



The  
University  
Of  
Sheffield.

**Using the  
Unix  
vi  
Editor.**

**CiCS**  
Wendy Thomson  
November 1991  
AP-Unix1

© University of Sheffield

# Contents

<b>1.</b>	<b>INTRODUCTION.....</b>	<b>3</b>
<b>2.</b>	<b>STARTING UP .....</b>	<b>4</b>
2.1	EDITING AN EXISTING FILE .....	5
2.2	RECOVERING A CRASHED EDIT .....	5
<b>3.</b>	<b>EDITING A FILE.....</b>	<b>6</b>
3.1	CURSOR MOVEMENT .....	6
3.2	SEARCHING FOR TEXT .....	7
3.3	INSERTING CHARACTERS AND LINES .....	7
3.4	SEARCH AND REPLACE.....	8
3.5	OTHER COMMANDS FOR CHANGING TEXT .....	9
3.6	DELETING, MOVING AND COPYING TEXT .....	9
3.7	OTHER COMMANDS .....	10
3.8	SPECIFYING A RANGE OF LINES AND SEARCH STRINGS .....	11
3.9	ENDING THE EDIT .....	12
<b>4.</b>	<b>SUMMARY OF UNIX COMMANDS.....</b>	<b>13</b>
4.1	TYPES OF COMMAND.....	13
4.2	SPECIAL CHARACTERS FOR USE IN SEARCH STRINGS.....	13
4.3	SPECIFYING A LINE OR RANGE OF LINES .....	14
4.4	CURSOR MOVEMENT AND SCREEN MANIPULATION .....	15
4.5	SEARCHING FOR TEXT .....	17
4.6	INSERTING CHARACTERS AND LINES.....	17
4.7	CHANGING TEXT .....	18
4.8	DELETING AND MOVING TEXT .....	19
4.9	COPYING TEXT .....	21
4.10	COPYING TEXT TO AND FROM FILES .....	21
4.11	LEAVING THE VI EDITOR .....	22
4.12	OTHER COMMANDS.....	22

# 1. Introduction

The vi editor comes as quite a shock to people who have been using screen editors on other systems. In the interests of universality, it makes no use of special keys, and this gives rise to a few peculiarities. In order to insert new text you must type an appropriate command, and thereafter everything you type is copied to the file until such time as you press the **Esc** key. Whilst typing new text you cannot even move the cursor about (if you attempt this, it will merely add the cursor movement as additional characters in your file); you can however, **Backspace** over previous text and retype it.

Some editing commands are called **last-line** commands, and when you type these you see your command appearing at the bottom (last line) of the file; otherwise, the characters you type in as commands do not appear on the screen.



When you have finished typing in your new text, press the **Esc** key to exit from insert mode.

You can now correct any mistakes.

## 2.1 Editing an Existing File

To edit an existing file you must start up as follows:

- (a) Enter the editor by typing one of:

```
vi filename
vi = n filename
vi +/`text' filename
```

To enter the vi editor with a portion of the file displayed on the screen ready for editing. In the first case, the first few lines will be displayed; in the second case, the portion of the file centred around line *n* will be displayed; in the third case, the portion of the file centred around the first occurrence of the specified *text* will be displayed. In all cases, the message at the bottom of the screen will be

```
"filename" n lines, m characters
```

where *n* and *m* show how many lines and characters (respectively) are contained in the file.

You can now edit your file.

## 2.2 Recovering a Crashed Edit

In some circumstances you can recover if you have a crash when using the vi editor. If this happens try typing

```
vi -r
```

to get a list of saved files which can be recovered. It will not always work, but is worth a try.

## 3. Editing a File

This section describes some of the editing techniques available with vi. A more extensive list is given in Appendix C.

Throughout this Section, the actual commands are shown in `Courier` font.

### 3.1 Cursor Movement

The following keys may be pressed to move the cursor:

<b>Enter</b>	move to first non-space character on the next line.
- (minus)	move to first non-space character on the previous line.
h or ←	move one position left.
j or ↓	move one position down.
k or ↑	move one position up.
l or →	move one position right.
\$	move to the end of the current line.
<b>Ctrl b</b>	move back one screen.
<b>Ctrl f</b>	move forward one screen.
<b>Ctrl d</b>	move down half a screen.
<b>Ctrl u</b>	move up half a screen.

