**Vim**

Command mode

h - Move cursor left

j - Move cursor down

k - Move cursor up

l - Move cursor right

i - Switch to insert mode

ESC - Switch to command mode

:q - Quit vim

:q! - Quit vim (ignoring unsaved edits)

:w - Save document

:wq - Save document and quit vim

i - Switch to INSERT mode before current position

I - Switch to INSERT mode at the beginning of the current line

a - Switch to INSERT mode after current position

A - Switch to INSERT mode at the end of the current line

o - Switch to INSERT mode after opening a new line after the current line

O - Switch to INSERT mode after opening a new line before the current line

w - Skip to next word

W - Skipe to next WORD (WORDs are only delineated by spaces)

e - Move to end of word

E - Move to end of WORD

r - Replace a single character

R - Replace all following characters

c - Change (Delete and enter insert mode) whatever we specify

C - Change until the end of a line

Shift + - go to start of the same line

* Go tho the start of the prev line

Shift ++ go to the start of the next line

* Go to the start of the line

Shit + $ end of the line

Shift+ # cursor move to previous same word

:set number print numbers in front of lines

: 273 go to 273th line

Delete multiple lines Select lines in visual mode and press d

D delte signle line #act like cut

Dw delete word

Diw delete inside word

dt<?> delete upto ?

Y copy with modifier

Y copy whole line

P paste in next line

Cap P paste in next line

Add four white spaces

Go to visual mode then : norm I----

(----are four letters or white spaces)

y - Yank what we specify

Y - Yank the current line

p - Paste after the cursor's position

P - Paste before the cursor's position

d - Delete what we specify (But also copy it - 'Cut')

D - Delete to the end of current line (But also copy it - 'Cut')

Here's some commands we learned:

gg - Skip to the beginning of a document

G - Skip to the end of a document

number + G - Skip to that line number

/ - Search forward

? - Search backward

n - Advance to next search result

N - Advance to previous search result

m + [letter] - Mark a line

' + [letter] - Move cursor to a mark

% - Jump to matching bracket

less-than - Indent

greater-than - De-indent

= - Properly Indent

~ - Change case of character

vimdiff:

