

Vim tips

David Rayner, Gavin Gilmour

August 15, 2010

Contents

1	Text manipulation	5
1.1	Searching	5
1.1.1	Search for declaration of subroutine/function under cursor	6
1.1.2	Search for visually highlighted text	6
1.2	Substitution	6
1.2.1	Filter all form elements into paste register	7
1.2.2	Substitutue within substituion	7
1.2.3	Substituting a visual area	8
1.3	Global command display	8
1.4	Global combined with substitute (power editing)	9
1.5	Changing case	9
1.6	Reformatting text	10
1.7	Deletion without destroying buffer	10
1.8	Essential	10
2	File Manipulation	12
2.1	Exploring	12

<i>CONTENTS</i>	<i>CONTENTS</i>
2.2 Opening files & other tricks	12
2.3 Multiple files management	13
2.4 File-name manipulation	13
2.5 Command over multiple files	14
2.6 Sessions (set of files)	14
2.7 Modelines	14
2.7.1 Creating your own GUI Toolbar entry	15
2.8 Markers & moving about	15
2.9 Editing/moving within insert mode	15
2.10 Abbreviations & maps	16
2.10.1 Display RGB colour under the cursor eg #445588	16
3 Registers	17
3.1 List your registers	17
3.2 Appending to registers	17
3.3 Using a register as a map (preload registers in .vimrc)	17
3.4 Redirection & paste register	17
3.4.1 Copy full path name	18
3.5 Useful tricks	18
4 Advanced	19
4.1 Command line tricks	19
4.2 External programs	19
4.3 Recording	20
4.3.1 Operating a Recording on a Visual BLOCK	20
4.4 Quick jumping between splits	21
4.5 Visual mode basics	21
4.6 vimrc essentials	21

CONTENTS

CONTENTS

4.6.1	Launching IE	21
4.6.2	FTPing from vim	22
4.6.3	Autocmd	22
4.7	Conventional shifting and indenting	22
4.8	Pulling objects onto command/search line	22
4.9	Capturing output of current script	23
4.10	Inserting DOS carriage returns	23
4.11	Perform an action on a particular file or file type	23
4.12	Inserting line number	24
4.13	Numbering lines	24
4.14	Advanced incrementing	24
4.14.1	Create list starting from 223 incrementing by 5 between markers a,b	24
4.14.2	Create a map for INC	25
4.15	Digraphs (non alpha-numeric)	25
4.16	Complex vim	25
4.17	Syntax highlighting	25
4.18	Preventions and security	26
4.19	Taglist	26
4.20	Folding	26
4.21	Renaming files	27
4.22	Reproducing previous line word by word	27
4.23	Reading MS-Word documents	27
4.24	Random functions	27
4.24.1	Save word under cursor to a file	27
4.24.2	Delete duplicate lines	28
4.24.3	Columnise a CSV file for display	28
4.24.4	Highlight a particular csv column	28

<i>CONTENTS</i>	<i>CONTENTS</i>
4.25 Miscallaenous commands	28
4.26 Vim traps	29
4.27 Help	29
5 Fun	31
6 Contact	32
6.1 Author (David Rayner)	32
6.2 Maintainer (Gavin Gilmour)	32
6.3 Source	32
6.4 Links	32

1 Text manipulation

1.1 Searching

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

1.2 Substitution

1.1.1 Search for declaration of subroutine/function under cursor

```
:nmap gx yiw/^\

```

1.1.2 Search for visually highlighted text

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

with spec chars

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

1.2 Substitution

```
:%s/fred/joe/igc
:%s/\r//g
:%s/\r/\r/g
:%s= *\$==
:%s= \+\$==
:%s#\s*\r\?\$##
:%s#\s*\r*\$##
:%s/^\n\{3}//
:%s/^\n\+/\r/
:%s#<[^>]\+>##g
:'a,'bg/fred/s/dick/joe/igc
:%s= [^ ]\+\$=&&=
:%s= \f\+\$=&&=
:%s= \S\+\$=&&
:s/\(.*\):\(.*\)/\2 : \1/
:%s/^\(.*\)\n\1\$/\1/
:%s/^\{-}pdf/new.pdf/
:%s#\<[zy]\?tbl\_ [a-z\_]\+\>#\L&#gc
:%s///
:help /\{-}
:s/fred/a/g
```