Most of the above may be preceded by a number to cause the command to be executed that number of times. For example if you press 5 then press **Enter**, this is equivalent to pressing **Enter** five times.

## 3.2 Searching for Text

You can search for lines containing a particular sequence of characters by using any of the following. When you type the initial `/` or `?` symbol, this will be shown on the bottom line of the screen; type in the required text, then the concluding `/` or `?` symbol, and finally press **Enter**.

<code>/string/</code>	move forwards to next occurrence of <code>string</code> ; if none is found, the search will continue from the top of the file down to the cursor position.
<code>?string?</code>	move back to previous occurrence of <code>string</code> ; if none is found, the search will continue from the bottom of the file, back up to the cursor position.
<code>n</code>	repeat previous search in the same direction.
<code>N</code>	repeat previous search in the opposite direction.

See Section 4.3.8 for more about specifying search strings.

## 3.3 Inserting Characters and Lines

The following commands are available for inserting characters and lines:

<code>a</code>	insert text to the right of the cursor.
<code>i</code>	insert text to the left of the cursor.
<code>A</code>	insert text at the end of the current line.
<code>I</code>	insert text at the beginning of the current line.
<code>o</code>	insert text as new line(s) below the cursor.
<code>O</code>	insert text as new line(s) above the cursor.

In each case, you must type the command letter, then type in your new text, then press **Esc** when you have finished. You can press **Enter** to continue your typing on a new line. You can use the **Backspace** key to backspace over your text so you can correct typing mistakes, but you cannot do any other editing. Note that the **Backspace** key does not wipe the text off the screen, but it does delete it; if you **Backspace** over some text without typing on top of it, the deleted characters will disappear when you press **Esc** to end the input.

### 3.4 Search and Replace

The command to do a search and replace, **on the current line only**, is

```
:s/search_string/replace_string/
```

or

```
:s/search_string/replace_string/g
```

the first form changes only the first occurrence of the string. whilst the second form changes all occurrences on the line.

If you wish to carry out your search-and-replace **over a block of lines**, you can write the command in the form

```
:lower,upper s/search_string/replace_string/g
```

For example

```
:1,100 s/abc/def/g
```

will change any occurrences of `abc` in lines 1 - 100 to `def`. You can use `$` to mean the last line of the file, and `.` to mean the current line, so you could type

```
:.,$ s/abc/def/g
```

To change lines from the current line to the end of the file, or

```
:1,$ s/abc/def/g
```

to make changes throughout the whole file. See Section 4.3.8 for more about specifying search strings and ranges. You can also add `c` to the command to cause it to ask for confirmation before making each change. When confirmation is requested, the line containing the occurrence is displayed near the bottom of the screen, with the next occurrence marked with `^`; type `y` (and press **Enter**) to confirm, or `n` (**Enter**) to leave unchanged. You can extricate yourself from the whole process by pressing **Ctrl** and `c`.

## 3.5 Other Commands for Changing Text

The following commands are available for changing text

- `cc` change the whole of the current line to the new text. This command works in a similar way to the commands in Section 4.3.3: you type the `cc`, then the new text and press **Esc** when you have finished.
- `rch` replace the character at the cursor position by the specified character. For example if you type `r*` the character at the cursor position will be replaced by `*`.
- `J` join the following line to the end of the current line.
- `~` change the case of the letter at the cursor position.

## 3.6 Deleting, Moving and Copying Text

The following commands are available for deleting text. Note that `d` and `dd` are not interchangeable: `dd` may be preceded by a number, and `d` must be preceded by `:range`; no other forms are possible.

- `x` delete the character at the cursor position.
- `dd` delete the current line. You can precede this command by a number to delete that number of lines, for example if you type `6dd` you will delete 6 lines; there must be no space between `dd` and the number.
- `:range d` delete a range of lines; the space between `:range` and `d` is optional. For example to delete lines 1 to 10, type either of
  - `:1,10 d`
  - `:1,10d`

See Section 4.3.8 for more about specifying a range of lines.

