

Vim tips

November 21, 2009

Contents

1	Text manipulation	4
1.1	Searching	4
1.1.1	Search for declaration of subroutine/function under cursor	4
1.1.2	Search for visually highlighted text	4
1.2	Substitution	4
1.2.1	Filter all form elements into paste register	5
1.2.2	Substitute within substitution	5
1.2.3	Substituting a visual area	6
1.3	Global command display	6
1.4	Global combined with substitute (power editing)	7
1.5	Changing case	7
1.6	Reformatting text	7
1.7	Deletion without destroying buffer	7
1.8	Essential	7
2	File Manipulation	8
2.1	Exploring	8
2.2	Opening files & other tricks	8
2.3	Multiple files management	9
2.4	File-name manipulation	9
2.5	Command over multiple files	9
2.6	Sessions (set of files)	10
2.7	Modelines	10
2.7.1	Creating your own GUI Toolbar entry	10
2.8	Markers & moving about	10
2.9	Editing/moving within insert mode	11
2.10	Abbreviations & maps	11
2.10.1	Display RGB colour under the cursor eg #445588	11

3 Registers	11
3.1 List your registers	11
3.2 Appending to registers	12
3.3 Using a register as a map (preload registers in .vimrc)	12
3.4 Redirection & paste register	12
3.4.1 Copy full path name	12
3.5 Useful tricks	12
 4 Advanced	 13
4.1 Command line tricks	13
4.2 External programs	13
4.3 Recording	13
4.3.1 Operating a Recording on a Visual BLOCK	14
4.4 Quick jumping between splits	14
4.5 Visual mode basics	14
4.6 vimrc essentials	15
4.6.1 Launching IE	15
4.6.2 Ftping from vim	15
4.6.3 Autocmd	15
4.7 Conventional shifting and indenting	15
4.8 Pulling objects onto command/search line	15
4.9 Capturing output of current script	16
4.10 Inserting DOS carriage returns	16
4.11 Perform an action on a particular file or file type	16
4.12 Inserting line number	16
4.13 Numbering lines	16
4.14 Advanced incrementing	17
4.14.1 Create list starting from 223 incrementing by 5 between markers a,b	17
4.14.2 Create a map for INC	17
4.15 Digraphs (non alpha-numeric)	17
4.16 Complex vim	17
4.17 Syntax highlighting	17
4.18 Preventions and security	18
4.19 Taglist	18
4.20 Folding	18
4.21 Renaming files	18
4.22 Reproducing lines	18
4.23 Reading MS-Word documents	19

<i>CONTENTS</i>	<i>CONTENTS</i>
4.24 Random functions	19
4.24.1 Save word under cursor to a file	19
4.24.2 Delete duplicate lines	19
4.24.3 Columnise a CSV file for display	19
4.25 Miscallaenous commands	19
4.26 Vim traps	20
4.27 Help	20
5 Fun	20
5.1 Contact	21
5.1.1 Author (David Rayner)	21
5.1.2 Maintainer (Gavin Gilmour)	21
5.1.3 Source	21
5.2 Links	21

1 Text manipulation

1.1 Searching

Command	Description
/joe/e	cursor set to end of match
/joe/e+1	cursor set to end of match plus 1
/joe/s-2	cursor set to start of match minus 2
/^joe.*fred.*bill/	normal
/^[A-J]\+/	search for lines beginning with one or more A-J
/begin\.*end	search over possible multiple lines
/fred\s*joe/i	any whitespace including newline
/fred\ joe	search for fred or joe
/.fred&.*joe	search for fred and joe in any order
/\<fred\>/i	search for fred but not alfred or frederick
/\<\d\d\d\d\>	search for exactly 4 digit numbers
/\D\d\d\d\d\>	search for exactly 4 digit numbers
/\<\d\{4\}\>	same thing
/\([^\0-9]\ ^\)\%.%%	search for absence of a digit or beginning of line
/^\n\{3\}	find 3 empty lines
/\((fred\).*\)(joe\).*\2.*\1	using rexp memory in a search
/^\([^\,]*\)\{8\}	using rexp memory in a search
/<\zs[^\>]*\ze>	search for tag contents, ignoring chevrons (:h /\zs)
/<\@<=[^\>]*>\@=	search for tag contents, ignoring chevrons
/<\@<=[^\>]*>\@=	search for tags across possible multiple lines
/i!-p{-}-i	search for multiple line comments
/fred\s*joe/i	any whitespace including newline (\s)
/bugs\(\.\.\)*bunny	bugs followed by bunny anywhere in file
:h \-	help
:bufdo /searchstr/	multiple file search (use :rewind to recommence search)
:bufdo %s/searchstr/&/gic	multiple file search better but cheating (n, then a to stop)
?http://www.vim.org/	search backwards for a URL without backslashing
/\c\v([^aeiou]&\a){4}	search for 4 consecutive consonants

