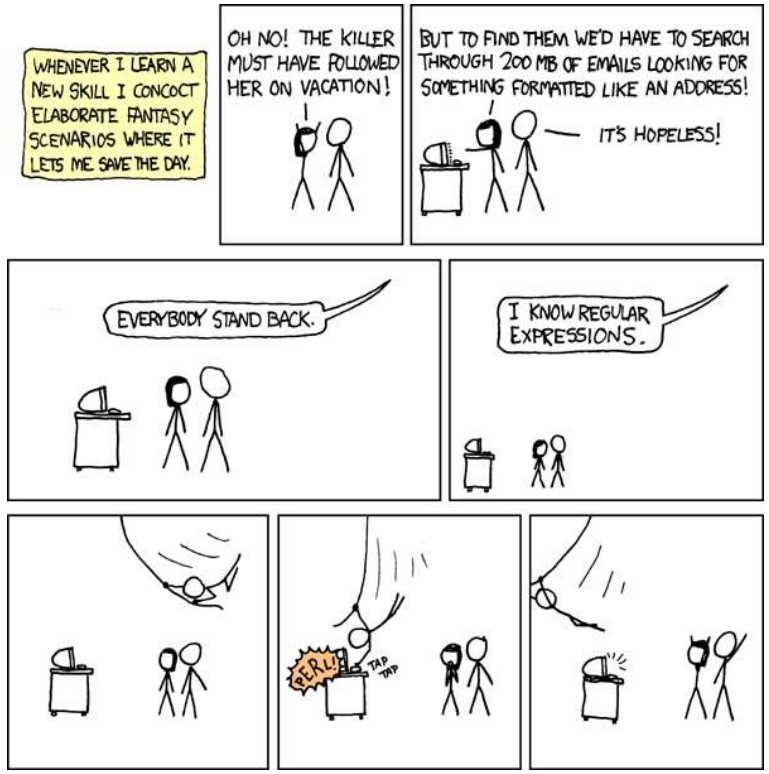

1878
Western
UNIVERSITY · CANADA

Regular Expressions

Winter 2022



Definition

- A regular expression is a special text string that represents a search pattern
- Used by
  - Some Unix utilities (e.g. grep)
  - Editors (e.g. vi, emacs, most IDEs)
  - Most programming languages (e.g. awk, sed, Perl, Python using re, Java using java.util.regex, C using regex.h)

Definition

- There are three regular expression syntaxes
  - “basic” (BRE) (POSIX standard)
  - “extended” (ERE) (POSIX standard)
  - “perl” (PCRE). (Not POSIX standard)
    - More powerful and flexible
    - Used by most programming languages

Definition

- grep on Gaul can understand all three
  - grep -F to disable regular expression matching (formerly: fgrep)
  - grep -G for basic (The -G is optional. This is the default)
  - grep -E for extended (formerly: egrep)
  - grep -P for Perl (Do not confuse this with pgrep!)

Regular expressions and filename expansion

- Regular expressions are *similar* to but not the *same* as filename expansion (globbing)
- Filename expansion is interpreted by the shell **first**
- Then regular expressions are interpreted by the utility (e.g. grep)

Regular expressions and filename expansion

- For this reason, it is not required but it is safest to wrap your regular expression in single quotes
- E.g.
  - grep '[uU]nix' file.txt matches unix or Unix in file.txt as intended

Regular expressions and filename expansion

- E.g. (continued)
  - grep [uU]nix file.txt attempts to expand to grep unix Unix file.txt which is probably not intended

BRE – Start and end

- ^ - Matches the start of the line
- $ - Matches the end of the line

BRE – Start and end

- E.g.
  - ^START – Match only lines that begin with START
  - END$ - Match only lines that end with END
  - ^WHOLE$ - Match only lines that contain WHOLE and nothing else
  - ^$ - Match empty lines