- `dw` delete from the cursor position to the start of the next word.
- `d0` `d` (followed by zero) delete from the beginning of the current line up to the cursor position.
- `D` delete the text on the current line to the right of the cursor.

These commands always copy the deleted text to a buffer, and you can copy the contents of the buffer back to the file by using either of:

- `p` (lower case `p`) insert buffer contents immediately to the right of the cursor position.
- `P` (upper case `P`) insert buffer contents immediately to the left of the cursor position.

It follows that if you delete text accidentally, you can replace it by pressing `P`. Alternatively, you can move the cursor to a different position, and then press `p` or `P` to move text from one part of the file to another.

If you wish to copy text from one part of the file to another you can use the command `yy`, (optionally preceded by a number) or `y` (preceded by `:range`) to 'yank' (i.e. copy) the current line or a number of lines to the buffer, then move the cursor to the appropriate place, and use `p` or `P` to insert the copy. Normally you will get a message to indicate how many lines have been yanked, but if you use `yy` to yank fewer than 5 lines you will not get any visible sign that anything has happened; however, when you have moved the cursor and pressed `p` or `P`, the insertion will work.

### 3.7 Other Commands

The following commands may also be useful:

- `u` undo the previous screen command.
- `.` repeat the previous command.
- Ctrl L** redraw the screen; you can use upper or lower case `L`.

The following commands will be displayed on the bottom line of the screen as you type them. You must type in the command, then press **Enter**. Spaces must be included where they are indicated; for example, you must type `:set nu`, and not `:setnu`.

- `:u` undo the previous last-line command.
  - `:set nu` display line numbers.
  - `:set nonu` switch off display of line numbers.
  - `:set list` display non-printing characters (e.g. end-of-line).
  - `:set nolist` switch off display of non-printing characters.
  - `:r filename` insert the contents of the specified file after the line containing the cursor.
-

---

<code>:range w filename</code>	write the lines specified by <i>range</i> (see Section 4.3.8) to the specified file. If no range is specified, then the whole file is written; if no <i>filename</i> is specified, the existing <i>filename</i> is assumed; if a file is specified, it must not already exist.
<code>:range w! filename</code>	as above, but the file may exist, in which case it will be over-written.
<code>: ! command</code>	executes the specified Unix <i>command</i> .

### 3.8 Specifying a Range of Lines and Search Strings

Some last-line commands allow you to specify a range of lines to which the command is to be applied. You must specify the range in the form

*lower, upper*

Where *lower* and *upper* may take any of the following forms:

<i>n</i>	line number <i>n</i> .
.	current line.
\$	last line.
<i>/string/</i>	identifies the line containing the specified <i>string</i> , searching forwards; if none is found, the search will continue from the top of the file downwards.
<i>?string?</i>	identifies the line containing the specified <i>string</i> searching backwards; if none is found, the search will continue from the bottom of the file upwards.

The search string may be preceded by ^ to limit the search to lines beginning with the specified string or it may be followed by \$ to limit the search to lines ending with the specified string. The search strings may contain special characters as follows:

.	any single character.
ch*	any sequence of zero or more of the specified character.
[ <i>charlist</i> ]	any of the specified characters.
[^ <i>charlist</i> ]	none of the specified characters.

Some examples of search strings:

<i>/a.c/</i>	identifies lines containing abc, adc, aec, a1c, a2c, ...
--------------	--

---

<code>/^a.c/</code>	identifies lines starting abc, adc, ...
<code>/a.c\$/</code>	identifies lines ending abc, adc, ...
<code>/ax*c/</code>	identifies lines containing ac, axc, axxc, axxc, ...
<code>/[e-h, r]</code>	identifies lines containing one or more of e, f, g, h, and r.
<code>/[^e-h, r]</code>	identifies lines starting with e, f, g, h, or r.