ctrl+w K horizontal to vertical split

ctrl+w H vertical to horizontal split

:set nu - Line Numbers

:set rnu - Relative Line numbers

:set autoindent - Automatically indent

:set noerrorbells - Disable error bells

:set showmode - Show the working mode

:set colorcolumn=X - Draw a column at specified character

:set ruler - Show position info

:set nohlsearch - Disable search highlighting

:set bg=dark/light - Change foreground to match dark/light background

:set tabstop=X - Set tab to be X spaces

:colors [theme] - Set a theme

### **complete key binding reference**

|  |  |  |
| --- | --- | --- |
| **Key** | **Action** | **Followed by** |
| a | enter insertion mode after current character | text, ESC |
| b | back [word](https://hea-www.harvard.edu/~fine/Tech/vi.html" \l "defs) |  |
| c | change command | [cursor motion command](https://hea-www.harvard.edu/~fine/Tech/vi.html" \l "defs) |
| d | delete command | [cursor motion command](https://hea-www.harvard.edu/~fine/Tech/vi.html" \l "defs) |
| e | end of [word](https://hea-www.harvard.edu/~fine/Tech/vi.html" \l "defs) |  |
| f | find character after cursor in current line | character to find |
| g | [UNBOUND](https://hea-www.harvard.edu/~fine/Tech/vi.html" \l "defs) |  |
| h | move left one character |  |
| i | enter insertion mode before current character | text, ESC |
| j | move down one line |  |
| k | move up one line |  |
| l | move right one character |  |
| m | mark current line and position | mark character tag (a-z) |
| n | repeat last search |  |
| o | open line below and enter insertion mode | text, ESC |
| p | put buffer after cursor |  |
| q | [UNBOUND](https://hea-www.harvard.edu/~fine/Tech/vi.html" \l "defs) |  |
| r | replace single character at cursor | replacement character expected |
| s | substitute single character with new text | text, ESC |
| t | same as "f" but cursor moves to just before found character | character to find |
| u | Undo (ctrl+ r for redo) |  |
| v | [UNBOUND](https://hea-www.harvard.edu/~fine/Tech/vi.html" \l "defs) |  |
| w | move foreward one [word](https://hea-www.harvard.edu/~fine/Tech/vi.html" \l "defs) |  |
| x | delete single character |  |
| y | yank command | [cursor motion command](https://hea-www.harvard.edu/~fine/Tech/vi.html" \l "defs) |
| z | position current line | CR = top; "." = center; "-"=bottom |
| A | enter insertion mode after end of line | text, ESC |
| B | move back one [Word](https://hea-www.harvard.edu/~fine/Tech/vi.html" \l "defs) |  |
| C | change to end of line | text, ESC |
| D | delete to end of line |  |
| E | move to end of [Word](https://hea-www.harvard.edu/~fine/Tech/vi.html" \l "defs) |  |
| F | backwards version of "f" | character to find |
| G | goto line number prefixed, or goto end if none |  |
| H | home cursor - goto first line on screen |  |
| I | enter insertion mode before first non-whitespace character | text, ESC |
| J | join current line with next line |  |
| K | [UNBOUND](https://hea-www.harvard.edu/~fine/Tech/vi.html" \l "defs) |  |
| L | goto last line on screen |  |
| M | goto middle line on screen |  |
| N | repeat last search, but in opposite direction of original search |  |
| O | open line above and enter insertion mode | text, ESC |
| P | put buffer before cursor |  |
| Q | leave visual mode (go into "ex" mode) |  |
| R | replace mode - replaces through end of current line, then inserts | text, ESC |
| S | substitute entire line - deletes line, enters insertion mode | text, ESC |
| T | backwards version of "t" | character to find |
| U | restores line to state when cursor was moved into it |  |
| V | [UNBOUND](https://hea-www.harvard.edu/~fine/Tech/vi.html" \l "defs) |  |
| W | foreward [Word](https://hea-www.harvard.edu/~fine/Tech/vi.html" \l "defs) |  |
| X | delete backwards single character |  |
| Y | yank entire line |  |
| Z | first half of quick save-and-exit | "Z" |
| 0 | move to column zero |  |
| 1-9 | [numeric precursor to other commands](https://hea-www.harvard.edu/~fine/Tech/vi.html" \l "prefix) | [additional numbers (0-9)] command |
|  | (SPACE) move right one character |  |
| ! | shell command filter | [cursor motion command](https://hea-www.harvard.edu/~fine/Tech/vi.html" \l "defs), shell command |
| @ | vi eval | buffer name (a-z) |
| # | [UNBOUND](https://hea-www.harvard.edu/~fine/Tech/vi.html" \l "defs) |  |
| $ | move to end of line |  |
| % | match nearest [],(),{} on line, to its match (same line or others) |  |
| ^ | move to first non-whitespace character of line |  |
| & | repeat last ex substitution (":s ...") not including modifiers |  |
| \* | [UNBOUND](https://hea-www.harvard.edu/~fine/Tech/vi.html" \l "defs) |  |
| ( | move to previous sentence |  |
| ) | move to next sentence |  |
| \ | [UNBOUND](https://hea-www.harvard.edu/~fine/Tech/vi.html" \l "defs) |  |
| | | move to column zero |  |
| - | move to first non-whitespace of previous line |  |
| \_ | similar to "^" but uses numeric prefix oddly |  |
| = | [UNBOUND](https://hea-www.harvard.edu/~fine/Tech/vi.html" \l "defs) |  |
| + | move to first non-whitespace of next line |  |
| [ | move to previous "{...}" section | "[" |
| ] | move to next "{...}" section | "]" |
| { | move to previous blank-line separated section | "{" |
| } | move to next blank-line separated section | "}" |
| ; | repeat last "f", "F", "t", or "T" command |  |
| ' | move to marked line, first non-whitespace | character tag (a-z) |
| ` | move to marked line, memorized column | character tag (a-z) |
| : | ex-submode | ex command |
| " | access numbered buffer; load or access lettered buffer | 1-9,a-z |
| ~ | reverse case of current character and move cursor forward |  |
| , | reverse direction of last "f", "F", "t", or "T" command |  |
| . | repeat last text-changing command |  |
| / | search forward | search string, ESC or CR |
| < | unindent command | [cursor motion command](https://hea-www.harvard.edu/~fine/Tech/vi.html" \l "defs) |
| > | indent command | [cursor motion command](https://hea-www.harvard.edu/~fine/Tech/vi.html" \l "defs) |
| ? | search backward | search string, ESC or CR |
| ^A | [UNBOUND](https://hea-www.harvard.edu/~fine/Tech/vi.html" \l "defs) |  |
| ^B | back (up) one screen |  |
| ^C | [UNBOUND](https://hea-www.harvard.edu/~fine/Tech/vi.html" \l "defs) |  |
| ^D | down half screen |  |
| ^E | scroll text up (cursor doesn't move unless it has to) |  |
| ^F | foreward (down) one screen |  |
| ^G | show status |  |
| ^H | backspace |  |
| ^I | (TAB) [UNBOUND](https://hea-www.harvard.edu/~fine/Tech/vi.html" \l "defs) |  |
| ^J | line down |  |
| ^K | [UNBOUND](https://hea-www.harvard.edu/~fine/Tech/vi.html" \l "defs) |  |
| ^L | refresh screen |  |
| ^M | (CR) move to first non-whitespace of next line |  |
| ^N | move down one line |  |
| ^O | [UNBOUND](https://hea-www.harvard.edu/~fine/Tech/vi.html" \l "defs) |  |
| ^P | move up one line |  |
| ^Q | XON |  |
| ^R | does nothing (variants: redraw; multiple-redo) |  |
| ^S | XOFF |  |
| ^T | go to the file/code you were editing before the last tag jump |  |
| ^U | up half screen |  |
| ^V | [UNBOUND](https://hea-www.harvard.edu/~fine/Tech/vi.html" \l "defs) |  |
| ^W | [UNBOUND](https://hea-www.harvard.edu/~fine/Tech/vi.html" \l "defs) |  |
| ^X | [UNBOUND](https://hea-www.harvard.edu/~fine/Tech/vi.html" \l "defs) |  |
| ^Y | scroll text down (cursor doesn't move unless it has to) |  |
| ^Z | suspend program |  |
| ^[ | (ESC) cancel started command; otherwise [UNBOUND](https://hea-www.harvard.edu/~fine/Tech/vi.html" \l "defs) |  |
| ^\ | leave visual mode (go into "ex" mode) |  |
| ^] | use word at cursor to lookup function in tags file, edit that file/code |  |
| ^^ | switch file buffers |  |
| ^\_ | [UNBOUND](https://hea-www.harvard.edu/~fine/Tech/vi.html" \l "defs) |  |
| ^? | (DELETE) [UNBOUND](https://hea-www.harvard.edu/~fine/Tech/vi.html" \l "defs) |  |

