UM-St. Louis Instructional Computing Labs

VI EDITOR

Starting vi

`vi` *file*	Edits *file*
`vi +`*n file*	Edits *file* at line *n*
`vi` *file1 file2*	Edits multiple files in order listed

Saving, quitting, and editing another file

`ZZ`	Exits and saves
`:wq`	Exits and saves
`:x`	Exits and saves
`:q!`	Exits and discards changes
`:q`	Quits if no changes were made
`:w`	Saves file
`:e` *filename*	Edits *filename*
`:n`	Edits next file in document list
`:r` *file*	Inserts *file* after cursor line

Moving Around

`<Ctrl B>`	Page up
`<Ctrl F>`	Page down
`<Ctrl G>`	Show file status and line number
`<Ctrl D>`	Scroll forward half a screen
`<Ctrl U>`	Scroll backward half a screen
`<Ctrl E>`	Scroll forward one line
`<Ctrl Y>`	Scroll backward one line
`G`	Move to last line in document
n`G`	Move to line *n* (also :*n)*
`H`	Move to first line on screen
`M`	Move to middle line on screen
`L`	Move to last line on screen
n`H`	Move *n* lines from top
n`L`	Move *n* lines before bottom
`0`	Move to beginning of line
`$`	Move to end of line
n`|`	Move to column #*n* on line
`^`	Move to first non-blank char.
`e`	Move to end of next word
`w`	Move to beginning of next word
`b`	Move to backward one word
`%`	Move to matching parenthesis
`)`	Next sentence
`(`	Beginning of sentence
`}`	Next paragraph
`{`	Beginning of current paragraph
`]]`	Next section
`[[`	Beginning of current section
`+`	Move to first char. of next line
`-`	Move to first char of previous line

Searching

/*word*	Search forward for *word*
?*word*	Search backward for *word*
n	Repeat previous search
N	Repeat in opposite direction
f*x*	Line Search. Move to next *x*
F*x*	Move to previous *x* on line

Text Manipulation

`i`	Insert text before cursor
`I`	Insert text before first non-blank
`a`	Insert text after cursor
`A`	Insert text after last non-blank
`o`	Open a line below the current
`O`	Open a line above the current
`r`*n*	Change single character to *n*
`cw`	Change word
`cc`	Change current line
`C`	Change to end of line
`r` *x*	Replace single character *x*
`R`	Type over characters
`s`	Substitute character
`S`	Substitute line
`x`	Delete character
`X`	Delete character before cursor
`d0`	Delete everything on line before cursor
`dw`	Delete word
`dd`	Delete current line
`D`	Delete to end of line
`:.,$d`	Delete from current line to end of file
`p`	Place deleted/yanked text after cursor
`P`	Place deleted/yanked text before cursor
`yw`	Yank (copy) word
`yy`	Yank current line
`u`	Undo last command
`U`	Restore current line
`J`	Join two lines
`~`	Change case at cursor
`:%s/`***old***`/`***new***`/g`	Globally replaces ***old*** with ***new***
`:%s/`***old***`/`***new***`/gc`	Same, but asks for a confirmation

Command Line

`:sh`	Invoke sub shell
`<Ctrl D>`	Return to vi from shell
`:!` ***command***	Invoke UNIX ***command***
`:`*n*`,`*m*`!` ***command***	Filter lines *n* to *m* through UNIX ***command***
`:r !` ***command***	Place output of ***command*** into file
`:set` ***option***	Activate *option* (see back for options)
`:set` *no***option**	Deactivate *option*
`:set`	List options currently set
`:map` *x command*	Sets keystroke *x* to execute *command*. Macro
`:unmap` *x*	Disables mapped keystroke *x*
`:ab` *abbr text*	When *abbr* is typed it is replaced with *text.*
`%`	Current filename, in command line mode.

Miscellaneous

`<Ctrl L>`	Redraw current screen
`m`*x*	Mark current position with *x*
`'`*x*	Move to first char. of line marked w/ *x*
`` `x ``	Move cursor to char. marked by *x*
`` `` ``	Return to previous mark
`''`	Return to beginning of line of prev. mark

Advanced Commands

Command Combinations:

Change	Delete	Copy	from Cursor to…
cH	dH	yH	top of screen
cL	dL	yL	bottom of screen
c+	d+	y+	next line
c*x*|	d*x*|	y*x*|	column *x* of current line
*x*c)	*x*d)	*xy*)	*x*th sentence following
c(	d(	y(	previous paragraph
c /*pattern*	d /*pattern*	y /*pattern*	*pattern*
cn	dn	yn	next *pattern*
cG	dG	yG	end of file
c*x*G	d*x*G	y*x*G	line number *x*

Buffers:

Buffers will allow you to move text around in current documents, or move text from one document to another. To do the later you can yank any text into a buffer and then switch to another file by using **:e *file*** and then past from that buffer there. To switch back to the previous file use **:e #** or **<Ctrl ^>**

Named Buffers	
a-z	There are buffers with names from a to z. They are there for you to use as needed (just remember which is which). Also, uppercase letters append to the corresponding buffer.
"***a***yy	Yank current line into buffer ***a***
"***a x***yy	Yank next ***x*** lines into buffer ***a***
"***a***P	Put the contents of buffer ***a*** before cursor
"***a***p	Put the contents of buffer ***a*** after cursor
"***a x***dd	Delete ***x*** lines into buffer ***a***
"***a***d)	Delete from cursor to end of current sentence and save in buffer ***a***
"***A***y)	Add next sentence to buffer ***a***

Recovering Deletions	
"*x*p	With vi you can recover any of your past *nine* deletions. The " identifies the buffer *x* (where *x* is from 1 – 9), p places text after the cursor.
"1p u.	This is a neat trick. If you can't remember which text you need you can search for it by using the a combination of the *u* (undo) command, and the **.** (dot – redo) command. So if it is not there after **"1p** then type **u.** until it appears.

Vi Options:

These options can be set manually while in vi by using the :set command (see command line section) or like the :map and :ab commands can be placed in a file called ***.exrc*** and will then be loaded every time that you begin a vi session. Here is a sample of useful options, for more refer to *UNIX in Nutshell* (sect. ***8-11***)

Options	
ai	Auto-indent. Indents line to the same level as the line above or below.
eb	Error Bell. Sounds bell if error occurs. Set by ***default***.
mesg	Allows System messages on screen. Set by ***default***.
nu	Displays line numbers at left of lines.
sm	Show match. Moves cursor briefly to matching (or { when) or } is typed.
showmode	When in insert mode, a message is displayed in the right bottom corner of screen.