Vim tips

August 15, 2010

Contents

1 Text manipulation

1.1 Searching

`/joe/e`	cursor set to end of match
`3/joe/e+1`	find 3rd joe 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/`	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\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 rexexp memory in a search
`/^\([^,]*,\)\{8}`	repeating the regexp (rather than what the regexp finds)
`/<\zs[^>]*\ze>`	search for tag contents, ignoring chevrons (`:h /\zs`)
`/<\@<=[^>]*>\@=`	search for tag contents, ignoring chevrons
`/<\@<=\_[^>]*>\@=`	search for tags across possible multiple lines
`/<!--\_p{-}-->`	search for multiple line comments
`/fred\_s*joe/`	any whitespace including newline (`\_`)
`/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 // y/<C-R>"<CR>
```

with spec chars

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

1.2 Substitution

`:%s/fred/joe/igc`	general substitute command
`:%s/\r//g`	delete dos returns `^M`
`:%s/\r/\r/g`	turn dos returns `^M` into real returns (fixes joined lines)
`:%s= *\$==`	delete end of line blanks
`:%s= \+\$==`	same as above
`:%s#\s*\r\?\$##`	clean both trailing spaces AND dos returns
`:%s#\s*\r*\$##`	same thing deleting empty lines
`:%s/^\n\{3}//`	delete blocks of 3 empty lines
`:%s/^\n\+/\r/`	compressing empty lines
`:%s#<[^>]\+>##g`	delete html tags, leave text
`:'a,'bg/fred/s/dick/joe/igc`	VERY USEFUL
`:%s= [^ ]\+\$=&&=`	duplicate end column
`:%s= \f\+\$=&&=`	same as above
`:%s= \S\+\$=&&`	usually the same
`:s/\(.*\):\(.*\)/\2 : \1/`	reverse fields separated by :
`:%s/^\(.*\)\n\1\$/\1/`	delete duplicate lines
`:%s/^.\{-}pdf/new.pdf/`	delete to 1st pdf only
`:%s#\<[zy]\?tbl\_[a-z\_]\+\>#\L&#gc`	lowercase with optional leading characters
`:%s///`	delete possibly multi-line comments
`:help /\{-}`	help non-greedy
`:s/fred/a/g`	sub "fred" with contents of register "a"

`:s/fred/\=@a/g`	better alternative as register not displayed
`:%s/\f\+\.gif\>/\r&\r/g | v/\.gif\$/d | %s/gif/jpg/`	multiple commands on one line
`:%s/a/but/gie|:update|:next`	then use @: to repeat
`:%s/suck|buck/loopy/gc`	ORing (must break pipe)
`:s/\_\_date\_\_/\=strftime("%c")/`	insert datestring
`:%s:\(\(\w\+\s\+\)\{2}\)str1:\1str2:`	working with columns sub any str1 in col3
`:%s:\(\w\+\)\(.*\s\+\)\(\w\+\)\$:\3\2\1:`	swapping first and last column (4 columns)
`:%s/\d\+/\=(submatch(0)-3)/`	decrement numbers by 3
`:g/loc|function/s/\d/\=submatch(0)+6/`	increment numbers by 6 on certain lines only
`:%s#txtdev\zs\d#\=submatch(0)+1#g`	better version of above
`:%s/\(gg\)\@<=\d\+/\=submatch(0)+6/`	increment only numbers `gg\d\d` by 6 (another way)
`:let i=10 | 'a,'bg/Abc/s/yy/\=i/ |let i=i+1`	convert yy to 10,11,12 etc
`:let i=10 | 'a,'bg/Abc/s/xx\zsyy\ze/\=i/ |let i=i+1`	convert xxyy to xx11,xx12,xx13 (more presise)
`:%s/"\([^.]\+\).*\zsxx/\1/`	find replacement text, use `\zs` to simplify substitute
`:nmap <leader>z :%s#\<<c-r>=expand("<cword>")<cr>\>#`	pull word under cursor into LHS of a substitute
`:vmap <leader>z :<C-U>%s/\<<c-r>*\>/`	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 Substitue within substituion

`:%s,\(all/.*\)\@<=/,_,g`	replace all / with _ AFTER "all/"
`:s#all/\zs.*#\=substitute(submatch(0), '/', '_', 'g')#`	same thing
`:s#all/#&^M#|s#/#_#g|-j!`	sub by splitting line, re-joining
`:%s/.*/\='cp '.submatch(0).' all/'.substitute(submatch(0),'/','_','g')/`	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/ & display all lines fred,joe	dick
: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 & delete string	10 previous lines
: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 qaq.)

`:g/fred/y A | :let @*=@a`	put into paste buffer
`:let @a="|g/Barratt/y A |:let @*=@a`	
`:'a,'b g/^Error/ . w >> errors.txt`	write out to errors.txt
`:g/./yank|put|-1s/'/"/g|s/.*/Print '&'/`	duplicate every line in a file wrap a print " around each duplicate
`:g/^MARK\$/r tmp.ex | -d`	replace string with contents of a file, -d deletes the "mark"
`:g/<pattern>/z#.5`	display with context
`:g/<pattern>/z#.5|echo "=========="`	display beautifully
`:g/|/norm 2f|r*`	replace 2nd — with a star
`:nmap <f3>  :redir @a<cr>:g//<cr>:redir END<cr>:new<cr>:put! a<cr><cr>`	send output of previous global command to a new window

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$\sim$$\sim$`	flip case line
`vEU`	upper case word
`vE$\sim$`	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<b><C-R>z</b><ESC>`	wrap `<b></b>` around VISUALLY selected text
`:vmap st "zdi<?= <C-R>z ?><ESC>`	wrap `<?= ?>` around VISUALLY selected text

2 File Manipulation

2.1 Exploring

`:Exp(lore)`	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]
`\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
`<c-a>,<c-x>`	increment, decrement number under cursor
`<c-r>=5*5`	insert 25 into text (mini-calculator)
`:e main_<tab>`	tab completes
`main_<c-x><c-f>`	include NAME of file in text (insert mode)

[1] Script 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 `ctrl-^`)
`: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 extention 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`	operate on all buffers
`:windo %s/foo/bar/e`	operate on all windows
`: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\ 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

`:reg`	display contents of all registers
`:reg a`	display content of individual registers
`"1p....`	retrieve numeric registers one by one
`:let @y='yy@"'`	pre-loading registers (put in .vimrc)
`qqq`	empty register "q"
`:let @a=@_`	clear register a
`:let @a=""`	clear register a
`:let @*=@a`	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

`:redir @*`	redirect commands to paste buffer
`:redir END`	end redirect
`:redir >> out.txt`	redirect to a file

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

3.4.1 Copy full path name

unix:

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

windows:

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

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

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
`0<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

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

4.7 Conventional shifting and indenting

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

4.8 Pulling objects onto command/search line

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

<C-R><C-A>	pull WORD under the cursor into a command line or search
<C-R>-	pull small register (also insert mode)
<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/$\<C-V><C-M>&/g	that's what you type
:%s/$\<C-Q><C-M>&/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<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

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

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)

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

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

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

4.18 Preventions and security

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

4.19 Taglist [4]

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

4.20 Folding

`zf}`	fold paragraph using motion
`v}zf`	fold paragraph using visual
`zf'a`	fold to mark
`zo`	open fold
`zc`	re-close fold

[4] Script 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>hhkyWjl?@@@<cr>P/@@@<cr>3s
nmap ] i@@@<esc>hhkyWjl?@@@<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
```

[5] Program 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

`: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

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

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

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