### **Definitions**

* **UNBOUND** - this key is not normally bound to any vi command
* **word** - a lower-case word ("w", "b", "e" commands) is defined by a consecutive string of letters, numbers, or underscore, or a consecutive string of characters that is not any of {letters, numbers, underscore, whitespace}
* **Word** - an upper-case word ("W", "B", "E" commands) is a consecutive sequence of non-whitespace.
* **sentence**
* **paragraph**
* **cursor motion command** - any command which positions the cursor is ok here, including the use of numeric prefixes. In addition, a repeat of the edit command usually means to apply to the entire current line. For example, "<<" means shift current line left; "cc" means replace entire current line; and "dd" means delete entire current line.

### **Key Bindings in Editing Modes**

While in any edit mode (insert, replace, etc.) there are some keys that are used to adjust behaviour, rather than just to insert text.

* ESC - leave edit mode, return to command mode
* ^D - move line backwards one shiftwidth. shiftwidth must be set, and either the line must be newly added, or ^T must have been used.
* ^T - move all after cursor forwards one shiftwidth
* ^H - deletes text that was entered during the current edit mode. Most versions of vi do not allow deleting to previous line.
* ^V - insert next character even if it is a editing character.

### **Repitition Counts**

Most commands can be prefixed with a multidigit number, that influences the way the command works.

|  |  |
| --- | --- |
| z | position *n*th line number |
| G | goto *n*th line number |
| | | goto *n*th column number |
| r | replace next *n* characters |
| s | substitute for next *n* characters |
| << | shift *n* lines left one shiftwidth |
| ^ | ignored? |
| \_ | advance *n*-1 lines |
|  |  |

Need to differentiate between such things as 5yj vs. y5j?

### **Multibuffer**

Standard vi does have an ability to toggle between two different files. These will be the last two edited files (edit new files with :efilename) To switch files, use control-^.

These filenames can be reffered to in ex commands, and subshell filters, using two special characters: "%" refers to the current file, and "#" refers to the previous file. Here's some handy things you can do with this feature:

:map v :!chmod 644 %^[ make world-readable

:map q :!ci -l %^[ RCS checkin

:map V :!diff # %^[ compare previous and current files

### **Tags**

Tags are cool, but I don't use them. Go figure. Maybe I'll write something up here someday.

