark!** displayed on the far right of the information line.

<**ALT**><**ESC**>ˆ@
indicates what line the mark is on, in relation to the end of the file.

On the information line you will see **−Line:** *109* where *109* is the negative line number on which the mark is located.

If there is no mark, you will see **No mark!** displayed on the far right of the information line.

ˆ[is the same as using the <**ESC**> key.

ˆ] is <**ALT**>. Use this sequence of keystrokes to start an **ALT** command.

ˆX is the lead in character for an extended command.

<**ESC**>**ˆ]**
is the same as using <**ALT**><**ESC**>.

ˆˆ quotes the next character so that it is taken literally and then is converted to a control character. For instance, if you want to include a command in a macro definition, you must quote the keystroke sequence. Otherwise, the command will be executed rather than become part of your macro.

<**ESC**>ˆˆ
quotes the next character so that is becomes a 8 bit character.

<**ALT**>ˆˆ
quotes the next character so that is becomes a 8 bit control character.

<**ALT**><**ESC**>ˆˆ
quotes the next character so that you type it in in a hexadecimal notation.

<**ALT**><**ALT**>
goto tag that is a word located at cursor position. See **FILES** section for more information on the **tag** file.

<**ESC**><**ALT**><**ALT**>
pop the tag stack by one entry. Every time you go to a tag, the old location is puched on top of the tag stack.

<**ESC**>**%**
search for matching parenthesis. If this command is entered, the cursor is put to the parenthesis that matches the parenthesis where the cursor is currently located.

<**ESC**>*
Execute temporary macro. One temporary macro may be set up using the command <**ESC**>**:macro** command.

<**ESC**>**:**
introduce a colon command. See section **COLON COMMANDS** for more information.

COLON COMMANDS

Colon commands are introduced via <**ESC**>**:**
The command then may be entered on the (top) status line of **ved**.

The colon commands are:

backup Backup current file. This command is the same as when <**ESC**>**ˆC** is entered. It may be needed if for some reason it it not possible to enter the interrupt character of the terminal driver.

bind Print the current command binding for **ved**. Later versions of **ved** may allow to modify the binding with this command.

help print online help for available colon commands.

macro *macro body*
set up a temporary macro that may be executed via <**ESC**>*

map

map *from-string to-string*
The first form lists all currently active mappings, the second form sets up a new mapping.

next switch to next file in file list. This command may only be used if **ved** has been called with more than one file argument.

prev switch to previous file in file list. This command may only be used if **ved** has been called with more than one file argument.

quit Quit **ved**. This command is the same as when **ˆC** is entered. It may be needed if for some reason it it not possible to enter the interrupt character of the terminal driver.

set

set *set-variable set-args*
The first form lists the current settings of **ved**, the second form sets a specific property of **ved**. The vaiables that may be modified via the **set** command are:

autoindent
If set, **ved** activates the auto indent mode. Subsequent lines will be automatically indented with the same indent as the previous line. Switch off with <**ESC**>**:set noautoindent**

linelen *new-linelen*
Set linelength that is used as base for screen updates. This linenength usually is

retrived from the terminal driver but may be modified manually.

magic Switch to **magic** mode for pattern matching (this is the default when **ved** has been started). If in magic mode, matching is done using regular expression pattern matching. This mode may be switched off with <**ESC**>**:set nomagic**. If **magic** mode is turned off, matching is done without using regular expression matching even if the pattern contains *magic characters for pattern matching.*

markwrap
Mark wrapped lines with a \ character at the end of a visible terminal line (this is the default when **ved** has been started). This mode may be switched off with <**ESC**>**:set nomarkwrap**.

optline *opt-line*
Set the *optimal line*. This is the line (counted from the top of the terminal) where the cursor is adjusted to when **ved** does scrolling. It is set by default to the half of the number of lines of the terminal.

pmargin *page-margin*
Set the distance you need to have between the cursoe and the upper or lower border of the terminal. If you cross this border, **ved** will start to scroll and adjust the cursor to the optimal line. It is set by default to 0 and allows to place the cursor to any visible line on the screen.

psize *page-size*
Set pagesize that is used as base for screen updates. This pagesize usually is retrived from the terminal driver but may be modified manually.

tabstop *tab-width*
Set the size of a visible tab chacaracter. This width is set to 8 by default.

taglength *tag-len-value*
Set the number of characters that are used to compare a tag name with a tag label in the **tags** file. If it is set to 0 (the default), **ved** compares the complete tag length.

tags *tag-file-list*
Set the list if files that is searched when looking for a tag data base. The tag file list is separated by blanks and by default set to

tags .. /usr/lib/tags

This means that **ved** will first look for a file **tags** in the current directory, then search up for (directory by directory until the root directors is reached) a file **tags** and last it vill look for a file **/usr/lib/tags**.

wrapmargin *wrap-margin*
If **wrapmagin** is set, **ved** will activate the auto-warp mode. In auto-wrap mode, **ved** will automatically insert a new-line before a word if the cursor is less than *wrap-magin* from the right terminal border.

substitute */old/new/*
Substitute pattern *old* by *new*. The pattern *new* may contain an & for the *old* string.

tag *tag-name*
goto tag named *tag-name*. See **FILES** section for more information on the **tag** file.

MACROS

You may create macros in **ved** that are a combination of different **ved** basic commands. To create a macro, you need to edit the file **.vedmac** in your home directory. You may do this bey either explicitly typing *ved ˜/.vedmac* or by using the built in **ved** command <**ALT**>**ˆG**.