1.1.1 Search for declaration of subroutine/function under cursor

```
:nmap gx yiw/\(sub\<bar>function\)\s\+<C-R>"<CR>
```

1.1.2 Search for visually highlighted text

```
:vmap <silent> // y/<C-R>"<CR>
```

```
:vmap <silent> // y/<C-R>=escape("@", '\./.*~\[')<CR><CR> (with spec chars)
```

1.2 Substitution

```
| :%s/fred/joe/igc
```

```
| general substitute command
```

1.2 Substitution

1 TEXT MANIPULATION

<code>:%s/\r//g</code>	delete dos returns ^M
<code>:%s/\r\r/g</code>	turn dos returns ^M into real returns (fixes joined lines)
<code>:%s=*\$==</code>	delete end of line blanks
<code>:%s= \+\$==</code>	same as above
<code>:%s#\s*\r\?\$##</code>	clean both trailing spaces AND dos returns
<code>:%s#\s*\r*\$##</code>	same thing deleting empty lines
<code>:%s/^\n\{3}\n/</code>	delete blocks of 3 empty lines
<code>:%s/^\n\+/\r/</code>	compressing empty lines
<code>:%s#<[^>]\+>##g</code>	delete html tags, leave text
<code>:a,bg/fred/s/dick/joe/igc</code>	VERY USEFUL
<code>:%s= [^]\+\$=&&=</code>	duplicate end column
<code>:%s= \f\+\$=&&=</code>	same as above
<code>:%s= \S\+\$=&&=</code>	usually the same
<code>:s/(.*)\:(.*)\2 : \1/</code>	reverse fields separated by :
<code>:%s/^\(.*\)\n\1\$/1/</code>	delete duplicate lines
<code>:%s/^\{-}\pdf/new.pdf/</code>	delete to 1st pdf only
<code>:%s#\<[zy]\?tbl-[a-z-]\+>#\L&#gc</code>	lowercase with optional leading characters
<code>:%s///</code>	delete possibly multi-line comments
<code>:help /\{-}</code>	help non-greedy
<code>:s/fred/a/g</code>	sub "fred" with contents of register "a"
<code>:s/fred/\=@a/g</code>	better alternative as register not displayed
<code>:%s/\f\+\.gif\>/\r&\r/g v/\.gif\$/d %s/gif/jpg/</code>	multiple commands on one line
<code>:%s/a/but/gie :update :next</code>	then use @: to repeat
<code>:%s/suck\ buck/loopy/gc</code>	ORing (must break pipe)
<code>:s/_date_/\=strftime("%c")/</code>	insert datestring
<code>:%s:\(\(w\+s\+)\)\{2}\)str1:\1str2:</code>	working with columns sub any str1 in col3
<code>:%s:\(\(w\+)\)\(.*s\+)\)\(w\+)\\$:\3\2\1:</code>	swapping first and last column (4 columns)
<code>:%s/d\+/\=(submatch(0)-3)/</code>	decrement numbers by 3
<code>:g/loc\ function/s/d/\=submatch(0)+6/</code>	increment numbers by 6 on certain lines only
<code>:%s#txtdev\zs\d#\=submatch(0)+1#g</code>	better version of above
<code>:%s/\(gg\)\@<=\d\+/\=submatch(0)+6/</code>	increment only numbers gg\d\d by 6 (another way)
<code>:let i=10 'a,bg/Abc/s/yy/\=i/ let i=i+1</code>	convert yy to 10,11,12 etc
<code>:let i=10 'a,bg/Abc/s/xx\zsy\ze/\=i/ let i=i+1</code>	convert xxyy to xx11,xx12,xx13 (more precise)
<code>:%s/"\([^\+]\+.\)*\zsxx\1/</code>	find replacement text, use \zs to simplify substitute
<code>:nmap z :%s#\<=expand(" ")>\>#</code>	pull word under cursor into LHS of a substitute
<code>:vmap z :%s/\<*\>/</code>	pull visually highlighted text into LHS of a substitute

