Command (Normal) Mode

In command mode, you can enter general commands such as to write a file, delete a line, perform a search-and-replace, or move the cursor. Many commands take a number prefix indicating number of times to perform the action, e.g., 13dd deletes 13 lines.

Ex Commands

Vi accepts commands from the older *ex* line editor. Type a colon from within command mode, and the cursor jumps to the bottom left corner of the window, displaying a colon. Type your *ex* command, press <enter>, and then the cursor will return to its previous position. *Ex* commands are indicated in this document with their leading colon.

Editing Commands

All deletes move to a buffer.

~	Change case of character under cursor.
x	Delete character under cursor. Same as <del>.
r<ch>	Replace character under cursor with character <ch>.
D	Delete to end-of-line.
dd	Delete entire line.
n dd	Delete *n* lines, starting with the current line.
J	Join the line below to the end of this line.
Y	Yank copy of line into buffer.
P	Insert buffer contents between this line and the previous one.
p	Insert buffer at cursor.
u	Undo last change.
U	Restore current line.
S or cc	Substitute entire line. <esc> to end.
C	Change rest of line.
:s/*str1*/*str2*/g	Change every *str1* to *str2* in current line.
:g/*str1*/s//*str2*/g	*or*
:%s/str1/str2/g	Change every *str1* to *str2*.
:g/*str1*/s//*str2*/gc	Change every *str1* to *str2*, with confirmation.
xp	Change order of two letters
ddp	Reverse order of two lines.
<esc>	Abandon incomplete vi command.

Searching Commands

/*str*	(find forward) Move cursor to next occurrence of string *str*.
?*str*	(find backward) Move cursor to previous occurrence of string *str*. Search strings may use regular expression notation: ^, $, ., *,[x-y], etc.
n *or* /	Repeat previous search in same direction.
N	Repeat previous search in opposite direction.

Cursor and Screen Manipulation Commands

<ctrl>-R	Redraw the screen. (Used when session becomes confused.)
G	Move cursor (go) to bottom of file.
*n*G	Move cursor (go) to line *n*
b	Move to beginning of word.
E	Move to end of word.
0 [zero]	Move to beginning of line
$	Move to end of line.
n<arrow-key>	Move cursor *n* lines or characters.
H	Go to first line on screen.
M	Go to middle line on screen.
L	Go to last line on screen.
n^f	Move *n* screens forward (up).
n^d	Move *n* half-screens forward (up).
n^b	Move *n* screens backward (down).
n^u	Move *n* half-screens backward (down).
n^e	Scroll window down *n* lines.
n^y	Scroll window up *n* lines.

When arrow-keys don't work properly, these keys will:

k	cursor-up
j or <cr>	cursor-down
h	cursor-left
l or <sp>	cursor-right

File commands

ZZ	Save your work and quit vi. (**Be sure to press ***<shift>-Z* **and not ***<ctrl>-Z* **!**) (<ctrl>-Z stops the vi job without saving or quitting. If you do this, type *fg* to restore vi to active state, and then exit vi properly.)
:wq	Same as ZZ, but without the danger of confusion with <ctrl>-Z.
:cd <dir>	Changes editor's working directory.
:r<file>	Read file and insert it at cursor.
:r !<cmd>	Insert output of Unix command <cmd> at cursor.
:q	Quit vi without saving the file.
:q!	Forced quit vi without saving.
:w	Save without quitting.
:w<file>	Save as <file> without quitting.
:!*cmd*	Execute shell command without quitting vi.
:!bash	Spawn a shell without quitting vi. Exit from the shell to return to vi.

Setting vi Options

:set all Prints all option settings.
:set <option-name> or
:set <option-name>=<value> Sets option.
Setting effective for only duration of vi session. For semi-permanent option settings, set environment variable EXINIT to "set <option1> <option2> ..." in .bash_profile file. Initializations may also be placed in a .vimrc file in your home directory.

Options	**Default**	
autoindent (ai)	noai	Indent subsequent lines the same as current line. <ctrl>-D backspaces over indentation.
ignorecase (ic)	noic	Ignore upper/lower case in searches.
magic	nomagic	., &, ^, [, * are special in searches.
number (nu)	nonu	Display line numbers.
readonly (ro)	noro	Change file status to read-only.
showmatch (sm)	nosm	Show matching) or } as (or { is typed.
tabstop (ts)	ts=8	Sets tab width.
wrapmargin (wm)	wm=0	Automatically break lines at a space at least *n* characters from right edge of page.
wrapscan (ws)	ws	Searches wrap around end of file.

The vi Text Editor

Michael P. Conlon, Ph.D.
Slippery Rock University

February 1, 2016

Starting vi: vi <filename>
Starting vi after a crash (recover): vi -r <filename>

Modes

vi has two modes, *command (normal) mode* and *insert mode* .

Insert Mode

In insert mode, you can enter text, or use <backspace> to delete just-entered text on the same line. Some vi-compatible editors, such as El*vi*s and *vi*m, allow you to move the cursor while in insert mode, and even delete old text.

If you see strange characters on the screen when you use cursor keys, you are using the original *vi*. Press <esc> to leave insert mode, delete the funny characters, cursor to your desired location, and re-enter insert mode.

Entering insert mode:

i insert immediately before cursor.
a (append) insert immediately after cursor.
A (append) insert at the end of the current line.
o open a new line below and start inserting there.
O open a new line above and start inserting there.
R Replace characters under cursor as you type. Once you press <enter>, replacement stops, but insertion can continue.

Leaving insert mode: <esc>

vi and vim:

Vim is an updated version of vi; it can do everything that vi can do and more. The most noticeable difference is that in vim you can move the cursor around without leaving insert mode. It also allows you to delete any text (using the backspace key), not just the text you have just inserted, while in insert mode. Vim is a syntax-directed editor, and it understands all major programming and markup languages. Under Linux, you are probably using vim. When you move to a system with vi instead of vim, you *will* notice.