The file ˜**/.vedmac** contains a number of lines, each holds a macro definition like this:

L:/*ˆ[74ˆM-*/: Comment Line

Each macro definition has three fields:

name The name of the macro. In our case, the name is simply **L** and the macro may executed by typing <**ESC**>**L**.

function
The implementation of the macro. In our case the function contains **/*ˆ[74ˆM-*/** and will draw a C-comment line that contains 74 '-' chars as body.

comment
A comment for better understanding of the macro file. The comment is ignored.

The fields are separated by a colon.

Macros may be combined to form new macros. They also may recursively call each other and you may set *number* to a value greater than one to execute a macro more than once. There is currently no way to execute a macro in a conditional way.

MAPPINGS

You may create mappings in **ved** that contain arbitrary character sequences. To create a mapping, you need to edit the file **.vedmap** in your home directory.

The file ˜**/.vedmap** contains a number of lines, each holds a map definition like this:

\E[193z:ˆC:Quit Editor (L2)

Each map definition has three fields:

from The from string (left hand side) of the map. In our case, the name is the sequence **\E[193z** which is the output of the L2 function key on a Sun keyboard. Every time when you type the character sequence **\E[193z** or hit a function key that emits this character sequence, it will replaced by the the replacement string before the input is read by **ved.**

replacement
The replacement (right hand size or to string) of the macro. In our case the map expands to **ˆC** and will quit **ved**.

comment
A comment for better understanding of the map file. The comment is ignored.

The fields are separated by a colon. The text in the macro file is fed through **tdecode()** to allow the same escape sequences as you may have in the termcap file.

ENVIRONMENT

SHELL
For the shell to use for shell command escapes, file name globbing and similar.

HOME To find the private files ˜**/.vedmac** ˜**/.vedmap** and ˜**/.termcap**

TERM This

TERMCAP

TERMPATH

PATH The environment variable **PATH** is used to find the on-line help file of **ved**.

SLASH If the environment variable **SLASH** is set to 'off', no slashes are accepted in filenames.

VED_FTMPDIR
Fast tmp directory. The directory where the **ved** swap file **vedF.***processid* is stored. If

VED_FTMPDIR is not specified, the swap file will usually be stored in /tmp/.

VED_TMPDIR
General tmp directory. The directory where all **ved** tmp files except the swap file are stored. If **VED_FTMPDIR** is not defined but **VED_TMPDIR** is defined, all **ved** tmp files will be placed into this directory. If **VED_TMPDIR** is not specified, the other ved tmp files will usually be stored in /var/tmp/.

VED_STATISTICS
If set and **ved** has been compiled with statistics support, **ved** will print statistic information at the end of a session.

FILES

/etc/termcap
The system's terminal capabilities database.

˜/.termcap
The private terminal capabilities database.

˜/.vedmac
Contains private macro definitions.

˜/.vedmap
Contains private mappings.

˜/.vedtmp.*uid*
Contains the names of the files that have been edit from the current directory as well as the cursor position of the last visit. **Ved** also keeps track of the last edited file.

tags This file contains a list of tags that tell **ved** where a definition for a tag may be found. Tags may contain e.g. function names, macro names and typedefs. By default, **ved** looks for a **tag** file in the current directory and if not found in all directories that are closer to the root directory. If a **tag** file can be found in a directory closer to the root directory, **ved** will modify the file name from within the **tag** file to reflect the directory distance to this **tag** file. This property is unique to **ved**. **Ved** accepts **tag** files in the same notation as **vi**.

/tmp/vedF.*processid*
The **ved** swap file.

/var/tmp/vedB.*processid*
The **ved** backward delete buffer.

/var/tmp/vedD.*processid*
The **ved** forward delete buffer.

/var/tmp/vedT.*processid*
The current take buffer.

/var/tmp/vedT*pastename.processid*
One file for every named take buffer.

/var/tmp/vedP.*processid*
The **ved** crash recovery protocol buffer.

/var/tmp/vedX.*processid*
The **ved** shell command execute buffer.

.../man/help/ved.help
The on-line help file is assumed to be in the same directory tree as the **ved** binary.

SEE ALSO

change(1), **match**(1).

NOTES

To edit files greater than 2 GB, you need to compile **ved** with large file support or create a 64 bit binary.

Unlike **vi**, **ved** is not based on a line editor. This avoids annoying bugs but may cause to irritate people who are used to work with **vi**.

Since **ved** uses a private implementation of the **termcap** terminal independent screen package, **ved** uses **/etc/termcap** even on systems that usually emulate **termcap** with **terminfo**. This allows to use the portable terminal description on all operating systems. From my current point of view, **ved** will never switch towards **terminfo**. Although **terminfo** provides more functionality than **termcap**, it tends to be non-portable between different platforms. While **termcap** allows to have a private terminal database in the NFS mounted home directory, this tend to gives problems with **terminfo** if a vendor did make private additions. Because **termcap** explicitly names all capabilities, there are less problems with private additions; **terminfo** only uses an array index to identify a specific capability so this method only works if there is a global unique implementation.

Make sure that the **TERM** environment is assigned to the correct terminal type.

If the eighth bit of a character is "on", the tilde character ˜ appears before characters, i.e., *˜C*.

BUGS

None currently known.

AUTHOR
Jörg Schilling
Seestr. 110
D-13353 Berlin
Germany

Mail bugs and suggestions to:

schilling@fokus.gmd.de or **js@cs.tu-berlin.de** or **joerg@schily.isdn.cs.tu-berlin.de**