1.2.1 Filter all form elements into paste register

```
:redir @*|sil exec 'g#\<(\input\|select\|textarea\|/\=form\)\>#p'|redir END
:nmap ,z :redir @*|sil exec 'g@\<(\input\|select\|textarea\|/\=form\)\>@p'|redir END
```

1.2.2 Substutue within substitiuon

<code>:%s,\(all/.*\)\@<=/,~,g</code>	replace all / with - AFTER "all/"
<code>:s#all/\zs.*#\=substitute(submatch(0), '/', '-', 'g')#</code>	same thing
<code>:s#all/#&^M# s#/#-#g ~j!</code>	sub by splitting line, re-joining
<code>:%s/.*\/=cp '.submatch(0).' all/'.substitute(submatch(0),'/', '-', 'g')/</code>	sub inside sub

1.2.3 Substituting a visual area

<pre> :':<,'>s/Emacs/Vim/g gv </pre>	<pre> remember you DONT type the '<.>' re-select the previous visual area (ULTRA) </pre>
--	--

1.3 Global command display

<pre> :g/gladiolli/# :g/fred.*joe.*dick/ :g/\<fred\>/ :g/^\s*/d :g!/^\dd/d :v/^\dd/d :g/fred/,/joe/d :g/-----/-10,.d :g/{/ ,/}/- s/\n\+/\r/g :v/S/d :v./././-j :g/^\\$/././-j :g/<input\ <form/p :g/^/put_ :g/^/m0 :'a,'b/^/m'b :g/^/t. :g/fred/t\$:g/stage/t'a :g/\(^I[^I]*\)\{80\}/d :'a,'bg/somestr/co/otherstr/ :'a,'bg/str1/s/str1/&&&/mo/str2/ :%norm jdd :.,\$g/^\d/exe "norm! \" :'a,'bg/\d\+ /norm! ^A :g/fred/y A :g/fred/y A :let @*=@a :let @a=" g/Barratt/y A :let @*=@a :'a,'b g/^Error/ . w >> errors.txt :g/./yank put -1s/'/' /g s/.*/Print '&'/ :g/^\MARK\$/r tmp.ex -d :g/z#.5 :g/z#.5 echo "======" :g/ /norm 2f r* :nmap :redir @a:g//:redir END:new:put! a </pre>	<pre> display with line numbers (YOU WANT THIS!) display all lines fred,joe & dick display all lines fred but not freddy delete all blank lines delete lines not containing string delete lines not containing string not line based (very powerfull) delete string & 10 previous lines delete empty lines but only between {...} delete empty lines (both types) compress empty lines compress empty lines ORing double space file (pu = put) reverse file (m = move) reverse a section a to b duplicate every line copy lines matching fred to EOF copy lines matching stage to marker a delete all lines containing at least 80 tabs match all lines containing "somestr" between markers a & b as above but also do a substitution delete every other line increment numbers increment numbers append all lines fred to register a (empty reg a first with qaq.) put into paste buffer write out to errors.txt duplicate every line in a file wrap a print " around each duplicate replace string with contents of a file, -d deletes the "mark" display with context display beautifully replace 2nd with a star send output of previous global command to a new window </pre>
---	--

1.4 Global combined with substitute (power editing)

:’a,’bg/fred/s/joe/susan/gic	can use memory to extend matching
:g/fred/,/joe/s/fred/joe/gic	non-line based (ultra)
:/fred;/joe/-2,/sid/+3s/sally/alley/gIC	find fred before beginning search for joe

1.5 Changing case

guu	lowercase line
gUU	uppercase line
Vu	lowercase line
VU	uppercase line
g~~	flip case line
vEU	upper case word
vE~	flip case word
gggUG	lowercase entire file
vmap ,c :s/<\(. \)\(\k*\)\>/\u\1\L\2/g	titlise visually selected text (map for .vimrc)
:%s/[!?!]_s\+_a/_U&_E/g	uppercase first letter of sentences
g<C-G>	count words in text file

1.6 Reformatting text

gq}	format a paragraph
gqap	format a paragraph
ggVGgq	reformat entire file
Vgq	current line
:s/.\{,69\};\s*\ .\{,69\}\s\+/\&\r/g	break lines at 70 chars, if possible after a ‘;

1.7 Deletion without destroying buffer

“_d	what you’ve ALWAYS wanted
“_dw	delete word (use blackhole)

1.8 Essential

* # g* g#	find word under cursor () (forwards/backwards)
%	match brackets and tags {}, [], (), etc.

