regex & grep

Pattern matching

POSIX standard

IEEE 1003.1-2001 IEEE Standard for IEEE Information Technology – Portable Operating System Interface (POSIX(TM))

Ref: `https://standards.ieee.org/standard/1003_1-2001.html`

Regex

- regex is a pattern template to filter text
- BRE: POSIX Basic Regular Expression engine
- ERE: POSIX Extended Regular Expression engine

Why learn regex?

- Languages: Java, Perl, Python, Ruby, ...
- Tools: grep, sed, awk, ...
- Applications: MySQL, PostgreSQL, ...

Usage

- `grep ‘pattern’ filename`
- `command | grep ‘pattern’`
- Default engine: BRE
- Switch to use ERE :
  ```
  egrep ‘pattern’ filename
  grep -E ‘pattern’ filename
  ```

Special characters (BRE & ERE)

.	Any single character except null or newline
*	Zero or more of the preceding character / expression
[ ]	Any of the enclosed characters; hyphen (-) indicates character range
^	Anchor for beginning of line or negation of enclosed characters
$	Anchor for end of line
\	Escape special characters

Special characters (BRE)

\{n,m\}	Range of occurances of preceding pattern at least n and utmost m times
\(\)	Grouping of regular expressions

Special characters (ERE)

{n,m}	Range of occurances of preceding pattern at least n and utmost m times
( )	Grouping of regular expressions
+	One or more of preceding character / expression
?	Zero or one of preceding character / expression
|	Logical OR over the patterns

Character classes

`[[:print:]]`	Printable	`[[:blank:]]`	Space / Tab
`[[:alnum:]]`	Alphanumeric	`[[:space:]]`	Whitespace
`[[:alpha:]]`	Alphabetic	`[[:punct:]]`	Punctuation
`[[:lower:]]`	Lower case	`[[:xdigit:]]`	Hexadecimal
`[[:upper:]]`	Upper case	`[[:graph:]]`	Non-space
`[[:digit:]]`	Decimal digits	`[[:cntrl:]]`	Control characters

Backreferences

- `\1` through `\9`
- `\n` matches whatever was matched by nth earlier paranthesized subexpression
- A line with two occurances of hello will be matched using:
  `\(hello\).*\1`

BRE operator precedence

highest

- `[..] [==] [::]` char collation
- `\metachar`
- `[ ]` Bracket expansion
- `\( \) \n` subexpresions and backreferences
- `* \{ \}` Repetition of preceding single char regex
- Concatenation
- `^ $` anchors

lowest

ERE operator precedence

highest

- `[..]` `[==]` `[::]` char collation
- `\metachar`
- `[ ]` Bracket expansion
- `( )` grouping
- `* + ? { }` Repetition of preceding regex
- Concatenation
- `^ $` anchors
- `|` alternation

lowest