Regular Expressions
Scripting and Computer Environment - Lecture 3

Saurabh Barjatiya

International Institute Of Information Technology, Hyderabad

26 July, 2011

Contents

1. Basics
   - Matching single character
   - Using special characters
   - Repetition match
2. Miscellaneous
   - OR of regular expressions
   - Backreference

Sample text file

For all future examples following will be contents of sample text file (say test.txt)

Contents

```
ABCD
XYZ
xyz
abcd

340
6$
there are two elf's in this room
there are two shelves in this room
  341 + 24 = 356
!@#$%
```

Matching single character - 01

All alphabets and numbers match themselves. Hence nothing fancy is required to match simple words or numbers.

Example 1 - Matching alphabets and numbers

```
Command: grep this test.txt

Output:
there are two elf's in this room
there are two shelves in this room
```

Matching single character - 02

Following characters are treated differently when used in regular expressions:

- Plus (‘+’)
- Pipe (‘|’)
- Brackets (‘{}[]()’)
- Dot (‘.’)
- Question mark (‘?’)
- Caret (‘^’)
- Dollar (‘$’)
- Backslash (‘\’)

Matching single character - 03

Special characters can be escaped using backslash (‘\’) so that they match themselves and do not have any special significance.

Example 2 - Matching special characters

```
Command: grep '\$' test.txt

Output:
6$
!@#$%
```

Using special characters - Brackets - 01

Square brackets ('[]') are used to specify range / set of characters that can be matched.

Example 3 - Matching set of characters

```
Command: grep '[iou]' test.txt

Output:
abcd
there are two elf's in this room
there are two shelves in this room
```

Using special characters - Brackets - 02

Example 4 - Matching range of characters

```
Command: grep '[0-9]' test.txt

Output:
340
6$
  341 + 24 = 356
```

Using special characters - Brackets - 03

`[a-z]` ⇒ Any small alphabet between a and z (both inclusive)

`[A-Z]` ⇒ Any capital alphabet between A and Z (both inclusive)

`[a-Z]` ⇒ Any alphabet capital or small

`[A-z]` ⇒ Wont match with anything, not even capital A or small z

Note that above description is based on grep behavior. Other regular expression matching programs or libraries can interpret the same sequences differently. Hence it is best to test regular expression when using new library / application or use long form like '[abcdefghijklmnopqrstuvwxyz]' to ensure that regular expression is portable.

Using special characters - Brackets - 04

Following and many other named classes can be used within square brackets '([])':

- `[:alpha:]` ⇒ Any alphabet small or capital
- `[:alnum:]` ⇒ Any alphabet or number
- `[:digit:]` ⇒ Any digit
- `[:lower:]` ⇒ Any lower case letter
- `[:upper:]` ⇒ Any upper case letter
- `[:space:]` ⇒ Whitespace

Usage of named classes

```
Command: grep '[[:alpha:]]' test.txt
```

The symbol '\w' is a synonym for [[:alnum:]]

Using special characters - Brackets - 05

If square brackets range begins with caret (‘^’) then it negates the range and all characters which are not specified within square brackets get matched.

Example 05 - Negation of range with square brackets

```
Command: grep '[^[:alpha:] ]' test.txt

Output:
340
6$
there are two elf's in this room
  341 + 24 = 356
!@#$%
```

The symbol ‘\W’ is a synonym for [^[:alnum:]]

Using special characters - Description

Period ('.') $\Rightarrow$ Matches any single character

Caret ('^') $\Rightarrow$ Matches empty character at start of line

Dollar ('$') $\Rightarrow$ Matches empty character at end of line

('\<') $\Rightarrow$ Matches empty character at beginning of word

('\>') $\Rightarrow$ Matches empty character at end of word

('\b') $\Rightarrow$ Matches empty character at edge of word

Using special characters - Period

Example 06 - Usage of period

```
Command: grep '6.' test.txt

Output:
6$
```

Using special characters - Caret

Example 07 - Usage of caret

```
Command: grep '^a' test.txt

Output:
abcd
```

Using special characters - Dollar

Example 08 - Usage of dollar

```
Command: grep '6$' test.txt

Output:
   341 + 24 = 356
```

Using special characters - \<

Example 09 - Usage of \<

```
Command: grep '\<s' test.txt

Output:
there are two elf's in this room
there are two shelves in this room
```

Using special characters - \>

Example 10 - Usage of \>

```
Command: grep 's\>' test.txt

Output:
there are two elf's in this room
there are two shelves in this room
```

Using special characters - \b

Example 11 - Usage of \b

```
Command: grep '\b3' test.txt

Output:
340
  341 + 24 = 356
```

Repetition match - Description

? ⇒ The preceding item is optional and matched at most once.

* ⇒ The preceding item will be matched zero or more times.

+ ⇒ The preceding item will be matched one or more times.

{n} ⇒ The preceding item is matched exactly n times.

{n,} ⇒ The preceding item is matched n or more times.

{n,m} ⇒ The preceding item is matched at least n times, but not more than m times.

Repetition match examples

Example 12 - Usage of ?

```
Command: grep -o '\<t...\?\>' test.txt

Output:
two
this
two
this
```

Repetition match examples

Example 13 - Usage of *

```
Command: grep -o 'b.*' test.txt

Output:
bcd
```

Repetition match examples

Example 14 - Usage of +

```
Command: grep -o 'th[a-z]\+' test.txt

Output:
there
this
there
this
```

Repetition match examples

Example 15 - Usage of {}

```
Commands: grep -o 'ro\{2\}m' test.txt
          grep -o 'ro\{1,\}m' test.txt
          grep -o 'ro\{1,2\}m' test.txt

Output:
room
room
```

Contents

1. Basics
   - Matching single character
   - Using special characters
   - Repetition match
2. Miscellaneous
   - OR of regular expressions
   - Backreference

OR of regular expressions

AND on regular expressions can be performed by just concatenating two regular expressions. For OR of regular expressions '|' operator can be used.

Example 16 - Usage of |

```
Commands: grep -o 's[a-z]\+\|[a-z]\+s' test.txt

Output:
this
shelves
this
```

Backreference

Backreferences can be used to refer to matched regular expression when using regular expressions for search and replace operations.

Example 17 - Usage of backreference

```
Commands: sed 's/\<i\([a-z]\+\)\>/I\1/g' test.txt | grep I

Output:
there are two elf's In this room
there are two shelves In this room
```