.	repeat last modification
@:	repeat last : command (then @@)
<C-N><C-P>	word completion in insert mode
<C-X><C-L>	line complete SUPER USEFUL
/<C-R><C-W>	pull onto search/command line
/<C-R><C-A>	pull onto search/command line
:set ignorecase	you nearly always want this
:syntax on	colour syntax in perl, HTML, PHP etc.
:h regexp	list all help topics containing regexp (TAB to step through list)
:nmap ,s :source \$VIM/_vimrc	read from vimrc
:nmap ,v :e \$VIM/_vimrc	open and edit local vimrc
:vmap sb "zdi<C-R>z<ESC>	wrap around VISUALLY selected text
:vmap st "zdi<?= <C-R>z ?><ESC>	wrap <?= ?> around VISUALLY selected text

2 File Manipulation

2.1 Exploring

:Explore)	file explorer (note: capital E)
:Sex(plore)	file explorer in split window
:ls	list of buffers
:cd ..	move to parent directory
:args	list of files
:lcd %:p:h	change to directory of current file
:autocmd BufEnter * lcd %:p:h	change to directory of current file automatically ¹ (put in _vimrc)
\be	buffer explorer list of buffers
\bs	buffer explorer (split window)

2.2 Opening files & other tricks

gf	open file name under cursor (SUPER)
:nnoremap gF :view	open file under cursor, create if necessary
ga	display hex,ascii value of char under cursor
ggVGg?	rot13 whole file
ggg?G	rot13 whole file (quicker for large file)
:8 normal VGg?	rot13 from line 8
:normal 10GVGg?	rot13 from line 8
,	increment, decrement number under cursor
=5*5	insert 25 into text (mini-calculator)
:e main_	tab completes
main_	include NAME of file in text (insert mode)

¹Script required: bufexplorer.vim http://www.vim.org/script.php?script_id=42

2.3 Multiple files management

:bn	goto next buffer
:bp	goto previous buffer
:wn	save file and move to next (super)
:wp	save file and move to previous
:bd	remove file from buffer list (super)
:bun	buffer unload (remove window but not from list)
:badd file.c	file from buffer list
:b 3	go to buffer 3
:b main	go to buffer with main in name eg main.c (ultra)
:sav php.html	save current file as php.html and “move” to php.html
:sav! %<.bak	save current file to alternative extension (old way)
:sav! %:r.cfm	save current file to alternative extension
:sav %:s/fred/joe/	do a substitute on file name
:sav %:s/fred/joe/:r.bak2	do a substitute on file name & ext.
:!mv % %:r.bak	rename current file (DOS use rename or del)
:e!	return to unmodified file
:w c:/aaa/%	save file elsewhere
:e #	edit alternative file (also cntrl-^)
:rew	return to beginning of edited files list (:args)
:brew	buffer rewind
:sp fred.txt	open fred.txt into a split
:sball:sb	split all buffers (super)
:scrollbind	in each split window
:map <F5> :ls<CR>:e #	pressing F5 lists all buffers, just type number
:set hidden	allows to change buffer w/o saving current buffer

2.4 File-name manipulation

:h filename-modifiers	help
:w %	write to current file name
:w %:r.cfm	change file extension to .cfm
:!echo %:p	full path & file name
:!echo %:p:h	full path only
<C-R>%	insert filename (insert mode)
“%p	insert filename (normal mode)
/<C-R>%	search for file name in text

2.5 Command over multiple files

:argdo %s/foo/bar/e	operate on all files in :args
---------------------	-------------------------------

:bufdo %s/foo/bar/e	
:windo %s/foo/bar/e	
:argdo exe '%!sort' w!	include an external command

2.6 Sessions (set of files)

gvim file1.c file2.c lib/lib.h lib/lib2.h	load files for “session”
:mksession	create a session file (default session.vim)
gvim -S Session.vim	reload all files

2.7 Modelines

vim:noai:ts=2:sw=4:readonly:	makes readonly
vim:ft=html:	says use HTML syntax highlighting
:h modeline	help with modelines

2.7.1 Creating your own GUI Toolbar entry

```
amenu Modeline.Insert\ at\ VIMt\ modeline <Esc><Esc>ggOvim:ff=unix ts=4 ss=4
<CR>vim60:fdm=marker<Esc>gg
```

(All on one line!)

