Linux Administration

Learning by example

vi cheat sheet

General Notes:

1. Before doing anything to a document, type the following command followed by a carriage return: :set showmode

2. VI is CaSe SEnsItiVe!!! So make sure Caps Lock is OFF.

Starting and Ending VI

Starting VI	
vi *filename*	Edits *filename*
vi -r *filename*	Edits last save version of *filename* after a crash
vi + n *filename*	Edits *filename* and places curser at line n
vi + *filename*	Edits *filename* and places curser on last line
vi +/*string filename*	Edits *filename* and places curser on first occurance of *string*
vi *filename file2* ...	Edits *filename*, then edits *file2* ... After the save, use :n
Ending VI	
ZZ or :wq or :x	Saves and exits VI
:w	Saves current file but doesn't exit
:w!	Saves current file overriding normal checks but doesn't exit
:w *file*	Saves current as *file* but doesn't exit
:w! *file*	Saves to *file* overriding normal checks but doesn't exit
:n,mw *file*	Saves lines n through m to *file*
:n,mw >>*file*	Saves lines n through m to the end of *file*
:q	Quits VI and may prompt if you need to save
:q!	Quits VI and without saving
:e!	Edits file discarding any unsaved changes (starts over)
:we!	Saves and continues to edit current file

Status

:.=	Shows current line number
:=	Shows number of lines in file
Control-G	Shows filename, current line number, total lines in file, and % of file location
l	Displays tab (^l) backslash (\) backspace (^H) newline ($) bell (^G) formfeed (^L^) of current line

Modes

Vi has two modes insertion mode and command mode. The editor begins in command mode, where the cursor movement and text deletion and pasting occur. Insertion mode begins upon entering an insertion or change command. [ESC] returns the editor to command mode (where you can quit, for example by typing :q!). Most commands execute as soon as you type them except for "colon" commands which execute when you press the ruturn key.

Inserting Text

i	Insert before cursor

Latest Articles

- Troubleshooting SYNs to LISTEN sockets dropped message from netstat
- Postfix as a SASL authenticated, TLS enabled Relay
- Updating Route53 records after EC2 instance restart
- Using LXC with OpenStack
- Deploying Prometheus for monitoring and stats collection
- Deploying MongoDB
- Deploying Apache Cassandra
- Deploying Elasticsearch cluster
- Simple Continuous Deployment System with Jenkins and Github
- Deploying services with Mesos, Marathon, Zookeeper and Docker
- Multitenant HA Redis on AWS
- Container Integration in systemd
- Fun with Linux Network Namespaces
- Deploying Apache Kafka and Apache Zookeeper
- Block device encryption with cryptsetup and LUKS
- Keepalived using unicast, track and notify scripts

I	Insert before line
a	Append after cursor
A	Append after line
o	Open a new line after current line
O	Open a new line before current line
r	Replace one character
R	Replace many characters
CTRL-v *char*	While inserting, ignores special meaning of char (e.g., for inserting characters like ESC and CTRL) until ESC is used
:r *file*	Reads *file* and inserts it after current line
:nr *file*	Reads *file* and inserts it after line n
CTRL-i or TAB	While inserting, inserts one shift width

Things to do while in Insert Mode:

CTRL-h or Backspace	While inserting, deletes previous character
CTRL-w	While inserting, deletes previous word
CTRL-x	While inserting, deletes to start of inserted text
CTRL-v	Take the next character literally. (i.e. To insert a Control-H, type Control-v Control-h)

