Regular expression syntax

This article is a simple explanation document to help user set **SnmpSvcs** service configuration file. It covers the basic regular expression syntax used by SnmpSvcs configure file.

Literals

All characters are literals except: ".", "*", "?", "+", "(", ")", "{", "}", "[", "]", "^" and "$". These characters are literals when preceded by a "\".

Wildcard

The dot character "." matches any single character except : the dot does not match a null character; then the dot does not match a newline character.

Repeats

A repeat is an expression that is repeated an arbitrary number of times. An expression followed by "*" can be repeated any number of times including zero. An expression followed by "+" can be repeated any number of times, but at least once. An expression followed by "?" may be repeated zero or one times only. When it is necessary to specify the minimum and maximum number of repeats explicitly, the bounds operator "{}" may be used, thus "a{2}" is the letter "a" repeated exactly twice, "a{2,4}" represents the letter "a" repeated between 2 and 4 times, and "a{2,}" represents the letter "a" repeated at least twice with no upper limit. Note that there must be no white-space inside the {}, and there is no upper limit on the values of the lower and upper bounds. All repeat expressions refer to the shortest possible previous sub-expression: a single character; a character set, or a sub-expression grouped with "()" for example.

Examples:

"ba*" will match all of "b", "ba", "baaa" etc.

"ba+" will match "ba" or "baaaa" for example but not "b".

"ba?" will match "b" or "ba".

"ba{2,4}" will match "baa", "baaa" and "baaaa".

Parenthesis

Parentheses serve two purposes, to group items together into a sub-expression, and to mark what generated the match. For example the expression "(ab)*" would match all of the string "abababab". In the example above reg_match[1] would contain a pair of iterators denoting the final "ab" of the matching string. It is permissible for sub-expressions to match null strings. Sub-expressions are indexed from left to right starting from 1, sub-expression 0 is the whole expression.

Alternatives

Alternatives occur when the expression can match either one sub-expression or another, each alternative is separated by a "|".

Examples:

"a(b|c)" could match "ab" or "ac".

"abc|def" could match "abc" or "def".

Sets

A set is a set of characters that can match any single character that is a member of the set. Sets are delimited by "[" and "]" and can contain literals, character ranges, character classes, collating elements and equivalence classes. Set declarations that start with "^" contain the compliment of the elements that follow.

Examples:

Character literals:

"[abc]" will match either of "a", "b", or "c".

"[^abc] will match any character other than "a", "b", or "c".

Character ranges:

"[a-z]" will match any character in the range "a" to "z".

"[^A-Z]" will match any character other than those in the range "A" to "Z".

Note that character ranges are highly locale dependent: they match any character that collates between the endpoints of the range, ranges will only behave according to ASCII rules when the default "C" locale is in effect.

Line anchors

An anchor is something that matches the null string at the start or end of a line: "^" matches the null string at the start of a line, "$" matches the null string at the end of a line.

Escape operator

The escape character "\" has several meanings.

Inside a set declaration the escape character is a normal character.

The escape operator may make the following character normal, for example "*" represents a literal "*" rather than the repeat operator.

Single character escape sequences

The following escape sequences are aliases for single characters:

Escape sequence	Character code	Meaning
\a	0x07	Bell character.
\f	0x08	Form feed.
\n	0x0A	Newline character.
\r	0x0D	Carriage return.
\t	0x09	Tab character.
\v	0x0B	Vertical tab.
\e	0x1B	ASCII Escape character.
\0dd	0dd	An octal character code, where *dd* is one or more octal digits.
\xXX	0xXX	A hexadecimal character code, where

		XX is one or more hexadecimal digits.
\x{XX}	0xXX	A hexadecimal character code, where XX is one or more hexadecimal digits, optionally a unicode character.
\cZ	z-@	An ASCII escape sequence control-Z, where Z is any ASCII character greater than or equal to the character code for '@'.

What gets matched?

The regular expression library will match the first possible matching string, if more than one string starting at a given location can match then it matches the longest possible string. In cases where their are multiple possible matches all starting at the same location, and all of the same length, then the match chosen is the one with the longest first sub-expression, if that is the same for two or more matches, then the second sub-expression will be examined and so on.