2.8 Markers & moving about

’.	jump to last modification line (SUPER)
‘.	jump to exact spot in last modification line
g;	cycle through recent changes (oldest first) ²
g,	reverse direction ³
:changes	show entire list of changes
:h changelist	help for above
<C-O>	retrace your movements in file (starting from most recent)
<C-I>	retrace your movements in file (reverse direction)
:ju(mps)	list of your movements
:help jump-motions	explains jump motions
:history	list of all your commands
:his c	commandline history
:his s	search history

²(new in vim 6.3)

³(new in vim 6.3)

q/	search history window (puts you in full edit mode)	
q:	commandline history window (puts you in full edit mode)	
:	history Window	

2.9 Editing/moving within insert mode

<C-U>	delete all entered	
<C-W>	delete last word	
<HOME><END>	beginning/end of line	
<C-LEFTARROW><C-RIGHTARROW>	jump one word backwards/forwards	
<C-X><C-E>,<C-X><C-Y>	scroll while staying put in insert	

2.10 Abbreviations & maps

:map <f7> :’a,’bw! c:/aaa/x	
:map <f8> :r c:/aaa/x	
:map <f11> :.w! c:/aaa/xr<CR>	
:map <f12> :r c:/aaa/xr<CR>	
:ab php	list of abbreviations beginning php
:map ,	list of maps beginning ,
set wak=no	allow use of F10 for win32 mapping (:h winaltkeys)
<CR>	enter
<ESC>	escape
<BACKSPACE>	backspace
<LEADER>	backslash
<BAR>	
<SILENT>	execute quietly
iab phpdb exit(" <hr>Debug <C-R>a ");	yank all variables into register a

2.10.1 Display RGB colour under the cursor eg #445588

```
:nmap <leader>c :hi Normal guibg=#<c-r>=expand("<cword>")<cr><cr>
```

3 Registers

3.1 List your registers

:reg	display contents of all registers	
:reg a	display content of individual registers	
"1p....	retrieve numeric registers one by one	

<code>:let @y='yy@'</code>	pre-loading registers (put in .vimrc)
<code>qqq</code>	empty register "q"
<code>:let @a=@_</code>	clear register a
<code>:let @a=""</code>	clear register a
<code>:let @*=@a</code>	copy register a to paste buffer
<code>map <F11> "qyy:let @q=@q."zzz"</code>	

3.2 Appending to registers

Yank 5 lines into "a" then add a further 5

1. "a5yy
2. 10j
3. "A5yy

3.3 Using a register as a map (preload registers in .vimrc)

```
| :let @m=":'a,'bs/"
| :let @s=":%!sort -u" |
```

3.4 Redirection & paste register

<code>:redir @*</code>	redirect commands to paste buffer
<code>:redir END</code>	end redirect
<code>:redir >> out.txt</code>	redirect to a file
<code>"*yy</code>	yank to paste
<code>"*p</code>	insert from paste buffer
<code>:'a,'by*</code>	yank range into paste
<code>:%y*</code>	yank whole buffer into paste
<code>:.y*</code>	yank current line to paster
<code>:nmap p :let @* = substitute(@*,'[:print:]','','g')*p</code>	filter non-printable characters

3.4.1 Copy full path name

unix: `nnoremap <F2> :let @*=expand("%:p")`

win32: `nnoremap <F2> :let @*=substitute(expand("%:p"), "/", "\\", "g")`

3.5 Useful tricks

“ayy@a	execute “vim command” in a text file
yy@”	same thing using unnamed register
u@.	execute command JUST typed in
:norm qqy\$jq	paste “normal commands” without entering insert mode

