If you are transitioning from windows, then in order of preference you will probably want to use, **evim**, **gvim**, **vim**.
evim makes gvim behave like windows editors (note Ctrl+L gets you to vim Normal mode). If one just wants the windows key mappings from evim mode in gvim for e.g. you can use the following command: **source $VIMRUNTIME/mswin.vim**. Other alternatives if you're transitioning from windows are **nedit** or **geany**.

If you're stuck with text mode access, then vim is probably the best option, so you need this info (all of which also applies to gvim and evim). Another reason that it is good to know this is that many programs use vim key bindings. For e.g. the readline library can be configured to use vi bindings and hence this info will be useful to you in bash, ftp,gnuplot,python,bc,... Also less (used to read man pages on Linux) defaults to vi key bindings.

Note most commands take an optional number (of times to run) on front
(e.g. 3**.** repeats the last action 3 times, or 3w moves forward 3 words etc.).

Here are my vim settings and my gvim settings.

Command	**Action**	**Notes**
vim file +54	open file and go to line 54	any : command can be run using + on command line
vim -O file1 file2	open file1 and file2 side by side	
Insert	enter insert mode	so you can start typing. Alternatively one can use **i** or **a**.
Esc	leave insert mode	so you can issue commands. Note in VIM the cursor keys & {Home, End, Page{up,down}} and Delete and Backspace work as expected in any mode, so you don't need to go back to command mode nearly as much as the origonal vi. Note even Ctrl+{left,right} jumps words like most other editors. Note also Ctrl+[and **Ctrl+c** are equivalent to Esc and may be easier to type. Also Ctrl+o in insert mode will switch to normal mode for one command only and automatically switch back.
:command	runs named command	
:help word	shows help on word	Typing Ctrl+d after word shows all entries containing word
:echo &word	shows value of word	
windows		
:e	set buffer for current window	you can optionally specify a new file or existing buffer number (#3 for e.g.). Note if you specify a directory a file browser is started. E.g. :e **.** will start the browser in the current directory (which can be changed with the :cd command).
:sp	new window above	ditto
:vs	new window to left	ditto
:q	close current window	
:qa	close all windows	add trailing ! to force

Ctrl+w {left,right,up,down}	move to window	
Ctrl+w Ctrl+w	toggle window focus	
Ctrl+w =	autosize windows	to new terminal size for e.g.
:ba	new window for all buffers	":vert ba" tiles windows vertically
buffers		
:ls	list buffers	
gf	open file under cursor	
:bd	delete buffer	and any associated windows
:w	save file	Note :up[date] only writes file if changes made, but it's more awkward to type
:sav filename	save file as filename	Note :w filename doesn't switch to new file. Subsequent edits/saves happen to existing file
undo/redo		
u	undo	
Ctrl+r	redo	
.	repeat	
navigation		
gg	Goto start of file	
G	Goto end of file	
:54	Goto line 54	
80|	Goto column 80	
Ctrl+g	Show file info	including your position in the file
ga	Show character info	g8 shows UTF8 encoding
Ctrl+e	scroll up	Ctrl+x needed first for insert mode
Ctrl+y	scroll down	Ctrl+x needed first for insert mode
zt	scroll current line to top of window	
w	Goto next word	Note Ctrl+{right} in newer vims (which work also in insert mode)
b	Goto previous word	Note Ctrl+{left} in newer vims
[{	Goto previous { of current scope	
%	Goto matching #if #else,{},(),[],/* */	must be one on line
zi	toggle folds on/off	
bookmarks		
m {a-z}	mark position as {a-z}	E.g. m a
' {a-z}	move to position {a-z}	E.g. ' a
''	move to previous position	
'0	open previous file	handy after starting vim
selection/whitespace		
v	select visually	use cursor keys, home, end etc.
Shift+v	line select	CTRL+v = column select
Delete	cut selection	
"_x	delete selection	without updating the clipboard or yank buffer. I remap x to this in my .vimrc
y	copy selection	
p	paste (after cursor)	P is paste before cursor
"Ay	append selected lines to register a	use lowercase a to initialise register
"ap	paste contents of a	
gq	reformat selection	justifies text and is useful with :set textwidth=70 (80 is default)

=	reindent selection	very useful to fix indentation for c code
>	indent section	useful with Shift+v%
<	unindent section	remember . to repeat and u to undo
:set list!	toggle visible whitespace	See also listchars in my .vimrc
clipboard shortcuts		
dd	cut current line	
yy	copy current line	
D	cut to end of line	
y$	copy to end of line	
search/replace		
/regexp	searches forwards for regexp	? reverses direction
n	repeat previous search	N reverses direction
*	searches forward for word under cursor	# reverses direction
:%s/1/2/gc	search for regexp 1 and replace with 2 in file	c = confirm change
:s/1/2/g	search for regexp 1 and replace with 2 in (visual) selection	
programming		
K	lookup word under cursor in man pages	2K means lookup in section 2
:make	run make in current directory	
Ctrl+]	jump to tag	Ctrl+t to jump back levels. I map these to Alt+⇦⇨ in my .vimrc
vim -t name	Start editing where name is defined	
Ctrl+{n,p}	scroll forward,back through autocompletions for word before cursor	uses words in current file (and included files) by default. You can change to a dictionary for e.g: set complete=k/usr/share/dicts/words Note only works in insert mode
Ctrl+x Ctrl+o	scroll through language specific completions for text before cursor	"Intellisense" for vim (7 & later). :help compl-omni for more info. Useful for python, css, javascript, ctags, ... Note only works in insert mode
external filters		
:%!filter	put whole file through filter	
:!filter	put (visual) selection through filter	
:,!command	replace current line with command output	
map <f9> :w<CR>:!python %<CR>	run current file with external program	