Chapter 3 Review: Mastering Editors

bit – a binary digit, representing on or off
machine language – the exclusive use of 0s and 1s as a way to communicate with a computer
byte – 8 bits
text files – computer files composed of nothing but printable characters
binary files – files that contain non-printable characters such as machine instructions
ASCII – American Standard Code for Information Interchange, a character set standard for mapping bytes to particular text characters
Unicode – a new character set standard for mapping multiple international character set standards to a single format, several encodings are used to represent subsets of Unicode
bitmap – a binary image file format
png – a binary image file format
pdf – a file format for encoding documents, can be binary or text depending on the format
sh - a text file for storing executable instructions for an interpreter such as bash or other parser
log – a text file containing transactional information output by a computer program
csv – a text file containing data stored as text and delimited by commas or tabs
html – a text file type whose standard formatting can be read by a web browser
compiling – a process of translating a program file into machine-readable language
executable program – a file that can be compiled or interpreted to issue commands
editor – a program for creating and modifying files
text editor – a simplified word processor for editing text files
screen editors – text editors that let you edit a document by seeing its contents laid out on a screen and manipulating portions of it with a cursor e.g. Vi, Emacs, and Nano
line editor – a programs that edits a document one line at a time
modal editor – an editor by which you switch modes to perform different functions

Vi Review

Modes

Vi has three modes insertion mode, command mode, and extended mode.

Insertion Mode is entered by pressing the i key on the keyboard. It lets you edit text
Command Mode is entered by pressing the ESC key. It lets you issue commands
Extended Mode is entered by typing a : while in command mode it is used to issue advanced file commands in the fashion of a line editor

Quitting

:x	Exit, saving changes
:q	Exit as long as there have been no changes
ZZ	Exit and save changes if any have been made
:q!	Exit and ignore any changes

Inserting Text

i	Insert before cursor
I	Insert before line
a	Append after cursor
A	Append after line
o	Open a new line after current line
O	Open a new line before current line
r	Replace one character
R	Replace many characters

Motion

h	Move left
j	Move down
k	Move up
l	Move right
w	Move to next word
W	Move to next blank delimited word
b	Move to the beginning of the word
B	Move to the beginning of blank delimted word
e	Move to the end of the word
E	Move to the end of Blank delimited word
(	Move a sentence back
)	Move a sentence forward
{	Move a paragraph back
}	Move a paragraph forward
0	Move to the begining of the line
$	Move to the end of the line
1G	Move to the first line of the file
G	Move to the last line of the file
nG	Move to nth line of the file
:n	Move to nth line of the file
fc	Move forward to c
Fc	Move back to c
H	Move to top of screen
M	Move to middle of screen
L	Move to botton of screen
%	Move to associated ( ), { }, [ ]

Deleting Text

Almost all deletion commands are performed by typing d followed by a motion. For example, dw deletes a word. A few other deletes are:

x	Delete character to the right of cursor
X	Delete character to the left of cursor
D	Delete to the end of the line
dd	Delete current line
:d	Delete current line

Yanking Text

Like deletion, almost all yank commands are performed by typing y followed by a motion. For example, y$ yanks to the end of the line. Two other yank commands are:

yy	Yank the current line
:y	Yank the current line

Changing text

The change command is a deletion command that leaves the editor in insert mode. It is performed by typing c followed by a motion. For wxample cw changes a word. A few other change commands are:

C	Change to the end of the line
cc	Change the whole line

Putting text

p	Put after the position or after the line
P	Put before the poition or before the line

Buffers

Named buffers may be specified before any deletion, change, yank or put command. The general prefix has the form "c where c is any lowercase character. for example, "adw deletes a word into buffer a. It may thereafter be put back into text with an appropriate "ap.

Markers

Named markers may be set on any line in a file. Any lower case letter may be a marker name. Markers may also be used as limits for ranges.

mc	Set marker c on this line
`c	Go to beginning of marker c line.
'c	Go to first non-blank character of marker c line.

Search for strings

/*string*	Search forward for *string*
?*string*	Search back for *string*
n	Search for next instance of *string*
N	Search for previous instance of *string*

Replace

The search and replace function is accomplished with the :s command. It is commonly used in combination with ranges or the :g command (below).