4 Advanced

4.1 Command line tricks

cat xx gvim - -c “v/^d\d\ ^[3-9]/d “	filter a stream
ls gvim -	edit a stream!
gvim ftp://www.somedomain.com/index.html	uses netrw.vim
gvim -h	help
gvim -o file1 file2	open into a split
gvim -c “/main” joe.c	open joe.c & jump to “main”
gvim -c “%s/ABC/DEF/ge update” file1.c	execute multiple command on a single file
gvim -c “argdo %s/ABC/DEF/ge update” *.c	execute multiple command on a group of files
gvim -c “argdo /begin/+1,/end/-1g/^/d update” *.c	remove blocks of text from a series of files
gvim -s “convert.vim” file.c	automate editing of a file (ex commands in convert.vim)
gvim -u NONE -U NONE -N	load vim without .vimrc and plugins (clean vim)
gvim -c ‘normal ggdG”*p’ c:/aaa/xp	access paste buffer contents (put in a script/batch file)
gvim -c ‘s/^/\=@*/ hardcopy! q!’	print paste contents to default printer
gvim -d file1 file2	vimdiff (compare differences)
dp	“put” difference under cursor to other file
do	“get” difference under cursor from other file
:grep somestring *.php	internal grep creates a list of all matching files
:h grep	use :cn(ext) :cp(rev) to navigate list

4.2 External programs

:r!ls.exe	reads in output of ls
!!date	same thing (but replaces/filters current line)
:%!sort -u	sort unique content
:’a,’b!sort -u	as above
!1} sort -u	sorts paragraph (note normal mode!!)
map <F9> :w!c:/php/php.exe %	run file through php
map <F2> :w!perl -c %	run file through perl
:runtime! syntax/2html.vim	convert txt to html

4.3 Recording

qq	record to q
q	end recording
@q	to execute
@@	to repeat
5@@	to repeat 5 times
"qp	display contents of register q (normal mode)
<ctrl-R>q	display contents of register q (insert mode)
"qdd	put changed contacts back into q
@q	execute recording/register q
nnoremap] @l:wbd	combining a recording with a map (to end up in command mode)

4.3.1 Operating a Recording on a Visual BLOCK

1. define recording/register
qq:s/ to/ from/g^Mq
2. Define Visual BLOCK
V}
3. hit : and the following appears
: '<,>
4. Complete as follows
: '<,>norm @q

4.4 Quick jumping between splits

```
map <C-J> <C-W>j<C-W>_
map <C-K> <C-W>k<C-W>_
```

4.5 Visual mode basics

v	enter visual mode
V	visual mode whole line
<C-V>	enter VISUAL BLOCK mode
gv	reselect last visual area (ultra)
o	navigate visual area
"*y	yank visual area into paste buffer
V%	visualise what you match
V}J	join visual block (great)
V}gJ	join visual block w/o adding spaces
0<C-V>10j2ld	delete first 2 characters of 10 successive lines

4.6 vimrc essentials

<pre> set incsearch set wildignore=*.o,*.obj,*.bak,*.exe set shiftwidth=3 set vb t_vb="" set browsedir=buffer </pre>	<pre> jumps to search word as you type tab complete now ignores these for shift/tabbing set silent (no beep!) make 'open directory' use current directory </pre>
--	--

4.6.1 Launching IE

```

nmap ,f :update:silent !start c:\progra~1\intern~1\iexplore.exe file://%:p
nmap ,i :update: !start c:\progra~1\intern~1\iexplore.exe

```

4.6.2 Ftping from vim

```

cmap ,r :Nread ftp://209.51.134.122/public_html/index.html
cmap ,w :Nwrite ftp://209.51.134.122/public_html/index.html

```

4.6.3 Autocmd

<pre> autocmd bufenter *.tex map <F1> :!latex % autocmd bufenter *.tex map <F2> :!xdvi -hush %<.dvi& autocmd BufRead * silent! %s/[\r\t]\+\$// autocmd BufEnter *.php :%s/[\t\r]\+\$//e </pre>	<pre> programming keys depending on file type launch xdvi with current file dvi automatically delete whitespace, trailing dos returns same but only for php files </pre>
---	--

4.7 Conventional shifting and indenting

<pre> :'a,'b>> :vnoremap < <gv :vnoremap > >gv >i{ >a{ >% <% </pre>	<pre> conventional Shifting/Indenting visual shifting (builtin-repeat) visual shifting (builtin-repeat) block shifting (magic) </pre>
---	---

4.8 Pulling objects onto command/search line

<pre> <C-R><C-W> <C-R><C-A> <C-R>- </pre>	<pre> pull word under the cursor into a command line or search pull WORD under the cursor into a command line or search pull small register (also insert mode) </pre>
---	---