BRE – Bracket expressions

- [ ] – Match any characters in the brackets
- [^] – Match any characters NOT in the brackets

BRE– Bracket expressions

- E.g.
  - [abc] – Matches a or b or c
  - [^abc] – Matches any character except a or b or c
  - [a-z] – Matches a, b, c, … y, z
  - [^a-z] – Matches any character except for a, b, c, … y, z

BRE - Repetition

- . – Matches any single character
- * or \? - Matches 0 or more characters
- \+ - Matches 1 or more characters
- \{m,n\} – Matches the preceding character at least m times but no more than n times.
  - \{m} – A shorthand for \{m,m\}
  - \{m,} or \{,n} – A shorthand for \{m,∞\} and \{0,n\}

BRE - Repetition

- E.g.
  - a.b – Matchces axb, a$b, abb, a.b
    - Does not match ab, axxb, a$bc
  - a*b – Matches b, ab, aab, aaab, …
    - Does not match axb
  - .* - Matches anything including nothing

BRE - Repetition

- E.g.
  - a\{3,5\} – Matches aaa, aaaa, aaaaa ONLY
  - [a-z]\{2\} – Matches all two character lower-case strings (e.g. as, we, to, aa, etc.)

BRE - Sets

- REGEX1\|REGEX2 – Matches either REGEX1 or REGEX2
  - E.g. cat\|dog – Matches cat or dog

BRE - Backreferences

- \(REGEX\) – "Save" anything between the parentheses matched by REGEX
- \1, \2, …, \9 – "Recall" the first, second, …, ninth REGEX match
- This is known as recall or backreference

BRE - Backreferences

- E.g.
  - \([a-z]\)e\1 – Matches aea, beb, cec, etc.

BRE – Special characters

- To match special characters, add or remove the \ character if necessary
- E.g.
  - \\ - Match the \ character
  - * - Match the * character
  - ? – Match the ? character (not \?)
  - …

ERE

- The same as BRE with some modifications
  - Removes backreferences
  - Most special characters no longer require backslashes

```
[wbeldman@compute ~]$ grep 'd{2}' readme.txt
[wbeldman@compute ~]$ grep 'd\{2\}' readme.txt
Add some more text forgot to add something here!
[wbeldman@compute ~]$ grep -E 'd{2}' readme.txt
Add some more text forgot to add something here!
[wbeldman@compute ~]$ grep -E 'd\{2\}' readme.txt
```

ERE

- Since backreferences are not available, the () can be used a little differently:
  - (ab)+ - Matches ab, abab, ababab, etc.

Character classes

- Supported by BRE and ERE
- Shorthand for multiple character types
  - [:alpha:] – Alphabetic characters
  - [:digit:] – Digits
  - [:space:] – Whitespace characters
  - A number of others

Character classes

- E.g.
  - [:upper:]et – Matches Bet, Get, Jet, etc.
  - [:xdigit:] – Matches 0-9, A-F, and a-f
  - [:punct:] – Matches punctuation characters

Regular Expressions - Continued

- The grep manpage has more information
- https://en.wikipedia.org/wiki/Regular_expression

grep

- Finally, some common grep options you should be aware of:
  - -c – "Count" the number of matches
  - -i – Case insensitive matching
  - -v – "invert" match. (Find the lines that do NOT match the expression)

```
[wbeldman@compute ~]$ grep -c '^un.*g$' /usr/share/dict/words
1866
```

grep

- grep with some additional context (not all versions of grep have this)
  - -A # - Print # lines after the match
  - -B # - Print # lines before the match
  - -C # - Print # lines before and after the match

grep

```
[wbeldman@compute ~]$ grep -A 2 3 test.txt
3
4
5
[wbeldman@compute ~]$ grep -B 2 3 test.txt
1
2
3
[wbeldman@compute ~]$ grep -C 2 3 test.txt
1
2
3
4
5
```


1878
Western
UNIVERSITY · CANADA