Introduction to Regular Expressions

Regular Expressions

A regular expression (regex or regexp for short) is a special text string which defines a set of one or more strings of characters. The power in regular expressions is that using letters, numbers, and special characters a single regex may define **many** different strings (A regex is said to match any string it defines).

Regular expression are widely used in GNU/Linux and many programming languages.
UNIX/Linux: emacs, VI, ed, grep, sed, AWK
Languages: Perl, PHP, Python, JavaScript, Java, Ruby, any .NET programming language

Why should we care?
If you wished to validate (almost all) email addresses submitted via an online form, what would you do? Remember, you wish to separate:
joe@@mercyhurst.edu or
joe.mercyhurst@edu or
@mercyhurst.edu or
joe@math@mercyhurst.edu or
joe@.mercyhurst.edu from a valid email address such as:
joe123@math.mercyhurst.edu
With regex one can use:

Validate an Email Address

```
/^[a-z0-9\._-]+@[a-z0-9\._-]+\.+[a-z]{2,4}$/i
or
/^\w+([\.-]?\w+)*@\w+([\.-]?\w+)+(\.+[a-z]{2,4})+$/
```

Let's see how.

By the way, there is a nice tool for working with Regular Expressions: *Kodos*
In Kodos, you do not need delimiters in the Regular Expressions Pattern.

Characters

Before we begin, I should remind you that regular expression are case sensitive, so 'a' is different than 'A'.
A *character* in this context is any single literal character except a NEWLINE. A *special character* is one that does not just represent itself - it has special meaning. Some special characters are : the dot (period) . , square brackets [ ], the asterisk *, the caret ˆ, the dollar sign $, the backslash question mark ?, the plus sign +, parentheses ( ), the pipe symbol |.

Delimiters

A character called a *delimiter* is used to mark the beginning and end of a regular expression. The forward slash / is used as the default in many settings.

regex	Matches	Examples
/ring/	ring	ring, spring, pouring
/or not/	or not	or not, poor nothing

Characters Classes and class-like constructs

A regular expression always matches the longest possible string, starting as far toward the beginning of the line as possible.

class	Meaning	..same as
[. . .]	A character class - matches any single character The hyphen (-) is used to create a range of characters	
[^ . . .]	will negate the character class	
.	matches any single character	
\w	Part-of-word character	[a-zA-Z0-9_]
\W	Non-word character	[^a-zA-Z0-9_]
\d	Digit	[0-9]
\D	Non-digit	[^0-9]
\s	Whitespace character	
\S	Non-whitespace character	

Escaping Special Characters

You may escape any special character (not a parenthesis) by preceding it with a backslash. Escaping a special character makes it represent itself.

Anchors and other zero-width tests

There are a tremendous number of metacharacters available in regular expressions. Many we have already used in shell scripts; **\n** for newline and **\t** for a tab. But there are also useful class short hands.

class	Meaning
^	(Caret) forces a match at the beginning of the string.
$	forces a match at the end of the string.
\b	Word boundary, matches only at word boundaries (beginning/end)
\B	Word non-boundary, matches at non word boundaries

Mode Modifiers

class	Meaning
i	Case insensitive match
g	Search for all matches (globally)

Grouping, control

Grouping subpatterns and capturing submatches is done using parentheses. Text matched by the subpattern within parentheses is captured for later use. We will mention grouping but not cover capturing.

sequence	Meaning
(. . .)	Group subpattern (capturing parenthesis) followed by a ? marks optional
(. | .)	The pipe character inside parentheses functions as a logical OR
*	Match 0 or more times
+	Match 1 or more times
?	Match 0 or 1 times following a match quantifier marks minimal (few as possible)
{n}	match exactly n times
{n,}	match at least n times
{n,m}	match at least n times but no more than m times
(?:)	Non-capturing parenthesis

Notes

Linux users, I might suggest the very nice utility “Kodos” a regular expression debugger.