

COMP284 Scripting Languages

Lecture 5: Perl (Part 4)
Handouts (8 on 1)

Ullrich Hustadt

Department of Computer Science
School of Electrical Engineering, Electronics, and Computer Science
University of Liverpool

Regular expressions (2)

Capture groups

Regular expressions: Capture groups

Via **capture variables** the strings matched by a **capture group** are also available outside the pattern in which they are contained

<code>\$N</code>	string matched by capture group <i>N</i> (where <i>N</i> is a natural number)
<code>\${name}</code>	string matched by a named capture group

The matched strings are available until the end of the enclosing code block or until the next successful match

Example:

```
$_ = "Yabba_dabba_doo";  
if (/((?<c1>\w)(?<c2>\w)\g{c2}\g{c1}))/ {  
    print "Match found: $_$1\n" }
```

Output:

```
Match found: abba
```

COMP284 Scripting Languages

Lecture 5

Slide L5 – 4

Contents

- 1 Regular expressions (2)
 - Capture groups
 - Alternations
 - Anchors
 - Modifiers
 - Binding operator

Regular expressions (2)

Alternations

Regular expressions: Alternations

- The regular expression *regexpr1*|*regexpr2* matches if either *regexpr1* or *regexpr2* matches
This type of regular expression is called an **alternation**
- Within a larger regular expression we need to enclose alternations in a capture group or non-capturing group:
(*regexpr1*|*regexpr2*) or (?:*regexpr1*|*regexpr2*)

Examples:

```
1 /Mr|Ms|Mrs|Dr/  
2 /cat|dog|bird/  
3 (?:Bill|Hillary|Clinton)/
```

COMP284 Scripting Languages

Lecture 5

Slide L5 – 1

Regular expressions (2)

Capture groups

Regular expressions: Capture groups and backreferences

- We often encounter situations where we want to identify the repetition of the same or similar text, for example, in HTML markup:

```
<strong> ... </strong>  
<li> ... </li>
```

- We might also not just be interested in the repeating text itself, but the text between or outside the repetition
- We can characterise each individual example above using regular expressions:

```
<strong>.*</strong>  
<li>.*</li>
```

but we cannot characterise both without losing fidelity, for example:

```
<\w+>.*</\w+>
```

does not capture the 'pairing' of HTML tags

Regular expressions: Alternations

- The regular expression *regexpr1*|*regexpr2* matches if either *regexpr1* or *regexpr2* matches

Example:

```
1 $_ = "cats_and_dogs";  
2 if (/cat|dog|bird/) { print "Match 1: $_$1\n" }  
3 if (/dog|cat|bird/) { print "Match 2: $_$1\n" }  
4 if (/dog|dogs/) { print "Match 3: $_$1\n" }  
5 if (/dogs|dog/) { print "Match 4: $_$1\n" }
```

Output:

```
Match 1: cat  
Match 2: cat  
Match 3: dog  
Match 4: dogs
```

COMP284 Scripting Languages

Lecture 5

Slide L5 – 6

Regular expressions (2)

Capture groups

Regular expressions: Capture groups

The solution are **capture groups** and **backreferences**

<i>(regexpr)</i>	creates a capture group
<i>(?<name>regexpr)</i>	creates a named capture group
<i>(?: regexpr)</i>	creates a non-capturing group
<i>\N, \gN, \g{N}</i>	backreference to capture group <i>N</i> (where <i>N</i> is a natural number)
<i>\g{name}</i>	backreference to a named capture group

Examples:

```
1 /Sorting took (\d+.\d+) seconds/  
2 /<(\w+>.*</\1>/  
3 /([A-Z])0{2}(\d+)/  
4 /(?<c1>\w)(?<c2>\w)\g{c2}\g{c1}/  
5 /((?<c1>\w)(?<c2>\w)\g{c2}\g{c1})/
```

Regular expressions (2)

Anchors

Regular expressions: Anchors

Anchors allow us to fix where a match has to start or end

<code>\A</code>	Match only at string start
<code>^</code>	Match only at string start (default) Match only at a line start (in <code>//m</code>)
<code>\Z</code>	Match only at string end modulo a preceding \n
<code>\z</code>	Match only at string end
<code>\$</code>	Match only at string end modulo a preceding \n Match only at a line end (in <code>//m</code>)
<code>\b</code>	Match word boundary (between <code>\w</code> and <code>\W</code>)
<code>\B</code>	Match except at word boundary