### **Mappings and Abbreviations**

:map lets you bind a list of keystrokes to a shortcut in command-mode. This shortcut can be a multiple-key sequence (with limitations), and the commands within can enter and exit edit-mode. Some examples of :map can be found above, in the multibuffer section. Below is a list of all the normally unbound keys in vi command-mode.

g q v K V # \* \ = ^A ^C ^I ^K ^O ^V ^W ^X ^[ ^\_

When you try to map multiple key sequences, you won't be able to start them with lower or upper case letters ("Too dangerous to map that"), but the punctuation and control characters are fair game. In addition, : can't be mapped, and sometimes a few other keys. Multiple key sequences can also be very useful with terminal-generated sequences, which is why the escape key is bindable. I have my xterm set to generate =f1 for function key one, and so on, so all the function keys are easier to use with bindings.

If you use multiple key shortcuts, you'll want to know about the timeout variable. With :se timeout, you have a limited time to generate the key sequence. This is useful if the key sequences are terminal generated. With :se notimeout, it just keeps waiting until the next character does or doesn't match any possible current sequences.

:map! lets you bind a list of keystrokes to a shortcut in edit-mode. This is useful for adding editing commands to edit mode. One popular trick is to bind the arrow keys to move up and down while (apparently) staying in edit-mode, as in the last four lines below.

:map! ^? ^H Make delete act like backspace

:map! ^[OA ^[ka xterm arrow sequences will

:map! ^[OB ^[ja exit edit-mode, move the

:map! ^[OC ^[la cursor, and re-enter edit-mode.

:map! ^[OD ^[ha

If you use the above trick for arrow-keys in edit-mode, you'll want to set timeout, because otherwise you won't get beeps at all when you hit escape, only when you use the next keystroke. With timeout, you get the beep, but after the timeout. Since both of these are annoying, it may be a useful choice to avoid multikey sequences that involve escape, as a matter of taste. Also, many systems now set up command-mode arrow keys in vi by default, which also leads to the same problem.

:ab lets you bind a key sequence to an abbreviation, for use in edit-mode. Abbreviations don't fire until vi decides that you've typed the shortcut as a whole word. So if taf is a shortcut for Thomas A. Fine, and I type taffy, it won't substitute because I didn't enter taf as a word by itself. (If I'd used :map!, then taffy would do the replacement before I got to the second "f".)

Abbreviations are echoed normally until complete, therefore the abbreviation can't contain escape (you'd leave edit-mode before completing the abbreviation), but the replacement expression can contain escape, and can leave and return to edit-mode.

:ab teh the

:ab #d #define

:ab #i #include

:ab cmain main(argc,argv)^Mint argc;^Mchar \*\*argv;^M{^M}^[O

:ab cmmap mmap(NULL,st.st\_size,PROT\_READ,MAP\_SHARED,fd,0);

:ab readsig ^[G:r ~/misc/sig^M

}

To keep a live abbreviation from going off in your hands, use ^V. For instance, if I want to type teh but have the the abbreviation above, I can let it "fix" it, then back up and unfix it; or I can type "teh^V..." and it won't expand the abbreviatoin.

[vi macros document](http://soma.npa.uiuc.edu/docs/vi.macros)

### **Repeating with .**

Commands can be repeated with the redo command, normally bound to ".", but I've found this to be occasionally unpredictable. If you use multiple key sequences in a macro, and vi is waiting to see if one of those sequences might complete, and you start a new command here, it won't be noticed by the redo. (Solaris, HPUX at least).

### **System Differences**

* Older versions of vi didn't automatically set up arrow-keys in command-mode; they didn't interfere with the beep. (Maybe multiple key bindings were new at the same time???)
* Some versions of vi have encryption, some don't.
* Options processing is handled differently from version to version. Solaris prefers -c command in place of +command, and -L instead of -r.
* Differnt systems may have other keys besides : that are "Too dangerous to map that".
* The size of macros (:map, etc.) are limited in different ways on different systems.
* On some systems the environment variable EXINIT overrides .exrc files (Solaris, HPUX), other systems it enhances it (SunOS???, FreeBSD).
* nvi allows backspace to previous line in edit-mode (if previous line was edited) (FreeBSD).
* variants of vi that have multiple undo have different styles. One style (linux) uses u as undo, and control-R as redo. The other style (FreeBSD) u acts normally, but . continues on in the same direction as the last u (whether it was undo or redo). I prefer the latter because it doesn't interfere as much with traditional vi behaviour.

[VI Reference Manual](http://www.engin.umd.umich.edu/ECS/online_help/vi.html) from the University of Michigan at Dearborn