:s/*pattern*/*string*/*flags*	Replace *pattern* with *string* according to *flags*.
g	Flag - Replace all occurences of pattern
c	Flag - Confirm replaces.
&	Repeat last :s command

Regular Expressions

. (dot)	Any single character except newline
*	zero or more occurances of any character
[...]	Any single character specified in the set
[^...]	Any single character not specified in the set
^	Anchor - beginning of the line
$	Anchor - end of line
\<	Anchor - begining of word
\>	Anchor - end of word
\(...\)	Grouping - usually used to group conditions
\n	Contents of nth grouping

[...] - Set Examples

[A-Z]	The SET from Capital A to Capital Z
[a-z]	The SET from lowercase a to lowercase z
[0-9]	The SET from 0 to 9 (All numerals)
[./=+]	The SET containing . (dot), / (slash), =, and +
[-A-F]	The SET from Capital A to Capital F and the dash (dashes must be specified first)
[0-9 A-Z]	The SET containing all capital letters and digits and a space
[A-Z][a-zA-Z]	In the first position, the SET from Capital A to Capital Z In the second character position, the SET containing all letters

Regular Expression Examples

/Hello/	Matches if the line contains the value Hello
/^TEST$/	Matches if the line contains TEST by itself
/^[a-zA-Z]/	Matches if the line starts with any letter
/^[a-z].*/	Matches if the first character of the line is a-z and there is at least one more of any character following it
/2134$/	Matches if line ends with 2134
/\(21|35\)/	Matches is the line contains 21 or 35 Note the use of ( ) with the pipe symbol to specify the 'or' condition
/[0-9]*/	Matches if there are zero or more numbers in the line
/^[^#]/	Matches if the first character is not a # in the line
Notes: 1. Regular expressions are case sensitive 2. Regular expressions are to be used where *pattern* is specified	

Counts

Nearly every command may be preceded by a number that specifies how many times it is to be performed. For example, 5dw will delete 5 words and 3fe will move the cursor forward to the 3rd occurence of the letter e. Even insertions may be repeated conveniently with thismethod, say to insert the same line 100 times.

Ranges

Ranges may precede most "colon" commands and cause them to be executed on a line or lines. For example :3,7d would delete lines 3-7. Ranges are commonly combined with the :s command to perform a replacement on several lines, as with :.,$s/pattern/string/g to make a replacement from the current line to the end of the file.

:n,m	Range - Lines n-m
:.	Range - Current line
:$	Range - Last line
:'c	Range - Marker c
:%	Range - All lines in file
:g/*pattern*/	Range - All lines that contain *pattern*

Files

:w *file*	Write to *file*
:r *file*	Read *file* in after line
:n	Go to next file
:p	Go to previos file
:e *file*	Edit *file*
!!*program*	Replace line with output from *program*

Other

~	Toggle upp and lower case
J	Join lines
.	Repeat last text-changing command
u	Undo last change
U	Undo all changes to line

Emacs Review

Basics (mandatory)

You simply can't get by without having these at your fingertips.

- **C-x C-c** - quit
- **C-x C-s** - save buffer
- **C-x C-f** - open file
- **C-x b** buffer name - switch to open buffer
- **C-g** - cancel
- **C-x k** - close current buffer
- **C-h a** command name - look up docs for command
- **M-x** command name - execute command
- **C-x u** - undo
- **C-/** - undo

Getting around (really useful, worth learning)

Navigation

- **C-a** - beginning of line
- **C-e** - end of line
- **C-f** - forward character
- **C-b** - backward character
- **C-p** - down a line
- **C-n** - up a line
- **M-f** - forward word
- **M-b** - backward word
- **S-**any of the above - navigate and select
- **C-space** - start selection

Cutting and pasting

- **C-w** - cut
- **C-y** - paste

Search and replace

- **C-s** - search
- **M-%** - search and replace

More advanced editing

Deletion and cutting

- **C-d** - delete character ahead
- **M-d** - delete word ahead
- **backspace** - delete character behind
- **M-backspace** - delete word behind
- **C-k** - cut from cursor to end of line

Insertion

- **C-o** - insert newline after cursor

Frames

- **C-x 2** - split window horizontally
- **C-x 3** - split window vertically
- **C-x 1** - unsplit window
- **C-x o** - switch to other pane in split window

Tricks

- **M-q** - auto-hard-wrap current paragraph
- **C-t** - swap the two characters at the cursor
- **M-u** - uppercase the word at the cursor
- **M-l** - lowercase the word at the cursor
- **C-u** n char - insert n copies of char

Power tools

- **M-x** replace-regexp - search and replace by regexp (the quoting/escaping is so weird that this always takes me several tries)
- **C-r t** - "string-rectangle" (this one's really weird but super useful sometimes; look up the docs)