4.11 Perform an action on a particular file or file type

4 ADVANCED

<C-R>[0-9a-z]	pull named registers (also insert mode)
<C-R>%	pull file name (also #) (also insert mode)
<C-R>=somevar	pull contents of a variable (eg :let sray="ray[0-9]")

4.9 Capturing output of current script

:new r!perl #	opens new buffer,read other buffer
:new! x.out r!perl #	same with named file
:new+read!ls	
:new +put q %!sort	create a new buffer, paste a register "q" into it, then sort new buffer

4.10 Inserting DOS carriage returns

:%s/\$/\&/g	that's what you type
:%s/\$/\&/g	for Win32
:%s/\$/\^M&/g	what you'll see where ^M is ONE character
:set list	display "invisible characters"

4.11 Perform an action on a particular file or file type

```
autocmd VimEnter c:/intranet/note011.txt normal! ggVGg?
autocmd FileType *.pl exec('set fileformats=unix')
```

“ Retrieving last command line command for copy & pasting into text

i:

“ Retrieving last Search Command for copy & pasting into text

i/

4.12 Inserting line number

```
:g/^/exec "s/^/".strpart(line(".")." ", 0, 4)
:%s/^/\=strpart(line(".")." ", 0, 5)
:%s/^/\=line('.') . ' '
```

4.13 Numbering lines

:set number	show line numbers
:map <F12> :set number!<CR>	map to toggle line numbers
:%s/^/\=strpart(line('.') " ",0,&ts)	

<pre>: 'a, 'b!perl -pne 'BEGIN{\$a=223} substr(\$_,2,0)=\$a++' qqmnYP'n^Aq :.,\$g/^\\d/exe "normal! \" o23qqYpq40@q</pre>	<pre>number lines starting from arbitrary number in recording q repeat with @q increment existing numbers to end of file generate a list of numbers 23-64</pre>
---	---

4.14 Advanced incrementing

```
let g:I=0
function! INC(increment)
  let g:I =g:I + a:increment
  return g:I
end function
```

4.14.1 Create list starting from 223 incrementing by 5 between markers a,b

```
:let I=223
:'a,'bs/^\\=INC(5)/
```

4.14.2 Create a map for INC

```
cab viminc :let I=223 \\| 'a,'bs/$/\\=INC(5)/
```

4.15 Digraphs (non alpha-numerics)

<pre>:digraphs :h dig i<C-K>e' i<C-V>233 i<C-Q>233 ga :::yl/<C-R>"</pre>	<pre>display table help enters enters (Unix) enters (Win32) View hex value of any character Pull a non-ascii character onto search bar</pre>
--	--

4.16 Complex vim

<pre>:%s/\\<\\(on\\ off\\)\\>/\\=strpart("offon", 3 * ("off" == submatch(0)), 3)/g :vnoremap <C-X> <Esc>'. "gvP"P</pre>	<pre>swap two words swap two words</pre>
---	--

4.17 Syntax highlighting

<pre>:set syntax=perl :set syntax off</pre>	<pre>force Syntax coloring for a file that has no extension .pl remove syntax coloring (useful for all sorts of reasons)</pre>
---	--

<code>:colorscheme blue</code>	change coloring scheme (any file in <code>~vim/vim??/colors</code>)
<code>vim:ft=html:</code>	force HTML Syntax highlighting by using a modeline
<code>:syn match DoubleSpace “ “</code>	example of setting your own highlighting
<code>:hi def DoubleSpace guibg=#e0e0e0</code>	sets the editor background

4.18 Preventions and security

<code>:set noma (non modifiable)</code>	prevents modifications
<code>:set ro (Read Only)</code>	protect a file from unintentional writes
<code>:X</code>	encryption (do not forget your key!)

4.19 Taglist

4

<code>:Tlist</code>	display tags (list of functions)
<code><C-]></code>	jump to function under cursor

4.20 Folding

<code>zf}</code>	fold paragraph using motion
<code>v}zf</code>	fold paragraph using visual
<code>zf'a</code>	fold to mark
<code>zo</code>	open fold
<code>zc</code>	re-close fold

4.21 Renaming files

Rename files without leaving vim

```
:r! ls *.c
:%s/\(.*\).c/mv & \1.bla
:w !sh
:q!
```