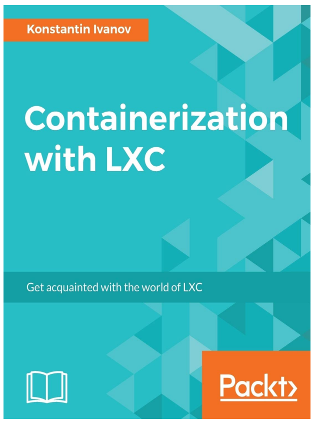
Motion

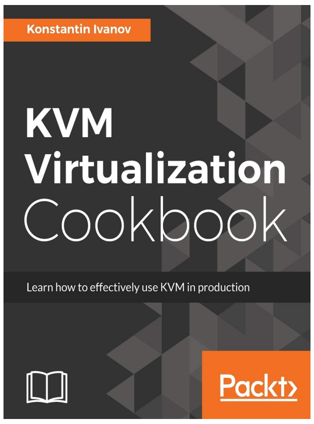
h	Move left
j	Move down
k	Move up
l	Move right
Arrow Keys	These do work, but they may be too slow on big files. Also may have unpredictable results when arrow keys are not mapped correctly in client.
w	Move to next word
W	Move to next blank delimited word
b	Move to the beginning of the word
B	Move to the beginning of blank delimted word
^	Moves to the first non-blank character in the current line
+ or	Moves to the first character in the next line
-	Moves to the first non-blank character in the previous line
e	Move to the end of the word
E	Move to the end of Blank delimited word
(	Move a sentence back
)	Move a sentence forward
{	Move a paragraph back
}	Move a paragraph forward
[[	Move a section back
]]	Move a section forward
0 or |	Move to the begining of the line
n|	Moves to the column n in the current line
$	Move to the end of the line
1G	Move to the first line of the file
G	Move to the last line of the file
nG	Move to nth line of the file
:n	Move to nth line of the file
fc	Move forward to c
Fc	Move back to c
H	Move to top of screen
nH	Moves to nth line from the top of the screen
M	Move to middle of screen
L	Move to botton of screen
nL	Moves to nth line from the bottom of the screen
Control -d	Move forward ½ screen
Control -f	Move forward one full screen
Control -u	Move backward ½ screen
Control -b	Move backward one full screen
CTRL-e	Moves screen up one line
CTRL-y	Moves screen down one line
CTRL-u	Moves screen up ½ page
CTRL-d	Moves screen down ½ page

- Deploying Highly Available MySQL with MHA and HAProxy
- Deploying HAProxy 1.5 from source
- Metrics visualisation and collection with Graphite, Grafana and python
- Deploying Highly Available NFS Server with DRBD and Heartbeat on Debian
- Injecting kernel modules in initrd.gz for the Debian Installer
- Creating an official Debian mirror with apt-mirror
- Creating secure LXC containers with virt-sandbox-service
- DROP versus REJECT a packet
- Diagnosing High CPU utilization and memory leaks
- Deploying OpenVZ Containers
- Connecting KVM or LXC to Open vSwitch

List of All Articles

My Publications:


Konstantin Ivanov
Containerization with LXC
Get acquainted with the world of LXC
Packt›


Konstantin Ivanov
KVM Virtualization Cookbook
Learn how to effectively use KVM in production
Packt›

CTRL-b	Moves screen up one page
CTRL-f	Moves screen down one page
CTRL-I	Redraws screen
z	z-carriage return makes the current line the top line on the page
nz	Makes the line n the top line on the page
z.	Makes the current line the middle line on the page
nz.	Makes the line n the middle line on the page
z-	Makes the current line the bottom line on the page
nz-	Makes the line n the bottom line on the page
%	Move to associated ( ), { }, [ ]

Deleting Text

Almost all deletion commands are performed by typing d followed by a motion. For example, dw deletes a word. A few other deletes are:

x	Delete character to the right of cursor
nx	Deletes n characters starting with current; omitting n deletes current character only
X	Delete character to the left of cursor
nX	Deletes previous n characters; omitting n deletes previous character only
D	Delete to the end of the line
d$	Deletes from the cursor to the end of the line
dd or :d	Delete current line
ndw	Deletes the next n words starting with current
ndb	Deletes the previous n words starting with current
ndd	Deletes n lines beginning with the current line
:n,md	Deletes lines n through m
d*Motion_cmd*	Deletes everything included in the Motion Command (e.g., dG would delete from current position to the end of the file, and d4 would delete to the end of the fourth sentence).
"np	Retrieves the last nth delete (last 9 deletes are kept in a buffer)
"1pu.u.	Scrolls through the delete buffer until the desired delete is retrieved (repeat u.)