Some example of ranges:

<code>:10, 100</code>	identifies lines 10 to 100 inclusive.
<code>:. , /abc/</code>	identifies lines from the current line to the line containing the string abc.

### 3.9 Ending the Edit

When you have finished your edit, you must use a `:w` command to save the entire file, and a `:q` command to quit from the editor. These commands can be run together to give

```
:wq filename
```

where *filename* is to be the name of the file. If you are editing an existing file then you can omit the *filename*, and the new version will over-write the old.

If your edit went wrong and you wish to quit from the editor leaving the original file unchanged, type

```
:q!
```

You will get no warnings, so don't do this unless you mean to.

## 4. Summary of Unix Commands

### 4.1 Types of Command

Commands for use with the vi editor are of two kinds. The first group do not normally appear on your screen as you type them, must not contain spaces and, unless stated otherwise, do not require you to press **Enter** at the end. The second group begin with the `:` symbol, they appear on the last-line of the screen as you type (and are sometimes called 'last-line' commands) and you must press **Enter** to terminate them.

### 4.2 Special Characters for Use in Search Strings

<code>.</code>	any single character.
<code>ch*</code>	any sequence of zero or more of the specified character.
<code>[charlist]</code>	any of the specified characters; characters may be specified singly or in hyphenated names separated by commas.
<code>[^charlist]</code>	characters other than those specified.
<code>^</code>	can precede the string to indicate that the line must start with the string.
<code>\$</code>	can come at the end of the string to indicate that the line must end with the string.

## 4.3 Specifying a Line or Range of Lines

Some commands allow you to specify a target line or a range of lines. The target line may take any of the following forms:

<code>n</code>	line number <code>n</code> .
<code>.</code>	the current line (i.e. the line containing the cursor).
<code>\$</code>	the last line of the file.
<code>/string/</code>	the line containing the string searching forwards.
<code>?string?</code>	the line containing the string searching backwards.
<code>`letter</code>	the line marked with the specified letter (see the <code>m</code> command).
<code>``</code>	the line where the cursor was before the last <code>/</code> <code>?</code> or <code>G</code> command.

The search strings may contain any of the characters described in Section 4.2.

A range of lines must be specified in the form

*upper,lower*

where *upper* and *lower* may take any of the forms described above.

For example, the command to move a range of lines to a new position immediately after a target line is

`:range m target`

so the command

`: 3,10 m .`

would move lines 3 to 10 to a new position immediately after the current line.

## 4.4 Cursor Movement and Screen Manipulation

Unless stated otherwise, the following commands may be preceded by a number to cause the command to be obeyed that number of times. For example: if you press 3 then press h, the cursor will move 3 places to the left; if you press 5, then press **Ctrl** and F together, the cursor will move forwards 5 screens.

h or ←	move one position left.
j or ↓	move one position down.
k or ↑	move one position up.
l or →	move one position right.
w	move one word right.
W	move one word right, past punctuation.
b	move one word left.
B	move one word left, ignoring punctuation.
e	move to end of current word.
<b>Enter</b>	move to first non-space character of next line.
<b>Space bar</b>	move one position to the right (same as l).
G	move to last line of file, if preceded by a number n move to nth line of file.
H	move to first non-space character of top line of screen; if preceded by a number n, move to first non-space character of nth line of screen.
L	move to first non-space character on last line of screen; if preceded by a number n, move to first non-space character of nth line from bottom of screen.
M	move to first non-space character of middle line of screen, may not be preceded by a number.
0 (zero)	move to beginning of current line.
- (minus)	move to first non-space character on previous line.
^	move to first non-space character of current line.
<i>col-num</i>	move to specified column number on current line.
+	move to first non-space character on next line.
\$	move to end of current line; if preceded by a number n, move to end of nth line further on.

