CHAPTER 7

Regular Expressions for Soft Properties Manager

This section is based on the documentation of the package GNU RegExp.

A regular expression consists of a character string in which some characters are given special meaning with regard to pattern matching. Regular expressions have been in use from the early days of computing, and provide a powerful and efficient way to parse, interpret, and search and replace text within an application.

Topics include:

Supported Syntax

Within a regular expression, the following characters have special meaning.

Positional Operators

Character	Description
^	Matches at the beginning of a line.
$	Matches at the end of a line.
\A	Matches the start of the entire string.
\Z	Matches the end of the entire string.
\b	Matches at a word break (Perl5 syntax only).
\B	Matches at a nonword break (opposite of \b) (Perl5 syntax only).
\<	Matches at the start of a word (egrep syntax only).
\>	Matches at the end of a word (egrep syntax only).

One-Character Operators

Character	Description
.	Matches any single character.
\d	Matches any decimal digit.
\D	Matches any nondigit.
\n	Matches a newline character.
\r	Matches a return character.
\s	Matches any whitespace character.
\S	Matches any nonwhitespace character.
\t	Matches a horizontal tab character.
\w	Matches any word (alphanumeric) character.
\W	Matches any nonword (alphanumeric) character.
x	Matches the character *x*, if *x* is not one of the above listed escape sequences.

Character Class Operator

Character	Description
[*abc*]	Matches any character in the set *a*, *b*, or *c*.
[^*abc*]	Matches any character not in the set *a*, *b*, or *c*.
[*a-z*]	Matches any character in the range *a* to *z*, inclusive.
Leading or trailing dash	Interpreted literally.

Within a character class expression, the following sequences have special meaning if the syntax bit RE_CHAR_CLASSES is enabled:

Character Sequences	Description
[:alnum:]	Any alphanumeric character.
[:alpha:]	Any alphabetic character.
[:blank:]	A space or horizontal tab.
[:cntrl:]	A control character.
[:digit:]	A decimal digit.
[:graph:]	A nonspace, noncontrol character.
[:lower:]	A lowercase letter.
[:print:]	Same as graph, but also space and tab.
[:punct:]	A punctuation character.
[:space:]	Any whitespace character, including newline and return.

Character Sequences	Description
[:upper:]	An uppercase letter.
[:xdigit:]	A valid hexadecimal digit.

Subexpressions and Back References

Expression	Description
(*abc*)	Matches whatever the expression *abc* would match, and saves it as a subexpression. Also used for grouping.
(?:...)	Pure grouping operator; does not save contents.
(?#...)	Embedded comment; ignored by engine.
n	Where 0 < *n* < 10, matches the same thing the *n*th subexpression matched.

Branching (Alternation) Operator

Expression	Description
a|*b*	Matches whatever the expression *a* or *b* would match.

Repeating Operators

These symbols operate on the previous atomic expression:

Symbols	Description
?	Matches the preceding expression or the null string.
*	Matches the null string or any number of repetitions of the preceding expression.
+	Matches one or more repetitions of the preceding expression.
{*m*}	Matches exactly *m* repetitions of the one-character expression.
{*m*,*n*}	Matches between *m* and *n* repetitions of the preceding expression, inclusive.
{*m*,}	Matches *m* or more repetitions of the preceding expression.

Stingy (Minimal) