CMSC 330: Organization of Programming Languages

Regular Expressions, Part 2

Regular Expression Coding Readability

```
> ls -l
drwx------   2 sorelle  sorelle  4096 Feb 18 18:05 bin
-rw-r--r--   1 sorelle  sorelle   674 Jun  1 15:27 calendar
drwx------   3 sorelle  sorelle  4096 May 11 12:19 cmsc330
drwxrw----   2 sorelle  sorelle  4096 Jun  4 17:31 cmsc351
drwx----wx   1 sorelle  sorelle  4096 May 30 19:19 cmsc630
drwx------   1 sorelle  sorelle  4096 May 30 19:20 cmsc631
```

What if we want to specify the format of the output of ls –l exactly?

```
/^(d|-)(r|-)(w|-)(x|-)(r|-)(w|-)(x|-)(r|-)(w|-)(x|-)
\s+\d+\s+\w+\s+\w+\s+\d+\s+(Jan|Feb
|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec)\s+\d\d
\s+\d\d:\d\d\s+\S+$/
```

This is unreadable!

Regular Expression Coding Readability

Instead, we can do each part of the expression separately and then combine them:

```
oneperm_re = '((r|-)(w|-)(x|-))'
permissions_re = '(d|-)' + oneperm_re + '{3}'
month_re = '(Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec)'
day_re = '\d{1,2}';   time_re = '(\d{2}:\d{2})'
date_re = month_re + '\s+' + day_re + '\s+' + time_re
total_re = '\d+';  user_re = '\w+';   group_re = '\w+'
space_re = '\d+';  filename_re = '\S+'

line_re = Regexp.new('^' + permissions_re + '\s+' + total_re
      + '\s+' + user_re + '\s+' + group_re + '\s+' +
      space_re + '\s+' + date_re + '\s+' + filename_re + '$')

if (line =~ line_re) then
  puts("found it!")
end
```


Extracting Substrings Based on r.e.'s

- Can be done using the String scan method, or backreferences
- Backreferences:
  - Ruby remembers or captures the parts of strings that match parenthesized parts of regular expressions
  - The captured substrings can be referred to using special global variables (called backreferences) named $1, $2,…
  - Examples:
    - `/^Status: (.*)/`
      - capture all chars to the right on lines beginning with "Status"
    - `/^Min: (\d+)  Max: (\d+)$/`
      - remember digits following “Min” and digits following “Max”

Backreference Example

- Extract information from a report

```
gets() =~ /^Min: (\d+)  Max: (\d+)$/
min, max = $1, $2
```

  - Note parallel assignment above
- Warning- despite their names, $1 etc are local variables

```
def m(s)
  s =~ /(banana)/
  puts $1   # prints banana
end

m("banana")
puts $1     # prints nil
```


Another Backreference Example

- Warning #2- if another search is performed, all backreferences are **reset** to nil

```
gets() =~ /(h)e(ll)o/
puts($1)
puts($2)
gets() =~ /h(e)llo/
puts($1)
puts($2)
gets() =~ /hello/
puts($1)
```

```
hello
h
ll
hello
e
nil
hello
nil
```


The scan Method

- Also extracts substrings based on regular expressions
- Can optionally use parentheses in regular expression to affect how the extraction is done
- Has two forms that differ in what Ruby does with the matched substrings
  - The first form returns an array
  - The second form uses a code block

First Form of the scan Method

- *str*.scan(*regexp*)
  - if *regexp* doesn't contain any parenthesized subparts, returns an array of matches (an array of **all** the substrings of *str* that matched)

```
s = "CMSC 330 Fall 2012"
s.scan(/\d+/)  # returns array [330, 2012]
```

First Form of the Scan Method (cont.)

- If regexp **does** contain parenthesized subparts, returns an array of arrays
  - Each subarray contains the parts of the string that matched one occurrence of the search

```
s = "CMSC 330 Fall 2012"
s.scan(/(\S+) (\S+)/)  # [["CMSC", "330"],
                       #  ["Fall", "2012"]]
```

  - Each subarray has the same number of entries as the number of parenthesized subparts
  - All strings that matched the first part of the search (or $1 in backreference terms) are located in the first position of each subarray

Practice with Scan and Backreferences

```
> ls -l
drwx------    2 sorelle  sorelle  4096 Feb 18 18:05 bin
-rw-------    1 sorelle  sorelle  674 Jun  1 15:27 calendar
drwx------    3 sorelle  sorelle  4096 May 11  2006 cmsc330
drwx------    2 sorelle  sorelle  4096 Jun  4 17:31 cmsc351
drwx------    1 sorelle  sorelle  4096 May 30 19:19 cmsc630
drwx------    1 sorelle  sorelle  4096 May 30 19:20 cmsc631
```

Extract just the file or directory name from a line using

- scan

```
name = line.scan(/\S+$/)  # ["bin"]
```

- backreferences

```
if line =~ /(\S+$)/
  name = $1   # "bin"
end
```


Second Form of the scan Method

- *str*.scan(*regexp*) { |*match*| *block* }
  - applies the code block to each match
  - short for *str*.scan(*regexp*).each() { |*match*| *block* }
  - the regular expression can also contain parenthesized subparts

Example of Second Form of scan

```
sum_a = sum_b = sum_c = 0
while (line = gets())
  line.scan(/(\d+)\s+(\d+)\s+(\d+)/) { |a, b, c|
    sum_a += a.to_i()
    sum_b += b.to_i()
    sum_c += c.to_i()
  }
end
printf("Total: %d %d %d\n", sum_a, sum_b, sum_c)
```

Sums up three columns of numbers

```
12  34  23
19  77  87
11  98  3
2   45  0
```

input file:
will be read line by line, but column summation is desired