Example:

```
$_ = "The_girl_who_nplayed_with_fire\n";  
if (/fire\z/) { print "fire' at string end\n" }  
if (/fire\Z/) { print "fire' at string end modulo \n\n" }  
if (/fire_\z/) { print "fire' at string end modulo \n\n" }
```

COMP284 Scripting Languages

Lecture 5

Slide L5 – 3

COMP284 Scripting Languages

Lecture 5

Slide L5 – 7

Regular expressions (2)

Modifiers

Regular expressions: Modifiers

Modifiers change the interpretation of certain characters in a regular expression or the way in which Perl finds a match for a regular expression

/ /	Default '.' matches any character except '\n' '^' matches only at string start '\$' matches only at string end modulo preceding \n
/ /s	Treat string as a single long line '.' matches any character including '\n' '^' matches only at string start '\$' matches only at string end modulo preceding \n
/ /m	Treat string as a set of multiple lines '.' matches any character except '\n' '^' matches at a line start '\$' matches at a line end

COMP284 Scripting Languages

Lecture 5

Slide L5 – 8

Regular expressions (2)

Modifiers

Regular expressions: Modifiers (/ /g and / /c)

The **current position** in a string for a regular expression *regexpr* is associated with the string, not *regexpr*

~ different regular expressions for the same strings will move forward the same position when used with / /g

~ different strings have different positions and their respective positions move forward independently

Example:

```
$_ = "ab11cd22ef33";  
if (/d+/g) { print "Match starts at $_-[0]: $_$&\n" }  
if (/a-z+/g) { print "Match starts at $_-[0]: $_$&\n" }  
if (/d+/g) { print "Match starts at $_-[0]: $_$&\n" }
```

Output:

```
Match starts at 3: 11  
Match starts at 6: cd  
Match starts at 9: 22
```

COMP284 Scripting Languages

Lecture 5

Slide L5 – 12

Regular expressions (2)

Modifiers

Regular expressions: Modifiers

Modifiers change the interpretation of certain characters in a regular expression or the way in which Perl finds a match for a regular expression

/ /sm	Treat string as a single long line, but detect multiple lines '.' matches any character including '\n' '^' matches at a line start '\$' matches at a line end
/ /i	perform a case-insensitive match

Example:

```
$_ = "bill\nClinton";  
if (/Bill|Hillary)/i { print "Match: $_\n" }
```

Output:

```
Match: bill  
Clinton
```

COMP284 Scripting Languages

Lecture 5

Slide L5 – 9

Regular expressions (2)

Modifiers

Regular expressions: Modifiers (/ /g and / /c)

A failed match or changing the target string resets the position

```
1 $_ = "ab11cd22ef33";  
2 if (/d+/g) { print "2: Match starts at $_-[0]: $_$&\n" }  
3 if (/ab/g) { print "3: Match starts at $_-[0]: $_$&\n" }  
4 if (/d+/g) { print "4: Match starts at $_-[0]: $_$&\n" }
```

Output:

```
2: Match starts at 3: 11  
4: Match starts at 3: 11
```

To prevent the reset, an additional modifier / /c can be used

```
1 $_ = "ab11cd22ef33";  
2 if (/d+/g) { print "2: Match starts at $_-[0]: $_$&\n" }  
3 if (/ab/gc) { print "3: Match starts at $_-[0]: $_$&\n" }  
4 if (/d+/g) { print "4: Match starts at $_-[0]: $_$&\n" }
```

Output:

COMP284 Scripting Languages

Lecture 5

Slide L5 – 10

Regular expressions (2)

Modifiers

Regular expressions: Modifiers (/ /g and / /c)

Often we want to process all matches for a regular expression, but the following code has not the desired effect

```
$_ = "112233";  
while (/d+/) { print "Match starts at $_-[0]: $_$&\n" }
```

The code above does not terminate and endlessly prints out the same text:

```
Match starts at 0: 11
```

To obtain the desired behaviour of the while-loop we have to use the / /g modifier:

/ /g	In scalar context , successive invocations against a string will move from match to match, keeping track of the position in the string In list context , returns a list of matched capture groups, or if there are no capture groups, a list of matches to the whole regular expression
------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

COMP284 Scripting Languages

Lecture 5

Slide L5 – 10

Regular expressions (2)

Binding operator

Generating regular expressions on-the-fly

The **current position** in a string for a regular expression *regexpr* is associated with the string, not *regexpr*