Yanking Text

Like deletion, almost all yank commands are performed by typing y followed by a motion. For example, y$ yanks to the end of the line. Two other yank commands are:

yy	Yank the current line
:y	Yank the current line
nyy or nY	Places n lines in the buffer-copies
y*Motion_cmd*	Copies everything from the curser to the Motion Command (e.g., yG would copy from current position to the end of the file, and y4 would copy to the end of the fourth sentence)
"(a-z)nyy or "(a-z)ndd	Copies or cuts (deletes) n lines into a named buffer a through z; omitting n works on current line

Changing text

The change command is a deletion command that leaves the editor in insert mode. It is performed by typing c followed by a motion. For example cw changes a word. A few other change commands are:

C	Change to the end of the line
cc or S	Change the whole line until ESC is pressed
xp	Switches character at cursor with following character
s*text*	Substitutes text for the current character until ESC is used
cw*text*	Changes current word to text until ESC is used
C*text*	Changes rest of the current line to text until ESC is used
c*Motion_cmd*	Changes to text from current position to Motion Command until ESC is used
<< or >>	Shifts the line left or right (respectively) by one shift width (a tab)
n<< or n>>	Shifts n lines left or right (respectively) by one shift width (a tab)
<*Motion_cmd* or >*Motion_cmd*	Use with Motion Command to shift multiple lines left or right

Putting text

p	Put after the position or after the line
P	Put before the poition or before the line
"(a-z)p or "(a-z)P	Pastes text from a named buffer a through z after or before the current line

Buffers

Named buffers may be specified before any deletion, change, yank or put command. The general prefix has the form "c where c is any lowercase character. for example, "adw deletes a word into buffer a. It may thereafter be put back into text with an appropriate "ap.

Markers

Named markers may be set on any line in a file. Any lower case letter may be a marker name. Markers may also be used as limits for ranges.

mc	Set marker c on this line
`c	Go to beginning of marker c line.
'c	Go to first non-blank character of marker c line.

Search for strings

/*string*	Search forward for *string*
?*string*	Search back for *string*
n	Search for next instance of *string*
N	Search for previous instance of *string*
%	Searches to beginning of balancing ( ) [ ] or { }
f*c*	Searches forward in current line to *char*
F*c*	Searches backward in current line to *char*
t*c*	Searches forward in current line to character before char
Tchar	Searches backward in current line to character before char
?str	Finds in reverse for str
:set ic	Ignores case when searching
:set noic	Pays attention to case when searching
:n,ms/*str1*/*str2*/*opt*	Searches from n to m for *str1*; replaces *str1* to *str2*; using opt-opt can be g for global change, c to confirm change (y to acknowledge, to suppress), and p to print changed lines
&	Repeats last :s command
:g/*str*/*cmd*	Runs *cmd* on all lines that contain *str*
:g/*str1*/s/*str2*/*str3*/	Finds the line containing *str1*, replaces *str2* with *str3*
:v/*str*/*cmd*	Executes *cmd* on all lines that do not match *str*
,	Repeats, in reverse direction, last / or ? search command

Replace

The search and replace function is accomplished with the :s command. It is commonly used in combination with ranges or the :g command (below).

