

Filter motion with shell

Use :earlier and :later to quickly jump backward and forward in a file's history.

:read

Read external program output into current file

ľ	:h up	l. o-down-m	otions												
					ts	ts sw	w sts et	tabstop	ts	Colum	ns per	tabstop)		
		use spaces only		n	n	n c	on	on	shiftwidth	sw	Colum	ns per	<<		
		use ta	bs onl	у	n	n	0	off		softtabstop	sts	Space	s per t	ab	
		Set n to desired tab width			h (d	efaul	t 8)		expandtab	et	(Tab)	inser	ts space	es	
		MIX	ING	TA	BS	Α	ND		SP	ACES :	(ti	RIG nat mea		OUT	
		: 1	e.	tal	b		eplac bsto			s with space	es ac	cording	to curr	rent	
		filefor	mat	ff		Tr	y cha	angi	ng t	his if your li	ne-er	ndings a	re mes	sed up	
		list				Di	ispla	y wh	ites	pace visibly	acco	rding to		ars t-right-m	not
	cha	next racter	1	end of word	е		ginnin of nex	ď.	N	end of WORD	b	eginning of next WORD	W	end of line	4

р	paste after cursor	Р	paste before cursor	^[return to Normal mode
u	undo	^r	redo		repeat
gf	find file under cursor in path and jump to it	dd	delete current line	уу	yank current line
x	delete character after cursor	%	jump to matching paren	r	replace char under cursor
<i>n</i> G	jump to line n	^0	jump back	^i jump-motions	jump forward
ZZ	center screen on cursor	zt	align top of screen with cursor	zb	align bottom of screen with cursor
==	auto-indent current line	<<	shift current line left by shiftwidth	>>	shift current line right by shiftwidth
	u gf x nG	U undo gf info file under cursor in path and jump to it. X detect character after cursor GG jump to line n ZZ center screen on cursor	undo ^r gf infer fle under cursor in path and jump to it. X delete character after cursor % nG jump to line n ^O zz carter screen on zt	undo undo nr gf find file under cursor in path and jump to it. X delete character after cursor % jump to line n yump to line n zz center screen on cursor zt dign top of screen with cursor structured line biff rurser li	undo vr redo ff ind file under cursor in path and jump to it. X delete character after cursor % jump to line n yum to line n very redo ve

				:h insert.tx
7		COOL	INSER	T MODE STUFF
1	^w	delete word before cursor	^u	delete line before cursor
	^r <i>r</i>	insert the contents of register r	^r=	use the expression register (try *r=5+10)
	^t	increase line indent by shiftwidth	^d	decrease line indent by shiftwidth
	^x^1	line completion	^n	find next completion suggestic according to complete
				:h cmdline.tx

Using ^ [to return to Normal mode lets you keep your fingers on the home row. It's even easier if you map Caps Lock to Control!

			:h cm	dline.txt			
	СОМ	MAND-LI	NE MODE C	NLY			
Normal mode cndwin	insert word under cursor cmdline-editing	^r^w	completion suggestions cmdline-completion	^d			
	Put encremap XX (C-R) = expand('K:h').'/'<0Rs in your vinze so you can type XX in Command-line mode to refer to the directory of the current file, regardless of page.						
Supply % as a range to t	the :substitu	te command to r	un it on every line in	the file.			
:%s/Scribbl/Desig	n/	"Scribbled" ->	"Designed"				
Specify the "g" flag to a	pply the substit	ution to every ma	tch on a line.				
:s/[dla]//g		"badly" -> "by"	:h s_fla	gs, :h /[]			
Vim supports many regular expression features.							
:s/k/ax/		"Mook" -> "Ma	x" :h usr	_27, :h /.			
Use \ instead of . if you want to search across multiple lines.							
:%s/heat*Bungle	e/anto/	"Cheatsheet\nE	Bungler" -> "Cantor"	:h /\			
Special escapes can be	used to change	the case of substi	tutions.				
:s_\(f\)_\U\1\E.	_	"foobar" -> "FO	OObar" :h sub-repla	ce-special			
Use : global to perform a command on matching lines.							
:g/foobar/delete		Delete all lines	containing "foobar"				
If your pattern contains	slashes, just us	e a different char	acter as your delimit	er.			
:s_Data/Lore_Brent Spiner_							

"10 25" -> "21 36"

Use $\backslash =$ to evaluate expressions with replacement groups.