<code>%</code>	move to parenthesis (of any type) that matches the one at the cursor position.
<code>Ctrl b</code>	moves back one screen.
<code>Ctrl f</code>	moves forward one screen.
<code>Ctrl d</code>	moves forward half a screen., if preceded by a number <code>n</code> , move forward <code>n</code> lines, and cause all subsequent use of <code>Ctrl d</code> and <code>Ctrl u</code> to travel <code>n</code> lines instead of half a page.
<code>Ctrl u</code>	similar to <code>Ctrl d</code> , but move back instead of forward.
<code>Ctrl e</code>	display an additional line at bottom of screen (i.e. move screen display up one line).
<code>Ctrl y</code>	display an additional line at top of screen (i.e. move screen display down one line).
<code>Ctrl g</code>	display information about current line.
<code>Ctrl l</code>	re-draws screen (for example to remove displays at bottom of screen).
<code>z-</code>	make current line the bottom line of the screen.
<code>z.</code>	centre the screen at current line.*
<code>z</code>	make current line the top line of screen.*
<code>:set ic</code>	ignore case when searching.*
<code>:set noic</code>	distinguish case when searching.*
<code>:set list</code>	show `hidden' characters.*
<code>:set nolist</code>	cancel the showing of `hidden' characters.*
<code>:set nu</code>	display line number.*
<code>:set nonu</code>	switch off display of line number.*
<code>:=</code>	display current line number.*

- You must press **Enter** after typing these commands.

## 4.5 Searching for Text

The following commands may be used to search for lines containing the specified string which may contain any of the special characters described in Section C.1.2.

<code>/string/</code>	search forwards for <i>string</i> . If no occurrence of the <i>string</i> has been found when the end of the file is reached, the search will continue from the beginning of the file down to the current position.*
<code>?string?</code>	search backwards for <i>string</i> . If no occurrence of the <i>string</i> has been found when the beginning of the file is reached, the search will continue from the end of the file, back up to the current position.*
<code>n</code>	repeat previous search for <i>string</i> . (Note this is letter <code>n</code> , not to be confused with <code>n</code> which is used to designate a number in Section B.2).
<code>N</code>	repeat previous search, but in opposite direction.
<code>fch</code>	search forwards on current line for a specified character.
<code>Fch</code>	search backwards on current line for a specified character.
<code>;</code>	repeat previous <code>f</code> or <code>F</code> command.
<code>,</code>	repeat previous <code>f</code> or <code>F</code> command in opposite direction.

\* You must press **Enter** after typing these commands.

## 4.6 Inserting Characters and Lines

For the following commands, type the appropriate letter for the command, then type the text to be inserted (including pressing **Enter** for a new line), then press **Esc** to terminate the input. Use **Backspace** to correct any mistakes.

<code>a</code>	insert text to right of cursor.
<code>A</code>	insert text at end of current line.
<code>i</code>	insert text to left of cursor.
<code>I</code>	insert text at beginning of current line.

---

- 
- o insert text as new line(s) below cursor.
  - O insert text as new line(s) above cursor.

## 4.7 Changing Text

For the commands `cw`, `cc`, `C`, `R` and `s`, type the appropriate letter(s) for the commands, then type the text which is to replace the existing text, then press **Esc** to terminate the input. The new text can contain more characters than the text it is replacing; you may press **Enter** in the new text to give several lines.

- `cw` change the rest of the current word (i.e. text to the right of the cursor) to the new text; press **Esc** to terminate new text.
- `cc` change the whole of the current line to the new text; press **Esc** to terminate new text.
- `C` change the part of the current line to the right of the cursor to the new text; press **Esc** to terminate new text.
- `s` substitute the new text for the character at the cursor position; press **Esc** to terminate new text.
- `rch` replace the character at the cursor position by the specified *character*. This character can be **Enter** to give a line break.
- `R` replace the character at the cursor position by new text; press **Esc** to terminate new text.
- `J` join the following line to the end of the current line.
- `j` join together all lines within the range; see Section C.1.3 for how to specify range.
- `~` change the case of the letter at the cursor position.
- `xp` transpose character at the cursor and character to the right.

`:range s/oldtext/newtext/gc`

replace all occurrences of *oldtext* by *newtext* in the specified range of lines, giving confirmation of each change. The range may be specified as described in Section C.1.3; `g` may be omitted if only the first occurrence of *oldtext* on each line is to be changed; `c` may be omitted if no confirmation is to be given. When confirmation is requested, type `y` or `n` followed by **Enter**; press **Ctrl** and `C` to terminate the search.\*

.

repeat previous change. This works in slightly different ways depending on the change being repeated.

