RegEx
(or Regular Expressions)

Kirk Carlson
Retired telecommunications guy

A little history…

- Started out as a computer science theory in the 1951.
- Studied as a finite automata
- Popularized by Unix in the 1960’s
- Use in that other OS by the Weedin Brothers, MKS Toolkit, Cygwin and others
- Picked up by the GNU project for GNU/Linux
- Now in many ..ix commands (sed, grep, vi) and most languages (Perl, PHP, Python, Java, JavaScript, C, C++, C#, Ruby, Delphi, R, Tcl)

What is a regular expression?

- A mini-language within a language
- A way to find strings of a pattern in any text
- A way to check if a string fits a pattern
- A way to confuse the heck out of beginners
- A write-only language, basically unreadable
- Something to contrast irregular expressions
- A way to make ...ix users totally love ..ix

Simplest Regular Expressions

- Just literal text
- What you type is what you search for
- ‘dog’ looks for the letters ‘dog’
- Literal characters are case sensitive
- Works for digits and letters and some symbols
  - These are the literal characters
  - Some symbols have special meaning or ‘**magic**’ which we will get into

Add a wild card match

- A period '**.**' will match any character
  - 't.p' will match tap, tip, top, tup, tcp, t3p, t.p, etc.

- NOTE: that this is similar to the shell's use of the question mark '**?**'

Add an any number modifier

- Characters in a regex can be modified by a count modifier
- The asterisk *, means any number (zero or more) of the preceding character).
  - ‘the*‘ will match ‘th‘, ‘the‘, ‘thee‘, ‘theee‘, etc.
  - ‘the.* ‘ will match ‘the ‘, ‘there ‘, ‘then ‘, ‘theory ‘, etc.
- NOTE: a RegEx ‘.*’ is similar to a shell ‘*’

Other character count modifiers

- * (asterisk) means any number (0 or more)
- ? (question mark) is for an optional character (0 or 1)
- + (plus sign) is for a required repeatable character (1 or more)
- {n} exactly n occurrences
- {,n} up to n occurrences
- {n,} n or more occurrences
- {n,m} between n and m occurrences

Sets

- A set can match any character that is specified by that set.
- A set is defined by square brackets [and]
  - [abc] is the set of characters a, b, and c
  - [abc] matches a, b, or c
- A set may have any number of characters
- A set may include ranges
  - [a-z] is the set of lower case letters
  - [0-9a-fA-F] is the set of characters used for hexidecimal

Why Not

- Sets can be negated by starting them with a caret **^**.
- ‘**[^**0**-**9**]**’ is the set of non-digit characters
- ‘<**[^**<>**]+**>’ will match any HTML tag by finding the tag delimiters < and > surrounding one or more characters that are not < or >.

Removing Magic

- How to include a '**[**', '**]**', '**-**', or other magic character in a set?
  - Just precede it with a **** backslash character
    - **[\[\]\-]** is a set for the square brackets and the hypen
    - Within square brackets, a hypen as the first or last character is treated also as a literal hypen, no backslash is needed so **[\[\]-]** or **[-\[\]]** works \as well
- Backslash makes any **magic** character literal
- Backslash makes some literal characters **magic**

NOTE: luckily this follows into many ..ix commands

Boundary Characters

- **^** caret means beginning of line
- **$** dollar sign means end of line
  - ‘**^**The’ finds a ‘The’ at the beggining of a line
  - ‘Amen**$**’ finds an ‘Amen’ at the end of a line
- **\<** matches the beginning of a word
- **\>** matches the end of a word
  - ‘**\<[**tT**]**he**\>**’ finds the word ‘the’ or ‘The’

Control characters

- Control characters just add magic to regular characters
  - **\t** is a tab
  - **\r** is a carriage return
  - **\n** is a new line
  - **\f** is a form feed or new page
  - **\v** is a vertical tab (should you ever need it)
- Very useful to make the invisible visible

Shortcuts

- **\d** is any digit, equivalent to **[**0**-**9**]**
- **\w** is any word character equivalent to **[**-_a**-**zA**-**z**]**
- **\s** is a white space character **[**t****r****n****f**]**
- **\x** is a hexidecimal digit equivalent to **[**0**-**9a**-**fA**-**F**]**
- **\O** is and octal digit equivalent to **[**0**-**7**]**

NOTE: may not work inside a set

Groups

- Used to match one of several possibilities
  - Group defined by open and close parentheses **()**
  - Solidus or pipe **|** may separate sequences within a group as a logical OR.
  - **(**cat**|**dog**|**lion**)** matches cat or dog or lion
  - **(**info**|**debug**)** matches info or debug

Text Captures

- Like a group, but what gets found is important and is captured as a group of characters
- '**([**-_a**-**zA**-**Z0**-**9**\.]+)**@**(([**-_a**-**zA**-**Z**]+**)**\.([**a**-**zA**-**Z**]+))**' to match and capture the user name, the full domain name, the domain name, and the domain of an e-mail address
- Use with replace commands
- Groups are numbered left to right and may be nested
- **\1** calls out the first group or the user name
- **\4** calls out the fourth group or domain

Greediness

- Tough concept in regular expressions
- Regular expressions are greedy, they take the longest string that matches the regex
  - For the string ‘<red>Word</red>’
  - ’<**.***>’ matches the entire string
  - To make it so called lazy or needy instead of greedy, add a **?** after the count magic character (***** here)
  - ‘<**.*?**>’ matches only ‘<red>’

How to play with RegEx

- Work on one component at a time
- Start with the command mode of vi
  - /something to search for something
  - :g/something/p to find all lines with something
- Then move to sed -Ee ‘g/something/s//somethingElse/g’
- Add and delete magic as necessary
- Check your expression or (someone else’s) at www.gskinner.com/RegExr/.

Caveats

- Implementations vary
  - Some magic characters are literal and vice versa
  - Some short cuts are not implemented or don’t work inside of sets
- Some commands require loading of a special interpreted with hyphen flags (sed -E)
- Just drop back to the more basic (and wordy) syntax

Personal use cases

- Fixing file names. Checks from a bank are named by the bank for the bank. I reorder the fields for me.
- Fixing command output. OpenWRT/LEDE has a list of DHCP assignments. I reorder the columns and sort by IP address.
- Pre-processing source files. In the mid ‘80’s used RegEx and sed to pre-process state tables for a key telephone system. Lots of features involving lots of states and lots of transitions yielding a high confusion potential. Included converting nroff commands into assembly language directives. (36-hour run time on a PDP 11/23 soured the deal and forced rewrite into c.)
- It was a dark period when PCs and Macs did not have RegEx and I was working on a Tandem box.