Contents

REGEX

TODO: - sed, when to use? - emphasize goal: use grep -P, regex to understand how REGEX works. Tired of every 6 months learning all over again. - greedy/not greedy and backtrack . Think like a regex engine! - see Blue pg 29. - How to aerate regex ! - regex can be used to: - find - validate - replace/insert - split - ... - But GOAL know is simply HOW it works. (grep -P) - 2nd Goal is do in R, but suspect much, much easier. - When whiz, can do summersaults with CLI, zsh tools (sed, grep , cut ...) and regex. Not NOW.

- Separate learning REGEX (grep -P, regex) and using REGEX in R, which I think is a tad easier.

DEFINTIIONS - as always, crucial

```
-   regex is a string;  do not forget this.
-   META CHARACTERS - ascii (?) characters which by-default have non-literal
    meaning to engine that digests them.  **Engine** specific. Must ESCAPE these characters to use as litera
    contexts, such as unix shell, have similar idea:  `<`, `>' for example,
    refer to **redirect** .    In C, sprintf, `%` indicates formatting and
    literal use.
-   **To Escape** indicate to underlying engine that this meta character
should be handled as though literal.

-   POSIX:
    -   backslash \
    -   [ ]
    -   { }
    -   ( )
    -   caret ^
    -   $
    -   dot .
    -   pipe |
    -   ?
    -   asterisk *
    -   `+ -`
    -   ``+ - ''
    -   \verbX + - X
    -   \begin{verbatim}
        + -
        \end{verbatim}
```

Render REGEX Verbatim - 4 ways

+ -

“+ - ’’

+ –

+ –

+ –

- **Character Class** Things like [0-9].
  Rmk: [0-9]+ means repeat one or more of the prior **Character class** So both 321 and 333 match this regex.

Specific to vim/neovim

- magic = ňo need to escape (wait till know what doing first)
- magic = \v no need to escape (wait till know what doing first)

Specific to R

- Before regex library (engine) sees code, the **compiler** (byte code?) gets it first. Must use double backslash for just one backslash to be seen by regex engine. Shell interpreters have no such compiler and single backslash suffices.

Reading (return to tech_reading)
Focus: **grep -P, regex useage**:

- https://linuxize.com/post/regular-expressions-in-grep/#grep-regular-expression (overview, not bad place to start)
- another overview: https://bsd.org/regexintro.html
- !wikipidia - several excellent articles and background.
- GNU grep documentation: https://www.gnu.org/savannah-checkouts/gnu/grep/manual/grep.html#Top

Read the wikipedia articles!

Because touch upon many issues: quoting, expansions, quasi-quotation, recursion, definitions which I have stumbled accross but never really understood at appropriate abstraction. Now it may clarify why do what we do and why the nomenclature is the way it is.

specific question (else can get lost in all the permutations.) - !so regex FAQ: https://stackoverflow.com/tags/regex/info

Too comprehensive? (docs that cover flavors, usage in languages are too confusing to me)

- https://www.regular-expressions.info/tutorial.html

Finite Automata?

- https://sodocumentation.net/regex
- https://swtch.com/~rsc/regexp/regexp1.html