general substitute command
delete dos returns ^M
turn dos returns ^M into real returns (fixes joined lines)
delete end of line blanks
same as above
clean both trailing spaces AND dos returns
same thing deleting empty lines
delete blocks of 3 empty lines
compressing empty lines
delete html tags, leave text
VERY USEFUL
duplicate end column
same as above
usually the same
reverse fields separated by :
delete duplicate lines
delete to 1st pdf only
lowercase with optional leading characters
delete possibly multi-line comments
help non-greedy
sub "fred" with contents of register "a"

```

:s/fred/\=@a/g
:%s/\f\+\.gif\>/\r&\r/g | v/\.gif\$/d | %s/gif/jpg/
:%s/a/but/giel:update|:next
:%s/suck\|buck/loopy/gc
:s/\_\_date\_\_/\=strftime("%c")/
:%s:\(\(\w\+\s\+\)\){2}\str1:\1str2:
:%s:\(\w\+\)\(.\*\s\+\)\(\w\+\)\$\:\3\2\1:
:%s/\d\+/\=(submatch(0)-3)/
:g/loc\|function/s/\d/\=submatch(0)+6/
:%s#txtdev\zs\d#\=submatch(0)+1#g
:%s/\(gg\)\@<=\d\+/\=submatch(0)+6/
:let i=10 | 'a,'bg/Abc/s/yy/\=i/ |let i=i+1
:let i=10 | 'a,'bg/Abc/s/xx\zsyy\ze/\=i/ |let i=i+1
:%s/"\([\^.]\\+)\).*\zsxx/\1/
:nmap <leader>z :%s#\<<c-r>=expand("<cword>")<cr>\>#
:vmap <leader>z :<C-U>%s/\<<c-r>*\>/

```

better alternative as register not displayed
 multiple commands on one line
 then use @: to repeat
 ORing (must break pipe)
 insert datestring
 working with columns sub any str1 in col3
 swapping first and last column (4 columns)
 decrement numbers by 3
 increment numbers by 6 on certain lines only
 better version of above
 increment only numbers gg\d\d by 6 (another way)
 convert yy to 10,11,12 etc
 convert xxyy to xx11,xx12,xx13 (more precise)
 find replacement text, use \zs to simplify substitute
 pull word under cursor into LHS of a substitute
 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 @*<Bar>sil exec
  \ 'g@\(\input\<Bar>select\<Bar>textarea\<Bar>/\=form\)\>@p'<Bar>redir END<CR>

```

1.2.2 Substitute within substitution

```

:%s,\(all/.*\)\@<=/_,\_#
:s#all/\zs.*#\=substitute(submatch(0), '/', '_','g')#
:s#all/#&^M#|s#/#_#g|-j!
:%s/.*/\='cp '.submatch(0).' all/' substitute(submatch(0),'/','_','g')/

```

replace all / with _ AFTER "all/"
 same thing
 sub by splitting line, re-joining
 sub inside sub

1.2.3 Substituting a visual area

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

1.3 Global command display

```
:g/gladiolli/#          | display with line numbers (YOU WANT THIS!)
:g/fred.*joe.*dick/ &  | dick
                        | display all lines fred, joe
:g/\<fred\>/          | display all lines fred but not freddy
:g/^s*\$/d            | delete all blank lines
:g!/^dd/d             | delete lines not containing string
:v/^dd/d              | delete lines not containing string
:g/fred/,/joe/d        | not line based (very powerfull)
:g/-----/.-10,.d &   | 10 previous lines
                        | delete string
:g/{/ ,/}- s/\n\+/\r/g | delete empty lines but only between ...
:v/S/d                 | delete empty lines (both types)
:v./././.-j           | compress empty lines
:g/^$/./.-j           | compress empty lines
:g/<input\|<form/p     | ORing
:g/^put_               | double space file (pu = put)
:g/^m0                 | reverse file (m = move)
:'a,'b/^/m'b          | reverse a section a to b
:g/^/t.                | duplicate every line
:g/fred/t\$            | copy lines matching fred to EOF
:g/stage/t'a           | copy lines matching stage to marker a
:g/\(^I[^\I]*\)\{80}/d | delete all lines containing at least 80 tabs
:'a,'bg/somestr/co/otherstr/ | match all lines containing "somestr" between markers a & b
:'a,'bg/str1/s/str1/\\\|mo/str2/ | as above but also do a substitution
:%norm jdd            | delete every other line
:.,\$g/^d/exe "norm! \" | increment numbers
:'a,'bg/d\+/norm! ^A  | increment numbers
:g/fred/y A           | append all lines fred to register a (empty reg a first with qa.)
```

1.5 Changing case

<pre> :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/<pattern>/z#.5 :g/<pattern>/z#.5 echo "======" :g /norm 2f r* :nmap <f3> :redir @a<cr>:g//<cr>:redir END<cr>:new<cr>:put! a<cr><cr> </pre>	<pre> 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)

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

1.5 Changing case

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

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

<code>:nmap ,v :e \\${VIM}/_vimrc</code>		open and edit local vimrc
<code>:vmap sb "zdi<C-R>z<ESC></code>		wrap around VISUALLY selected text
<code>:vmap st "zdi<?= <C-R>z ?><ESC></code>		wrap <?= ?> around VISUALLY selected text

2 File Manipulation

2.1 Exploring

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

2.2 Opening files & other tricks

<code>gf</code>	open file name under cursor (SUPER)
<code>:nnoemap gF :view</code>	open file under cursor, create if necessary
<code>ga</code>	display hex,ascii value of char under cursor
<code>ggVGg?</code>	rot13 whole file
<code>ggg?G</code>	rot13 whole file (quicker for large file)
<code>:8 normal VGg?</code>	rot13 from line 8
<code>:normal 10GVGg?</code>	rot13 from line 8
<code><c-a>,<c-x></code>	increment, decrement number under cursor
<code><c-r>=5*5</code>	insert 25 into text (mini-calculator)
<code>:e main_<tab></code>	tab completes
<code>main_<c-x><c-f></code>	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 <code>ctrl-^</code>)
: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

<code>:!echo %:p</code>		full path & file name
<code>:!echo %:p:h</code>		full path only
<code><C-R>%</code>		insert filename (insert mode)
<code>"%p</code>		insert filename (normal mode)
<code>/<C-R>%</code>		search for file name in text

2.5 Command over multiple files

<code>:argdo %s/foo/bar/e</code>		operate on all files in :args
<code>:bufdo %s/foo/bar/e</code>		operate on all buffers
<code>:windo %s/foo/bar/e</code>		operate on all windows
<code>:argdo exe '%!sort' w!</code>		include an external command

2.6 Sessions (set of files)

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

2.7 Modelines

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

2.7.1 Creating your own GUI Toolbar entry

