Computer Systems and Architecture

Regular Expressions

Ruben Van den Bossche
Original slides by Bart Meyers

University of Antwerp

Outline

What?

Tools

Anchors, character sets and modifiers

Advanced Regular expressions

Exercises

Outline

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

Outline

Tools

- Grep (`grep`)
  - Print lines matching a pattern
- Sed (`sed`)
  - Read and modify the input stream as specified by a pattern.
- Awk (`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 occurences of regex *from* by *to*
- Substitute command:
  - `s` : Substitute command
  - `/../../` : 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|'`
- Using '&' 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 commands
  - `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 occurence of a pattern on a single line?
  - `/../../` : Only the first occurrence of *from* is replaced
  - `/../../g` : Global replacement
  - `/../../3` : Replace the third occurrence
  - `/../../2g` : Replace all but the first occurrence
  - `/../../p` : Print the modified lines
    - `sed -n 's/pattern/&/p'` duplicates the function of `grep`
  - `/../../w filename` : Write all modified lines to filename

Outline

Anchors

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

`^A`	“A” at the beginning of a line
`A$`	“A” at the end of a line
`A^`	“A^” anywhere on a line
`$A`	“$A” anywhere on a line
`^^`	“^” at the beginning of a line
`$$`	“$” 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 “]”
`[0-9-z]`	Any number, or any character between “9” and “z”

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
  - `[0-9]*` matches zero or more numbers
  - `[0-9][0-9]*` matches one or more numbers
  - `[0-9]\{5\}` matches five numbers
  - `[0-9]\{5,8\}` matches five to eight numbers
  - `[0-9]\{5,\}` matches five or more numbers
- Match only words: use `\<` and `\>`
  - Surrounding characters are anything but a letter, number, underscore, new line or end of line
  - `\<[tT]he\>` matches any line with the word the or The.

Outline

Backreferences

- Reuse patterns: remember what you found earlier
  - Mark pattern with `\(` and `\)`
  - Refer to previously market patterns with `\1`, `\2`, `\3`...
- Examples
  - `\([a-z]\)\1` matches two identical letters.
  - `\<\([a-z]\)[a-z]*\1\>` matches every word that starts and ends with the same letter.
  - `\([a-z]\)\([a-z]\)[a-z]\2\1` matches every 5-letter palindrome.

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 a special meaning
- `^(Ruben|Pieter)` matches every line that starts with "Ruben" or "Pieter"

Outline

Exercises

- http://msdl.cs.mcgill.ca/people/hv/teaching/ComputerSystemsArchitecture/#csw3