:s_\d_\=submatch(0) + 1_g

	:help
:h <i>cmd</i>	Normal mode <i>cmd</i> help
:h i_ <i>cmd</i>	Insert mode <i>cmd</i> help
:h v_ <i>cmd</i>	Visual mode <i>cmd</i> help
:h c_ <i>cmd</i>	Command-line editing cmd help
:h : <i>cmd</i>	Command-line <i>cmd</i> help
:h 'option'	Option help
:helpgrep	Search through all help docs!

	vim
	:h tags-and-searches
^]	Jump to tag under cursor, including [tags] in help files
^t	Jump back up the tag-list
g^]	Jump to tag if it's the only match; else list matching tags

			:h keycodes
<cr></cr>	^m	\r	Enter
<tab></tab>	^i	\t	Tab
<c-n></c-n>	^n		Ctrl-n
<m-<i>n></m-<i>			Alt-n
<esc></esc>	^[Escape
<bs></bs>	^h	\b	Backspace
			Delete

7 words -- word-actions http://www.vimcheatsheet.com 1 WORD

	:h option
:set opt?	View current value of opt
:set no <i>opt</i>	Turn off flag opt
:set opt	Turn on flag opt
:set opt=val	Overwrite value of opt
:set opt+=val	Append to value of opt
:echo &opt	Access opt as a variable

	:h buffers
:ls	List all open files
:b path	Jump to unique file matching <i>path</i> . Use <tab></tab> to scroll through available completions!
:b <i>n</i>	Jump to file <i>n</i> , number from first column of :ls
:bnext	Jump to next file
:bprev	Jump to previous file
:bdelete	Remove file from the buffer list
:edit	Open a file for editing
:enew	Open a blank new file for editing
	:h windows
:split	Split current window horizontally
:vsplit	Split current window vertically
^w hjkl	Move cursor to window left, below, above or to the right of the current window
^w HJKL	Move current window to left, bottom, top, or right of screen
^w r	Rotate windows clockwise
^w +-<>	Increase/decrease current window height/width
^w T	Move current window to a new tab
:only	Close all windows except current window
: bufdo	Execute a command in each open file

hidden	hid	Lets you switch buffers without saving
laststatus	Is	Show status line never (0), always (2) or with 2+ windows (1)
hisearch	hls	Highlight search matches. Also see 'highlight'
number	nu	Show line numbers
showcmd	sc	Show commands as you type them
ruler	ru	Show line and column number of the cursor
backspace	bs	Set to '2' to make backspace work like sane editors
wrap		Control line wrapping
background	bg	Set to 'dark' if you have a dark color scheme

include delimiters or surrounding whitespace. For example, d1 (i will change "(foo)" into "()", but da(i will delete the parentheses as well.

Use : map to

view all current

mappings. Read

keys for a guide on which keys are best for your

custom key

own custom mappings. Get

used to Vim's help system it's a fantastic

Use a instead of i when

beginning text-

object motions to

REGISTERS are CLIPBOARDS

All commands that delete, copy, or paste text use registers. To change which register is used by a command, type the register before the command. The default register is called "the unnamed register", and it is invoked with a pair of double-quotes (""). Typing @d or | ""yy. Think of the first " as a short way of saying "register", so " is pronounced "register", and " is "register a".

of saying "register", s	o iii is pronounced	"register "", and "a, "register a".
:regis	ters	View all current registers
:echo	@ r	Access register r as a variable
"/	Last search pattern register	Contains the last pattern you searched for
"_	The black hole register	Use this to delete without clobbering any register ("_dd)
"0	Last yank register	Contains the last text you yanked
"1	Last big delete register	Contains the last line(s) you deleted
"2-"9	Big delete register stack	Every time "1 is written to, its content is pushed to "2, then "2 to "3, and so on
"-	Small delete register	Contains the last text you deleted within a single line
"+	System clipboard	If the OS integration gods smile upon you, this register reads and writes to your system clipboard.
"a-"z	Named registers	26 registers for you to play with
"A-"Z	Append registers	Using upper-case to refer to a register will append to it rather than overwrite it
q r	Record	Record into register ${m r}$. Stop recording by hitting ${f q}$ again
@ r	Playback	Execute the contents of register <i>r</i>
@@	Repeat last playback	Repeat the last $@r$, this is particularly useful with a count
	: 0000	$\label{eq:vimone-liner} \begin{tabular}{ll} vim one-liner used to sort the list of names by length: $$ 'g/^/let $$ = len(getline(".")) normal "b"a " sort n :g//normal $$ defined by $$ $$ (and the line of the line o$