```
amenu Modeline.Insert\ a\ VIM\ modeline
      \ <esc><esc>gg0vim:ff=unix ts=4 ss=4<CR>vim60:fdm=marker<esc>gg
```

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
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

²(new in vim 6.3)

³(new in vim 6.3)

<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

<code>:reg</code>		display contents of all registers
<code>:reg a</code>		display content of individual registers
<code>"1p...</code>		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

3.2 Appending to registers

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

```
"a5yy
10j
"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

<pre>"*yy "*p :'a,'by* :%y* :.y* :nmap p :let @* = substitute(@*,'[^\[:print:]]','','g')"*p</pre>	<pre>yank to paste insert from paste buffer yank range into paste yank whole buffer into paste yank current line to paster filter non-printable characters</pre>
---	--

3.4.1 Copy full path name

unix:

```
nnoremap <F2> :let @*=expand("%:~")<cr>
```

windows:

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

3.5 Useful tricks

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

4 Advanced

4.1 Command line tricks

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

4.2 External programs

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

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

define recording/register

```
qq:s/ to/ from/g^Mq
```

define Visual BLOCK

V

hit : and the following appears

```
:'<,'>
```

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
O<C-V>10j2ld	delete first 2 characters of 10 successive lines

4.6 vimrc essentials

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

4.6.1 Launching IE

```
:nmap ,f :update<CR>:silent !start c:\progra~1\intern~1\iexplore.exe file:///p<CR>
```

```
:nmap ,i :update<CR>: !start c:\progra~1\intern~1\iexplore.exe <cWORD><CR>
```

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
gvim ftp://www.somedomain.com/index.html # uses netrw.vim
```

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>	<table border="0"> <tr> <td style="border-left: 1px solid black; padding-left: 5px;">programming keys depending on file type</td> </tr> <tr> <td style="border-left: 1px solid black; padding-left: 5px;">launch xdvi with current file dvi</td> </tr> <tr> <td style="border-left: 1px solid black; padding-left: 5px;">automatically delete whitespace, trailing dos returns</td> </tr> <tr> <td style="border-left: 1px solid black; padding-left: 5px;">same but only for php files</td> </tr> </table>	programming keys depending on file type	launch xdvi with current file dvi	automatically delete whitespace, trailing dos returns	same but only for php files
programming keys depending on file type					
launch xdvi with current file dvi					
automatically delete whitespace, trailing dos returns					
same but only for php files					

4.7 Conventional shifting and indenting

<pre>: 'a, 'b>> :vnoremap < <gv :vnoremap > >gv >i{ >a{ >% <%</pre>	<table border="0"> <tr> <td style="border-left: 1px solid black; padding-left: 5px;">conventional Shifting/Indenting</td> </tr> <tr> <td style="border-left: 1px solid black; padding-left: 5px;">visual shifting (builtin-repeat)</td> </tr> <tr> <td style="border-left: 1px solid black; padding-left: 5px;">visual shifting (builtin-repeat)</td> </tr> <tr> <td style="border-left: 1px solid black; padding-left: 5px;">block shifting (magic)</td> </tr> </table>	conventional Shifting/Indenting	visual shifting (builtin-repeat)	visual shifting (builtin-repeat)	block shifting (magic)
conventional Shifting/Indenting					
visual shifting (builtin-repeat)					
visual shifting (builtin-repeat)					
block shifting (magic)					

4.8 Pulling objects onto command/search line

<C-R><C-W> | pull word under the cursor into a command line or search

4.11 Perform an action on a particular file or file type

4 ADVANCED

<code><C-R><C-A></code>		pull WORD under the cursor into a command line or search
<code><C-R>-</code>		pull small register (also insert mode)
<code><C-R>[0-9a-z]</code>		pull named registers (also insert mode)
<code><C-R>%</code>		pull file name (also #) (also insert mode)
<code><C-R>=somevar</code>		pull contents of a variable (eg <code>:let sray="ray[0-9]"</code>)

4.9 Capturing output of current script

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

4.10 Inserting DOS carriage returns

<code>:%s/\$\<C-V><C-M>&/g</code>		that's what you type
<code>:%s/\$\<C-Q><C-M>&/g</code>		for Win32
<code>:%s/\$/\^M&/g</code>		what you'll see where ^M is ONE character
<code>:set list</code>		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<c-r>:
```

