

Regular Expressions

1

YUAN LONG
CSC 3320 SYSTEM LEVEL PROGRAMMING
FALL 2016

Updated based on original notes from Raj Sunderraman and Michael Weeks

Regular Expression

2

- A pattern describing a certain amount of text.
- A “match” is the piece of text that pattern was found to correspond to by the *regex* processing software.
- For example,

Pattern	Will Match
<code>/\d{1,2}\V\d{1,2}\V\d{4}/</code>	Date (e.g. 9/5/2016)
<code>/\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3} /</code>	IP Address (e.g. 255.255.0.0)

Different Regular Expression Engines

3

- A regular expression "engine" is a piece of software that can process regular expressions, trying to match the pattern to the given string.
- Different regular expression engines are not fully compatible with each other.
 - E.g. Perl, PHP, .NET, Java, JavaScript, Python, R, POSIX
- In this class, we learn **BREs**(Basic Regular Expression), and **EREs**(Extended Regular Expression), used by UNIX utilities, such as ***grep***, ***egrep***, ***awk***, ***sed***, and ***vi***.

How to write a regular expression?

4

- A regular expression is composed of
 - Characters
 - Delimiters
 - Simple strings
 - Special characters
 - Other metacharacters
 - Character classes

Characters

5

- Any character on the keyboard (except newline character ‘\n’)
- **Literals**
 - The characters represent themselves within a regex
- **Special characters**
 - The characters not represent themselves within a regex
 - E.g. \$ matches the end of a line only.
- In a regex, the literal characters can be treated as *words*, while the special characters can be treated as *grammar*.
- The most basic regex consists of a single literal character, e.g.: **a**

Delimiters

6

- Special characters used to mark the beginning and end of a regex.
- Any characters can be used as a delimiter.
- But most often, people use forward slash '/'
 - E.g. `/\d{1,2}\/\d{1,2}\/\d{4}/`
- Delimiters are not used with **grep** family of utilities

Simple Strings

7

- The most basic regular expression
- Matches only itself

Regex	Matches	Examples
/the/	the	So turn off the light, Say all your prayers and then , Beautiful mermaids will swim through the sea, And you will be swimming there too

Special Characters

8

- The metacharacters with special meaning are

Extended
Basic

- The dash (-) can only be considered as a metacharacter within the square brackets to indicate a range.
 - E.g. **[0-9]** matches the digits through 0 to 9
- The brace **{ }** can only be considered as a metacharacter when it is part of repetition operator
 - E.g. **(cat){1,3}** matches the strings containing one to three “cat”

Special Characters

9

- To match any single character, use `.`
- To match any of the single characters in a range, use `[]`
 - E.g. `[a-z]`, `[2-5]`, `[abc]`
 - Other metacharacters lose their special meanings inside square bracket
- To convert a metacharacter into a literal, use backslash `\`
 - E.g. `\[`, `\/`

Special Characters

10

Regex	Matches	Examples
<code>/.nd/</code>	All strings with any character preceding nd	Say all your prayers and then, Oh you sleepy young heads dream of wonderful things, And you will be swimming there too.
<code>[A-D]</code>	Member of the character class A through D	B eautiful mermaids will swim through the sea, A nd you will be swimming there too.
<code>\.</code>	A period	And you will be swimming there too .
<code>a.</code>	A string with prefix a	S ay a ll your prayers and then, Oh you sleepy young head s dream of wonderful things, Be a utiful mermaid s will swim through the sea ,

Special Characters

11

- **\$** matches the end of a line only
 - E.g. >\$
- **^** matches the beginning of a line only
 - E.g. ^ID
 - Note: **[^abc]** means any characters excluding a, b and c

Regex	Matches	Examples
/a.\$/	A string with prefix a ends a line	Beautiful mermaids will swim through the sea a ,
/^.*nd/	All strings with any character preceding nd and at beginning of a line	And you will be swimming there too.

Special Characters

12

- **?** matches **zero or one** occurrence of the single preceding character.
 - E.g. a?, abc?
- ***** matches **zero or more** occurrences of the character that precedes it.
 - E.g. a*, abc*
- **+** matches **one or more** occurrences of the single preceding character.
 - E.g. a+, abc+

Special Characters

13

Regex	Matches	Examples
/sw.*ng/	sw followed by zero or more other characters followed by ng	And you will be swimming there too.
/s.*w/	s followed by zero or more other characters followed by w	Oh you sleepy young heads dream of wonderful things, Beautiful mermaids will swim through the sea, And you will be swimming there too.
/s.+w/	s followed by more other characters followed by w	Oh you sleepy young heads dream of wonderful things, Beautiful mermaids will swim through the sea,
/s.?w/	s followed by zero or one other character followed by w	And you will be swimming there too.

Special Characters

14

- **()** group the matched string by regex inside it
 - E.g. `(abc)*` v.s. `abc*`
- **|** matches either expression, acts like an “or” operator
 - E.g. `(a|b)`, `(a|bc)`, `a|bc*`

Regex	Matches	Examples
<code>/off will/</code>	off or will may appear	And you will be swimming there too.
<code>(ab){1,2}c</code>	ababc abc	ababc Aababd This is abc

Character Classes

15

Character class	Matches
<code>\s</code>	White space
<code>\S</code>	Not white space
<code>\d</code>	Digit
<code>\D</code>	Not digit
<code>\w</code>	Word
<code>\W</code>	Not word
<code>\x</code>	Hexadecimal digit
<code>\O</code>	Octal digit

Rules

16

- Longest match possible
 - A regex always matches the longest possible string, starting as far towards the beginning of the line as possible
- Empty regular expressions
 - An empty regular expression always represents the last regular expression used
 - E.g. in vi editor
 - ✦ Try **:s/john/kate/**
 - ✦ If you want to make the same substitution again, use
 - ✦ **:s//kate**

Practices

17

- Write regular expressions for following patterns.
 - Phone number with the format ###-###-####
 - Date with the format MM/DD/YYYY
 - Passwords: strings with at least one upper case letter, one lower case letter, and one digit
- Choose all matched strings of the given regular expression.
 - (1) a(bc)+de
A) abcde B) ade C) abcbcd D) abc
 - (2) [a-z\s]*hello
A) Othello B) hello C) 2hello D) say hello