~ different regular expressions for the same strings will move forward the same position when used with / /g

~ different strings have different positions and their respective positions move forward independently

Example:

```
$_ = "Bart teases Lisa";  
@keywords = ("bart", "lisa", "marge", 'L\\w+', 't\\w+');  
while ($keyword = shift(@keywords)) {  
    print "Match found for $_$keyword: $_$&\n" if /$keyword/i;  
}
```

Output:

```
Match found for bart: Bart  
Match found for lisa: Lisa  
Match found for L\\w+: Lisa  
Match found for t\\w+: teases
```

COMP284 Scripting Languages

Lecture 5

Slide L5 – 14

Regular expressions (2)

Modifiers

Regular expressions: Modifiers (/ /g and / /c)

With the / /g modifier our code works as desired:

```
$_ = "112233";  
while (/d+/g) { print "Match starts at $_-[0]: $_$&\n" }
```

Output:

```
Match starts at 0: 11  
Match starts at 3: 22  
Match starts at 6: 33
```

An example in a **list context** is the following:

```
$_ = "ab11cd22ef33";  
@numbers = (/d+/g);  
print "Numbers: ", join(" | ", @numbers), "\n";
```

Output:

```
Numbers: 11 | 22 | 33
```

Read / /g as: Start to look for a match from the position where the last match using / /g ended

COMP284 Scripting Languages

Lecture 5

Slide L5 – 11

Regular expressions (2)

Binding operator

Binding operator

Perl offers two **binding operators** for regular expressions

<i>string</i> =~ / <i>regexpr</i> /	true iff <i>regexpr</i> matches <i>string</i>
<i>string</i> !~ / <i>regexpr</i> /	true iff <i>regexpr</i> does not match <i>string</i>

- Note that these are similar to **comparison operators** not **assignments**
- Most of the time we are not just interested whether these expressions return true or false, but in the side effect they have on the special variables **\$N** that store the strings matched by **capture groups**

Example:

```
$name = "DrUllrichHustadt";  
if ($name =~ /(Mr|Ms|Mrs|Dr)?\s*(\w+)/) {print "Hello $_$2\n"}  
$name = "DaveShield";  
if ($name =~ /(Mr|Ms|Mrs|Dr)?\s*(\w+)/) {print "Hello $_$2\n"}  
  
Hello Ullrich  
Hello Dave
```

COMP284 Scripting Languages

Lecture 5

Slide L5 – 15

Pattern matching in a list context

- When a pattern match `/regexpr/` is used in a **list context**, then the return value is
 - a list of the strings matched by the capture groups in `regexpr` if the match succeeds and `regexpr` contains capture groups, or
 - (a list containing) the value 1 if the match succeeds and `regexpr` contains no capture groups, or
 - an empty list if the match fails

```
$name = "Dr_Ullrich_Hustadt";
($t,$f,$l) = ($name =~ /(Mr|Ms|Mrs|Dr)?\s*(\w+)\s+(\w+)/);
print "Name: $t, $f, $l\n";
$name = "Dave_Shield";
($t,$f,$l) = ($name =~ /(Mr|Ms|Mrs|Dr)?\s*(\w+)\s+(\w+)/);
print "Name: $t, $f, $l\n";
```

Output:

```
Name: Dr, Ullrich, Hustadt
Name: , Dave, Shield
```

Pattern matching in a list context

- When a pattern match `/regexpr/g` is used in a **list context**, then the return value is
 - a list of the strings matched by the capture groups in `regexpr` each time regex matches provided that `regexpr` contains capture groups, or
 - a list containing the string matched by `regexpr` each time `regexpr` matches provided that `regexpr` contains no capture groups, or
 - an empty list if the match fails

```
$string = "firefox:10.3seconds;chrome:9.5seconds";
%performance = ($string =~ /(\w+)\s*:(\d+\.\d+)/g);
foreach $system (keys %performance) {
    print "$system->performance($system)\n";
}
```

Output:

```
firefox -> 10.3
chrome -> 9.5
```

Revision

Read

- Chapter 7: In the World of Regular Expressions
- Chapter 8: Matching with Regular Expressions

of

R. L. Schwartz, brian d foy, T. Phoenix:
Learning Perl.
O'Reilly, 2011.

- <http://perldoc.perl.org/perlre.html>
- <http://perldoc.perl.org/perlretut.html>
- <http://www.perlfect.com/articles/regextutor.shtml>

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro