Session 6: Vim and Regular Expressions

P. S. Langeslag

22 November 2018

Selective Vim Settings

`:set ic`	Ignore case (I have set this as default)
`:set noic`	Heed case
`:set wrap`	Use line wrapping (default)
`:set nowrap`	Disable line wrapping
`:syntax on`	Enable syntax highlighting (default)
`:syntax off`	Disable syntax highlighting

- See `:help options`
- For help on individual options, use single quotes: `:help 'syntax'`
- Set persisent options in `~/.vimrc`

Interacting With Files and Programs

`:r[ead] file`	Read contents of `file` into current file
`:so[urce] file`	Interpret `file` as a sequence of Vim commands
`:shell`	Open a shell; return with `exit`
`:!cmd`	Run `cmd` (substitite program name); e.g. `:!ls`

- See
  - `:help :r`
  - `:help :so`
  - `:help :shell`
  - `:help :!cmd`

Windows (Panes)

`:new` / `CTRL+W n`	Start a new file in a new window (horizontal split)
`:vne`	Start a new file in a new window (vertical split)
`CTRL+W j` (or `<DOWN>`)	Make the lower window active
`CTRL+W k` (or `<UP>`)	Make the upper window active
`:q`	Close the active window
`:on[ly]`	Close all windows except the active one

- `:shell` and `:!cmd` will always take the whole Vim viewport

File Encoding and Character Encoding

These commands may help convert non-unicode files (e.g. Windows-1252):

Command	Action
`:set encoding`	Prints current display encoding
`:set encoding=utf-8`	Sets display encoding to `utf-8`
`:set fileencoding`	Prints encoding for current file
`:set fileencoding=utf-8`	Sets file encoding to `utf-8`

- See:
  - `:help encoding`
  - `:help fileencoding`

`vi` Mode on the Command Line

For Vim navigation on the command line, add to `~/.bashrc`:

```
set -o vi
```

(Run `source ~/.bashrc` to reload the file, or log in anew.)

Remember sed?

Operation	Effect
`sed s/He/She/ file`	Replace all instances of `He` in `file` with `She` (`-s` for "substitute"); print result to `stdout`, leaving `file` untouched
`sed -n s/He/She/p file`	Idem, but print affected lines only (`-n` for "no output"; `p` for "print")
`sed -n s/He/She/pI file`	Idem, but case insensitive (`I` for "insensitive"); note unwanted effects
`sed -n s/He/She/gpI file`	Idem, replace beyond the first match in the string (sentence) (`g` for "global")
`sed -i.bak s/He/She/ file`	Idem, but save results to original file and copy original file to backup file `file.bak` (`-i` for "insert"); no output
`sed -i s/He/She/ file`	Idem, but without backup file! Risky!
`sed -i s/He/She/ *`	Idem, but for every file in working directory. Highly risky!

Search in Vim

Command	Action
`/pattern`	Search for `pattern`
`n`	Repeat search (next hit)
`N`	Repeat search (previous hit)

Pattern Anchors

Syntax	Represents
`^`	Start of line
`$`	End of line
`\<`	Word boundary, initial
`\>`	Word boundary, terminal

```
/\<he\>
```

Logical Operator

Form	Represents
|	“or”

```
/p\|t\|k
/they\|their\|them
```

Quantifiers

Expression	Matches
*	0 or more of the preceding atom, greedy
\+	1 or more of the preceding atom, greedy
\=	0 or 1 of the preceding atom, greedy
\{n}	Exactly n of the preceding atom
\{n,m}	n to m inclusively of the preceding atom, greedy
\{-n,m}	n to m inclusively of the preceding atom, non-greedy
\{,m}	No more than m inclusively of the preceding atom, greedy
\{n,}	At least n of the preceding atom, greedy
\{-n,}	At least n of the preceding atom, non-greedy
\{-,m}	No more than m inclusively of the preceding atom, non-greedy
\{-}	0 or more of the preceding atom, non-greedy

- Greedy: Return the largest match
- Non-greedy: Return the smallest match

```
/t\{2,3}
```

Collections (Character Classes)

Syntax	Matches
`[aeiouy]`	“a,” “e,” “i,” “o,” “u,” or “y”
`[!@#$%^]`	“!,” “@,” “#,” “$,” “%,” or “^”
`[^aeiouy]`	Any character other than “a,” “e,” “i,” “o,” “u,” or “y”
`[0-9]`	Any digit
`[^0-9]`	Any character other than a digit
`[a-zA-Z]`	Any alphabetical character (strictly ASCII/Latin)

- To look for `^`, don’t put it first in the class; or escape it (`\^`)
- To look for `-`, put it first or last; or escape it (`\-`)
- To look for `\`, escape it (`\\`)
- See `:help /[]`

```
/[0-9]\{4}
```

Metacharacters and Predefined Character Classes

Syntax	Matches
`.`	Any character except newline
`\s`	Whitespace character (`<SPACE>` or `<TAB>`)
`\S`	Non-whitespace character
`\t`	Tab
`\n`	Newline ("line feed")
`\r`	Carriage return (there's a difference! Cf. here)
`\d`	Digit, i.e. `[0-9]`
`\D`	Non-digit, i.e. `[^0-9]`
`\a`	Alphabetical character (strictly ASCII), i.e. `[a-zA-z]`
`\A`	Non-alphabetical character (strictly ASCII), i.e. `[^a-zA-z]`
`\w`	Word character (strictly ASCII), i.e. `[0-9a-zA-Z_]`
`\W`	Non-word character (strictly ASCII), i.e. `[^0-9a-zA-Z_]`
`\X`	Unicode grapheme
`\p`	Printable character

A workaround for unicode alphabeticals is `[[:lower:][:upper:]]`.

Substitution in Vim

```
:range s[ubstitute]/pattern/string/options
```

Substitution in Vim

```
:range s[ubstitute]/pattern/string/options
```

Options

`c`	confirm each substitution
`g`	apply globally within the line, not just to the first hit
`i`	ignore case
`I`	heed case

- Without `i` or `I`, Vim applies the global `ic`/`noic` setting.

Substitution in Vim

```
:range s[ubstitute]/pattern/string/options
```

Range

`.`	current line (default)
`15`	line 15
`1,10`	lines 1 up to and including 10
`$`	last line
`%` or `1,$`	all lines
`/pattern/`	next line matching `pattern`

- When using `/pattern/`, `string` will be matched against `pattern`, not the containing line.
- See `:help range`

Escape Characters

To Match This	Type This
`.`	`\.`
`/`	`\/`
`\`	`\\`
(space)	`\` (backslash + space)
`{`	`\{`

When matching forward slashes, try using a different delimiter, e.g. `+`:

```
:%s+http://+https://+g
```

To avoid having to type:

```
:%s/http:\/\//https:\/\//g
```

Grouping and Backreferences

`\(. . .\)` in pattern	treats sub-expression as an atom
`\0` in replacement string	reproduces full match
`\1` in replacement string	reproduces first sub-expression
`\l` in replacement string	reproduces following character in lowercase
`\u` in replacement string	reproduces following character in uppercase
`\L` in replacement string	reproduces following sequence in lowercase
`\U` in replacement string	reproduces following sequence in uppercase
`\E` in replacement string	ends `\L`/`\U`-sequence
`\r` in replacement string	insert linebreak

```
:%s/\(two\)\ \(three\)\ \(one\)/\3 \1 \2
```

Lookaround

Function	Syntax
Positive lookahead	`\@=`
Negative lookahead	`\@!`
Positive lookback	`\@<=`
Negative lookback	`\@<!`

Syntax	Matches
`/child\(hood\)\@=`	“child” where followed by “hood”
`/child\(hood\)\@!`	“child” where not followed by “hood”
`/\(child\)\@<=hood`	“hood” where preceded by “child”
`/\(child\)\@<!hood`	“hood” where not preceded by “child”

- See `:help /\@=` ff.

Zoom Anchors

A simpler solution to matching part of a pattern.

Syntax	Function
`\ze`	End the match here
`\zs`	Start the match here

Syntax	Action
`:%s/child\zehood/mother/g`	Replace “child” with “mother” where followed by “hood”
`:%s/child\zshood/lessness/g`	Replace “hood” with “lessness” where preceded by “child”

Vim's `global` Command

```
:range g[lobal]/pattern/command
```

The default range is all lines (`%`).

Command Options

- `s[ubstitute]`
- `co[py]`
- `d[elete]`
- etc.

Examples

```
:g/friend/s//foe/g      # same as ':%s/friend/foe/g'
:g/^$/d                 # deletes all empty lines
```

Recommended Reading

Moolenaar, Bram. "Vim Online," n.d. http://www.vim.org.

Raisky, Oleg. "Vim Regular Expressions 101," n.d. http://vimregex.com.

Robbins, Arnold. *vi and Vim Editors: Pocket Reference*. 2nd ed. Sebastopol, CA: O'Reilly, 2011.

Robbins, Arnold, Elbert Hannah, and Linda Lamb. *Learning the vi and Vim Editors: Pocket Reference*. 7th ed. Sebastopol, CA: O'Reilly, 2008.