Advanced Features of Vim

Topics

- In-Editor help
- Search and Replacement
- Customize your vim environment
- Advanced editing

In-Editor Help

- While in command mode, enter:
  - `:help`
  - `:help [topic]` or `:h [topic]`
- Jump to a subject:
  - Position the cursor on a tag (e.g., `|bars|` and hit `CTRL-]`)
  - Enable mouse `:set mouse=a`
- Jump back:
  - `CTRL-T` or `CTRL-O`

Vim modes

- Command mode `<ESC>`
- Insert mode `i`
- Visual mode `v, V, Ctrl-V`
- Command line mode : `<-> <Esc>`

Cursor Motions

- Cursor motion commands move the cursor position
  - Left-right motion: `h,l,^,$`
  - Up-down motion: : `k, j, gk, gj, +, -`
  - Word motions: `w, W, e, E, b, B`
  - f{char}/F{char}: forward/backward to the next/previous occurrence of {char}
  - Others: `%, [(, [{, …`
- See `:h motion.txt` for detail

The Visual Mode

- Three visual modes:
  - Visual: `v`
  - Blockwise visual mode : `Ctrl-V`
  - Linewise visual mode: `V`
- Stop visual mode: `v-<Esc>, v-Ctrl-V`

Text Objects

- Text objects define regions of text by structure
- Text objects consist of 2 elements
  - Exclusiveness: i for inside, a for all
  - Delimiter: {}, “” , etc.
- See :h text-objects

Text Objects Used in Visual Mode

Object	Selection
aw	A word with white space
iw	Inner word
aW	A WORD with white space
iW	Inner WORD
as	A sentence
is	Inner sentence
ap	A paragraph with white space
ip	Inner paragraph
ab	A () block with parenthesis
ib	Inner () block
aB	A {} block with braces
iB	Inner block

Changing The Visual Area

- o : go to the other end of highlighted text
  - Visual: `v`
  - Blockwise visual mode : `Ctrl-V`
  - Linewise visual mode: `V`
- Stop visual mode: `v-<Esc>, v-Ctrl-V`

Visual Mode Operators

Operator	Meaning
c	Change
d	Delete
y	Yank into register
~	Swap case (only if ‘tilderop’ is set
g~	Swap case
gu	Make lower case
gU	Make upper case
>	Shift right
<	Shift left
=	Autoindent

Text Object Selection Motions

TO Motions	Meaning
(	Sentence backward
)	Sentence forward
{	Paragraph backward
)	Paragraph forward
]]	Section forward
[[	Section backward

Operator + Motion = Action

- An action is composed from an operator followed by a motion.
  - `dap deletes the paragraph (command mode)`
  - `v+i(gU (visual mode)`
- When an operator command is invoked in duplicate, it acts upon the current line.
  - `dd deletes the current line`
  - `gUgU or gUU turns the current line uppercase`

Accessing the Shell

- Vim lets you interact with the shell without leaving the editor
  - Enter the shell temporarily `:sh`
  - Enter the shell without leaving vim: `:!cmd`
  - Insert text from file to the current buffer `:r filename`
  - Exit the shell: `exit`

Searching

- /text searches forward for text
- ?text searches backward for text
- n repeats previous search
- N repeats previous search in opposite direction

Search with Regex

- Search string is interpreted as regex
- Vim regex works differently than the one we are accustomed to using.

Empty Regular Expressions

- In some utilities such as vim and less (but not grep), an empty regex represents the last pattern

```
:s/mike/robert
:s//robert/   (repeat the above command)
```

or

```
/mike/
:s//robert/
```

Vim Regex

- In some utilities such as vim and less (but not grep), an empty regex represents the last pattern

```
:s/mike/robert
:s//robert/   (repeat the above command)
```

or

```
/mike/
:s//robert/
```

Lookahead & Lookbehind

- Lookahead
  - Positive lookahead: \@=
  - Negative lookahead: \@!
- Lookbehind
  - Postive lookbehind:\@<=
  - Negative lookbehind:\@<!

Search and Substitute

- `:s/old/new/`

  replaces first old with new on current lin
- `:s/old/new/g`

  replaces every old with new on current line
- `:%s/old/new`

  replaces first old with new on **every** line
- `:g%s/old/new/g`

  replaces **every** old with new on **every** line

Substitute Syntax

:[g][address]s/search-string/replacement[/option]

g – global

Address – range

search-string – A regular expression

replacement – A replacement string. Can be a back reference.