:s/*pattern*/*string*/*flags*	Replace *pattern* with *string* according to *flags*.
g	Flag - Replace all occurences of pattern
c	Flag - Confirm replaces.
&	Repeat last :s command

Regular Expressions

. (dot)	Any single character except newline
*	zero or more occurances of any character
[...]	Any single character specified in the set
[^...]	Any single character not specified in the set
\<	Matches beginning of word
\>	Matches end of word
^	Anchor - beginning of the line
$	Anchor - end of line
\<	Anchor - begining of word
\>	Anchor - end of word
\(...\)	Grouping - usually used to group conditions
\n	Contents of nth grouping
\	Escapes the meaning of the next character (e.g., \$ allows you to search for $)
\\	Escapes the \ character

[...] - Set Examples

[A-Z]	The SET from Capital A to Capital Z
[a-z]	The SET from lowercase a to lowercase z
[0-9]	The SET from 0 to 9 (All numerals)

[./=+]	The SET containing . (dot), / (slash), =, and +
[-A-F]	The SET from Capital A to Capital F and the dash (dashes must be specified first)
[0-9 A-Z]	The SET containing all capital letters and digits and a space
[A-Z][a-zA-Z]	In the first position, the SET from Capital A to Capital Z In the second character position, the SET containing all letters
[a-z]{m}	Look for *m* occurances of the SET from lowercase a to lowercase z
[a-z]{m,n}	Look for at least m occurances, but no more than n occurances of the SET from lowercase a to lowercase z

Regular Expression Examples

/Hello/	Matches if the line contains the value Hello
/^TEST$/	Matches if the line contains TEST by itself
/^[a-zA-Z]/	Matches if the line starts with any letter
/^[a-z].*/	Matches if the first character of the line is a-z and there is at least one more of any character following it
/2134$/	Matches if line ends with 2134
/\(21|35\)/	Matches is the line contains 21 or 35 Note the use of ( ) with the pipe symbol to specify the 'or' condition
/[0-9]*/	Matches if there are zero or more numbers in the line
/^[^#]/	Matches if the first character is not a # in the line
Notes: 1. Regular expressions are case sensitive 2. Regular expressions are to be used where *pattern* is specified	

Counts

Nearly every command may be preceded by a number that specifies how many times it is to be performed. For example, 5dw will delete 5 words and 3fe will move the cursor forward to the 3rd occurence of the letter e. Even insertions may be repeated conveniently with this method, say to insert the same line 100 times.

Ranges

Ranges may precede most "colon" commands and cause them to be executed on a line or lines. For example :3,7d would delete lines 3-7. Ranges are commonly combined with the :s command to perform a replacement on several lines, as with :.,$s/pattern/string/g to make a replacement from the current line to the end of the file.

:n,m	Range - Lines n-m
:.	Range - Current line
:$	Range - Last line
:'c	Range - Marker c
:%	Range - All lines in file
:g/*pattern*/	Range - All lines that contain *pattern*

Shell Functions

:! cmd	Executes shell command cmd; you can add these special characters to indicate:% name of current file# name of last file edited
!! cmd	Executes shell command cmd, places output in file starting at current line
:!!	Executes last shell command
:r! cmd	Reads and inserts output from cmd
:f file	Renames current file to file
:w !cmd	Sends currently edited file to cmd as standard input and execute cmd
:cd dir	Changes current working directory to dir
:sh	Starts a sub-shell (CTRL-d returns to editor)
:so file	Reads and executes commands in file (file is a shell script)
!*Motion_cmd*	Sends text from current position to Motion Command to shell command *cmd*
!}sort	Sorts from current position to end of paragraph and replaces text with sorted text

Files

:w *file*	Write to *file*
:r *file*	Read *file* in after line
:n	Go to next file
:p	Go to previous file
:e *file*	Edit *file*
!!*program*	Replace line with output from *program*

VI Settings

--noto

Note: Options given are default. To change them, enter type :set option to turn them on or :set no*optioni* to turn them off.To make them execute every time you open VI, create a file in your HOME directory called .exrc and type the options without the colon (:) preceding the option