\* You must press **Enter** after typing this command.

## 4.8 Deleting and Moving Text

All the following commands write deleted text to a buffer.

x	delete character at cursor position.
X	delete character to left of cursor position.
dd	delete the current line; this command can be preceded by a number <i>n</i> to delete <i>n</i> lines.
D	delete the text on the current line to the right of the cursor.
: <i>range</i> d	delete a range of lines (see Section C.1.3). Press <b>Enter</b> after typing this command.
d <i>cursor_move</i>	delete from the cursor position to the position designated by <i>cursor_move</i> ; <i>cursor_move</i> may be specified in any of the ways listed in Section C.2 (except for <b>Ctrl</b> key combinations or arrow keys), or Section C.3. For example <i>dh</i> , <i>de</i> , <i>dt</i> , <i>d/abc/</i> , <i>dfx</i> are all valid commands. You are advised to try these composite commands out before doing anything important.

To move the deleted text to a new position, move the cursor to the required place and then use either of

p	insert text from the buffer immediately to the right of the cursor.
P	insert text from the buffer immediately to the left of the cursor.

alternatively you can use the command

: *range* m *target*

to move a range of lines and insert them after the *target* line. The *target* and *range* may be specified in any of the ways described in Section C.1.3. You

---

must press **Enter** after typing this command.

## 4.9 Copying Text

To copy text, use the commands `yy`, `:range y, y cursor_move`, to copy text to the buffer, then use `p` or `P` to insert the text at the appropriate place. The `y` commands work in the same way as the `d` commands described in Section C.6 except that text is not deleted from its original position. Alternatively, you can use the command

```
:range co target
```

to copy a range of lines and insert them immediately after the target line. The range and target can be specified as described in Section C.1.3. You must press **Enter** after typing this command.

## 4.10 Copying Text to and From Files

In the following commands, target and range can be specified as described in Section C.1.3.

<code>:target r filename</code>	insert the contents of a file after the target line. If no target is given, the file is inserted after the current line.
<code>:range w filename</code>	write a range of lines to a file, which must not already exist. If range is omitted all lines are written; if <code>filename</code> is omitted, the current filename is assumed.
<code>:range w! filename</code>	as above but will over-write contents of an existing file.

You must press **Enter** after typing all the above commands.

## 4.11 Leaving the vi Editor

The following commands enable you to leave the vi editor, with various options of saving the edited file. The term 'original file' means the original name of the file being edited.

<code>zz</code>	save edited file, over-writing the original version, and leave vi.
<code>:wq filename</code>	save edited file, to specified new file, and leave vi. If the specified file exists, the command will fail; if no file is specified, the edited file will over-write the original file.*
<code>:w! filename</code>	save edited file to specified file which will be over-written without warning if it already exists.
<code>:q!</code>	leave editor, losing all your edits without warning.*

\* You must press **Enter** after typing these commands.

## 4.12 Other Commands

<code>.</code>	repeat the last command that changed the file.
<code>u</code>	undo previous (screen) command.
<code>U</code>	undo all changes to current line, but only if no other edits have been carried out in between.
<code>mletter</code>	mark the current line with the specified <i>letter</i> (a-z)
<code>:u</code>	undo previous last-line command.*
<code>:range/string/command</code>	carries out the command on all lines within the range containing string; range is as described in Section C.1.3, and string may contain characters as described in Section C.1.2.*
<code>:range v /string/command</code>	similar to <code>g</code> , but command is executed for lines that do not contain string.*
<code>:target k label</code>	assign the specified label, which must be a single letter to the target line. The target is specified as described in Section C.1.3; if no target is given the cursor line is labelled. The label may now be used in the line specification as described in C.1.3.*
<code>:! command</code>	execute the specified Unix <i>command</i> .*

`:e filename`

start editing new file.\*

\* You must press **Enter** after typing these commands.