

# Computer Systems and Architecture

## Regular Expressions

Stephen Pauwels

Academic Year 2021-2022

# Outline

- What is a Regular Expression?
- Tools
- Anchors, Character sets and Modifiers
- Advanced Regular Expressions



# Regular Expressions

- A regular expression is a pattern that describes a set of strings
- Search and manipulate text based on patterns
- Flexible and powerful
- Three parts
  - **Anchors**: specify the position of the pattern in relation to a line of text
  - **Character** sets: match one or more characters in a single position
  - **Modifiers**: specify how many times the previous character set is repeated



# Anchors

- `^` (beginning of line) and `$` (end of line)
- Examples:

<code>^A</code>	“A” at the beginning of a line
<code>A\$</code>	“A” at the end of a line
<code>A^</code>	“A^” anywhere on a line
<code>\$A</code>	“\$A” anywhere on a line
<code>^^</code>	“^” at the beginning of a line
<code>\$\$</code>	“\$” at the end of a line

# Character Sets

- Simplest character set:
  - `abc` matches the character sequence `abc`
  - `.` represents any single character
- Ranges:
  - Between `[` and `]`: one of these characters/patterns
  - `[^` and `]`: NOT one of these characters/patterns
  - Use `-` between characters to denote a range between these characters
- Want to use literal characters with a special meaning?
  - “Escape” with backslash `\`
  - `\.` matches a `“.”`
  - `\*` matches an asterix
  - `{`, `}`, `(`, `)`, `<`, `>` don’t have a special meaning

# Character sets

[A-Z]	Any capital letter
[A-Za-z]	Any letter
[]	The characters “[”
[0]	The character “0”
[0-9]	Any number
[^0-9]	Any character other than a number
[-0-9]	Any number or a “-”
[0-9-]	Any number or a “-”
[^-0-9]	Any character except a number or a “-”
[]0-9]	Any number or a “[”
[0-9]]	Any number followed by a “[”

# Modifiers

- Combining character sets:
  - `^T[a-z][aeiou]`
  - Matches a line that starts with T, followed by a letter and a vowel
- Use modifiers to repeat character sets

<code>[0-9]*</code>	Matches zero or more numbers
<code>[0-9][0-9]*</code>	Matches one or more numbers
<code>[0-9]\{5\}</code>	Matches five numbers
<code>[0-9]\{5,8\}</code>	matches five to eight numbers
<code>[0-9]\{5,\}</code>	matches five or more numbers

- Match only words: use `\<` and `\>`
  - Surrounding characters are anything but a letter, number, underscore
  - `\<[tT]he\>` matches any line with the word the or The

# Backreferences

- Reuse patterns: remember what you found earlier
  - Mark pattern with `\(` and `\)`
  - Refer to previously marked patterns with `\1`, `\2`, `\3`, ...
- Examples

<code>\([a-z]\)\1</code>	Matches two identical letters
<code>\&lt;\([a-z]\)[a-z]*\1\&gt;</code>	Matches every word that starts and ends with the same letter
<code>\([a-z]\)\([a-z]\)[a-z]\2\1</code>	Matches every 5-letter palindrome



# Tools

- Grep
  - Print lines matching a pattern
- Sed
  - Read and modify the input stream as specified by a pattern
- Awk
  - More advanced string handling



# Grep

- `grep 'class' /usr/share/dict/words`
  - Print all words that contain the string 'class'
- `grep '^class' /usr/share/dict/words`
  - Print all words that begin with the string 'class'
- `grep 'class$' /usr/share/dict/words`
  - Print all words that end with the string 'class'
- `grep '^c..ss$' /usr/share/dict/words`
  - Print all 5-letter words that begin with 'c' and end with 'ss'
- `grep '^c.*ss^' /usr/share/dict/words`
  - Print all words that begin with 'c' and end with 'ss'



# Sed

- `sed 's/from/to/g'`
  - Replace all occurrences of regex *from* to *to*
- Substitute command:
  - `s`: Substitute
  - `/.../.../`: Delimiter
  - `from`: Regular expression
  - `to`: Replacement string
  - `g`: Flags
- Usage:
  - `cat oldfile.txt | sed 's/from/to/'`
  - `sed 's/from/to/' < oldfile.txt`
  - `sed 's/from/to/' < oldfile.txt > newfile.txt`



# Sed

- Other delimiters
  - `sed 's:/usr/local/bin:/home/bin:'`
  - `sed 's|/usr/local/bin|/home/bin|'`
- Use '&' as the matched string
  - `sed 's/[a-z]*/(&)/'`
    - places parenthesis around a string
- Using '\1', '\2' ... to keep part of the pattern

# Sed Options

- `sed -e`: combine options
  - `sed -e 's/a/A/' -e 's/b/B/'`
- `sed -f`: read commands from script file
- `sed -n`: silent mode



# Sed Flags

- What to do when there is more than one occurrence of pattern on a single line?
  - `/.../.../`: Only the first occurrence is replaced
  - `/.../.../g`: Global replacement
  - `/.../.../3`: Replace the third occurrence
  - `/.../.../2g`: Replace but the first occurrence
  - `/.../.../p`: Print modified lines
    - `sed -n 's/pattern/&/p'` duplicates the function of `grep`
  - `/.../.../w filename`: Write all modified lines to filename



# Extended Regular Expressions

- Used by `egrep` and `awk`
- `?` matches 0 or 1 instances of the character set before
- `+` matches 1 or more instances of the character set before
- `\{`, `\}`, `\(`, `\)`, `\<`, `\>` no longer have special meaning
- `^(Ruben|Pieter)` matches every line that starts with “Ruben” or “Pieter”



# Exercises

- **Course webpage**
  - <http://msdl.uantwerpen.be/people/hv/teaching/ComputerSystemsArchitecture/#CS2>