Options:

c	confirm each substitution
g	replace all occurrences (w/o g only first)
i	ignore case for the pattern
I	Don’ t ignore case for the pattern

Range, Line Addressing & Marks

Specifier	Description
number	an absolute line number
.	the current line
$	the last line in the file
%	the whole file. The same as **1,$**
'*t*	position of mark "t"
/*pattern*[/]	the next line where text *"pattern*" matches.
?*pattern*[?]	the previous line where text "*pattern*" matches
\/	the next line where the previously used search pattern matches
\?	the previous line where the previously used search pattern matches
\&	the next line where the previously used substitute pattern matches

- Range limits the command execution in a particular part of the text.
- Line range consists of one or more line addresses, separated by a comma or semicolon
- If no line range is specified, the current line will be operated on.

Special Characters for Replacement

Symbol	Represents
`\r`	Insert a carriage return
`\t`	Insert a tab character
`\\`	Insert a single backslash
`\1`	Insert the first submatch
`\2`	Insert the second submatch (and so on, up to \9)
`\0`	Insert the entire matched pattern
`&`	Insert the entire matched pattern
`~`	Use {string} from the previous invocation of :substitute

Substitute Examples

Command	Result
`:%s/.*/(&)/`	Reproduce the entire line, but add parentheses
`:%s/(\(.*\))/\1/`	Remove the parentheses in the above lines
`:s/.*/mv & &.old/`	Change a wordlist (one word per line) into mv commands
`:g/^$/d`	Delete blank line
`:g/^[ tab]*$/d`	Delete blank line plus lines containing spaces or tabs
`:%s/  */ /g`	Turn one or more spaces into one space
`:s/^[0-9]/Item &:/g`	Turn line

Empty Regular Expressions

- In some utilities such as vim and less (but not grep), an empty regex represents the last pattern

```
:s/mike/robert
:s//robert/   (repeat the above command)
```

or

```
/mike/
:s//robert/
```

Search & Replace in Visual Mode

- When text is visually selected, press : to a command
- The command will automatically enter the range: `'<,'>`
- You can then enter a command, like this:

```
:'<,'>s/old/new/
```

Lookahead & Lookbehind

- Lookahead
  - Positive lookahead: \@=
  - Negative lookahead: \@!
- Lookbehind
  - Postive lookbehind:\@<=
  - Negative lookbehind:\@<!

Advanced Editing Features

The Dot (.) Command

- The dot command is a macro
- Macro allows to capture an action and play back later.

Keystrokes	Buffer Contents
{start}	var foo = 1 var bar = 'a' var foobar = foo + bar
A;<Esc>	var foo = 1; var bar = 'a' var foobar = foo + bar
j	var foo = 1 var bar = 'a'; var foobar = foo + bar
.	var foo = 1; var bar = 'a'; var foobar = foo + bar

j.	var foo = 1;
	var bar = 'a';
	var foobar = foo + bar;

Advanced Editing

- `>G` increases the indentation from the current line until the end of the file
- The dot command (`.`) repeats the last change (see `:h .`)

Keystrokes	Buffer Contents

`{start}` With a couple of keystrokes, we can use these to select a chunk of text	`Line one`
	`Line two`
	`Line three`
	`Line four`

`>G`	`Line one`
	`  Line two`
	`  Line three`
	`  Line four`

`j`	`Line one`
	`  Line two`
	`  Line three`
	`  Line four`

`.`	`Line one`
	`  Line two`
	`    Line three`
	`    Line four`

`j.`	`Line one`
	`  Line two`
	`    Line three`
	`      Line four`

Indenting, Auto-indent, Word Wrap

- >> indents current line
- << outdents current line

Filtering Through Shell Commands

`!!` filters current line through shell commands

`n!!` filters **n** lines

`!%` filters to matching parenthesis brace or bracket

`!}` filters next paragraph

`!{` filters previous paragraph

- Useful commands include `fmt`, `tr`, `grep`, and `awk`

Customize Vim to Suit Your Preferences

- Change settings on the fly
  - `:set ignorecase #turn if on`
  - `:set noignorecase #turn it off`
  - `:set ignorecase! #toggle the setting`
  - `:set ignorecase? #what is the setting`
- See `:h option-list` for a quick list and `:h options` for details
- Save your configuration in `~/.vmrc`