Retrieving last Search Command for copy & pasting into text

```
i<c-r>/
```

4.12 Inserting line number

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

4.13 Numbering lines

<pre>:set number :map <F12> :set number!<CR> :%s/^/\=strpart(line('.') . " ",0,&ts) :'a,'b!perl -pne 'BEGIN{\$a=223} substr(\$_,2,0)=\$a++' qqmnp'n^Aq :.,\$g/^/d/exe "normal! \<c-a>"</pre>	<pre>show line numbers map to toggle line numbers number lines starting from arbitrary number in recording q repeat with @q increment existing numbers to end of file</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-numeric)

<pre>:digraphs :h dig i<C-K>e' i<C-V>233 i<C-Q>233 ga :%s/\[<C-V>128-<C-V>255\]//gi :%s/\[-\]//gi :%s/\[<C-V>128-<C-V>255<C-V>01-<C-V>31\]//gi %:exec "norm /\[\\x00-\\x1f\\x80-\\xff\]" yl/<C-R>"</pre>	<pre>display table help enters é enters é (Unix) enters é (Win32) View hex value of any character where you have to type the Control-V Should see a black square & a dotted y All pesky non-asciis same thing Pull a non-ascii character onto search bar</pre>
--	--

4.16 Complex vim

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

4.17 Syntax highlighting

```
:set syntax=perl | force Syntax coloring for a file that has no extension .pl
```

<pre> :set syntax off :colorscheme blue vim:ft=html: :syn match DoubleSpace " " :hi def DoubleSpace guibg=#e0e0e0 </pre>	<pre> remove syntax coloring (useful for all sorts of reasons) change coloring scheme (any file in ~vim/vim?*/colors) force HTML Syntax highlighting by using a modeline example of setting your own highlighting sets the editor background </pre>
--	---

4.18 Preventions and security

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

4.19 Taglist ⁴

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

4.20 Folding

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

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

4.21 Renaming files

Rename files without leaving vim

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

4.22 Reproducing previous line word by word

```
imap ] @@@<esc>hhkyWj1?@@@<cr>P/@@@<cr>3s
nmap ] i@@@<esc>hhkyWj1?@@@<cr>P/@@@<cr>3s
```

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
```

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

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
```

4.24.4 Highlight a particular csv column

```
function! CSVH(x)
  execute 'match Keyword /\^\[^\,]*,\)\{'.a:x.'\}\zs[^\,]*/'
  execute 'normal `'.a:x.'f,'
endfunction
command! -nargs=1 Csv :call CSVH(<args>)
```

Call with `:Csv 5` to highlight fifth column

4.25 Miscallaenous commands

<code>:scriptnames</code>	list all plugins, .vimrcs loaded (super)
<code>:verbose set history?</code>	reveals value of history and where set
<code>:function</code>	list functions
<code>:func SearchCompl</code>	List particular function

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)

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

4.27 Help

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

5 Fun

```
:h 42
:h holy-grail
:h!
vim -c ":%s*s*%Cyrnfr)fcbafbe[Oenz(Zbbyranne%|:%s)[[()])~)Ig|norm Vg?"]}
```

6 Contact

6.1 Author (David Rayner)

Please email any errors or further tips etc to [david\(at\)rayninfo.co.uk](mailto:david(at)rayninfo.co.uk)

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

6.2 Maintainer (Gavin Gilmour)

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

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

6.3 Source

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

6.4 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