4.22 Reproducing lines

⁴Script required: taglist.vim (http://www.vim.org/scripts/script.php?script_id=273)

imap] @@@hhkyWjl?@@@P/@@@3s	reproduce previous line word by word
nmap] i@@@hhkyWjl?@@@P/@@@3s	reproduce previous line word by word

4.23 Reading MS-Word documents

```
5 :autocmd BufReadPre *.doc set ro
:autocmd BufReadPre *.doc set hlsearch!
:autocmd BufReadPost *.doc %!antiword "%"
```

4.24 Random functions

4.24.1 Save word under cursor to a file

```
function! SaveWord()
    normal yiw
    exe ':!echo '.@0.' $>$>$ word.txt'
endfunction
```

4.24.2 Delete duplicate lines

```
function! Del()
    if getline(''.') == getline(line(''.') - 1)
        norm dd
    endif
endfunction
```

4.24.3 Columnise a CSV file for display

```
:let width = 20
:let fill=' ' | while strlen(fill) < width | let fill=fill.fill | endwhile
:%s/\([^;]*\);\\=\\/\\=strpart(submatch(1).fill, 0, width)/ge
:%s/\\s\\+$/\\ge
```

```
function! CSVH(x)
    execute 'match Keyword /\\^{'}$\\backslash$([\\^{},]*,$\\backslash$)$\\backslash$\\{' . a:x . '\\}$\\backslash$zs[\\^{},]*/'
    execute 'normal \\{' . a:x . 'f,'
endfunction
```

```
command! -nargs=1 Csv :call CSVH()
:Csv 5 : highlight fifth column
```

4.25 Miscallaenous commands

:scriptnames	list all plugins, _vimrcs loaded (super)
:verbose set history?	reveals value of history and where set
:function	list functions
:func SearchCompl	List particular function

⁵Program required: Antiword (<http://www.winfield.demon.nl/>)

4.26 Vim traps

In regular expressions you must backslash + (match 1 or more)

In regular expressions you must backslash | (or)

In regular expressions you must backslash ((group)

In regular expressions you must backslash { (count)

/fred\+/ /\(fred\)\{2,3}/ /codes\(\n\—\s\)*where /\vcodes(\n s)*where	matches fred/freddy but not free note what you have to break normal regexp very magic
--	--

4.27 Help

:h quickref	vim quick reference sheet (ultra)
:h tips	vim's own tips help
:h visual<C-D><TAB>	obtain list of all visual help topics
:h ctrl<C-D>	list help of all control keys
:helpg uganda	grep help files use :cn, :cp to find next
:h :r	help for :ex command
:h CTRL-R	normal mode
:h /\r	what's \r in a regexp (matches a <CR>)
:h \\zs	double up backslash to find \zs in help
:h i_CTRL-R	help for say <C-R> in insert mode
:h c_CTRL-R	help for say <C-R> in command mode
:h v_CTRL-V	visual mode
:h tutor	vim tutor
<C-[>, <C-T>	Move back & forth in help history
gvim -h	vim command line help
:helptags /vim/vim64/doc	rebuild all *.txt help files in /doc
:help add-local-help	

5 Fun

```
:h 42
```

```
:h holy-grail
```

```
:h!
```

```
vim -c “:‰s‰s*‰Cyrnfr)fcbafe[Oenz(Zbbyranne‰|:‰s)[[( )]-)Ig|norm Vg?”
```

5.1 Contact

5.1.1 Author (David Rayner)

Please email any errors or further tips etc to `david(at)rayninfo.co.uk`

Updated version at <http://www.rayninfo.co.uk/vimtips.html>

5.1.2 Maintainer (Gavin Gilmour)

Please email any comments, suggestions including spelling, grammatical and/or formatting issues with this document to `gavin(at)brokentrain.net`

Updated version available at: <http://gavin.brokentrain.net/projects/vimtips/vimtips.pdf>

5.1.3 Source

The source of this document is available at: <http://github.com/gaving/vimtips/>.

5.2 Links

http://www.vim.org/	Official site
http://chronos.cs.msu.su/vim/newsgroup.html	Newsgroup and Usenet
http://groups.yahoo.com/group/vim	Specific newsgroup
http://u.webring.com/hub?ring=vim	VIM Webring
http://www.truth.sk/vim/vimbook-OPL.pdf	Vim Book
http://vimdoc.sourceforge.net/	Searchable VIM Doc
http://www.faqs.org/faqs/editor-faq/vim/	VIM FAQ