Set	Default	Description	
:set ai	noai	Turns on auto indentation	
:set all	--	Prints all options to the screen	
:set ap	aw	Prints line after d c J m :s t u commands	
:set aw	noaw	Automatic write on :n ! e# ^^ :rew ^} :tag	
:set bf	nobf	Discards control characters from input	
:set dir=*tmp*	dir = /tmp	Sets *tmp* to directory or buffer file	
:set eb	noed	Precedes error messages with a bell	
:set ed	noed	Precedes error messages with a bell	
:set ht=	ht = 8	Sets terminal hardware tabs	
:set ic	noic	Ignores case when searching	
:set lisp	nolisp	Modifies brackets for Lisp compatibility.	
:set list	nolist	Shows tabs (^l) and end of line ($)	
:set magic	magic	Allows pattern matching with special characters	
:set mesg	mesg	Allows others to send messages	
:set no*option*		Turns off *option*	
:set nu	nonu	Shows line numbers	
:set opt	opt	Speeds output; eliminates automatic RETURN	
:set para=	para = LIlPLPPPQPbpP	macro names that start paragraphs for { and } operators	
:set prompt	prompt	Prompts for command input with :	
:set re	nore	Simulates smart terminal on dumb terminal	
:set remap	remap	Accept macros within macros	
:set report	noreport	Indicates largest size of changes reported on status line	
:set ro	noro	Changes file type to "read only"	
:set scroll=n	scroll = 11	set n lines for CTRL-d and z	
:set sh=*shell_path*	sh = /bin/sh	set shell escape (default is /bin/sh) to *shell_path*	
:set showmode	nosm	Indicates input or replace mode at bottom	
:set slow	slow	Pospone display updates during inserts	
:set sm	nosm	Show matching { or (as) or } is typed	
:set sw=n	sw = 8	Sets shift width to n characters	
:set tags=x	tags = /usr/lib/tags	Path for files checked for tags (current directory included in default)	
:set term	$TERM	Prints terminal type	
:set terse	noterse	Shorten messages with terse	
:set timeout		Eliminates one-second time limit for macros	
:set tl=n	tl = 0	Sets significance of tags beyond n characters (0 means all)	
:set ts=n	ts = 8	Sets tab stops to n for text input	
:set wa	nowa	Inhibits normal checks before write commands	
:set warn	warn		Warns "no write since last change"
:set window=n	window = n	Sets number of lines in a text window to n	
:set wm=n	wm = 0	Sets automatic wraparound n spaces from right margin.	
:set ws	ws	Sets automatic wraparound n spaces from right margin.	

Key Mapping

NOTE: Map allows you to define strings of VI commands. If you create a file called ".exrc" in your home directory, any map or set command you place inside this file will be executed every time you run VI. To imbed control characters like ESC in the macro, you need to precede them with CTRL-v. If you need to include quotes ("), precede them with a \ (backslash). Unused keys in vi are: K V g q v * = and the function keys. Example (The actual VI commands are in blue): :map v /I CTRL-v ESC dwiYou CTRL-v ESC ESC Description: When v is pressed, search for "I" (/I ESC), delete word (dw), and insert "You" (iYou ESC). CTRL-v allows ESC to be inserted

:map *key cmd_seq*	Defines *key* to run *cmd_seq* when pressed
:map	Displays all created macros on status line

:unmap key	Removes macro definition for key
:ab *str string*	When *str* is input, replaces it with *string*
:ab	Displays all abbreviations
:una *str*	Unabbreviates *str*

Other

~	Toggle upper and lower case
J	Join lines
nJ	Joins the next n lines together; omitting n joins the beginning of the next line to the end of the current line
.	Repeat last text-changing command
u	Undo last change (Note: u in combination with . can allow multiple levels of undo in some versions)
U	Undo all changes to line
;	Repeats last f F t or T search command
:N or :E	You can open up a new split-screen window in (n)vi and then use ^w to switch between the two.

Newer Post Home Older Post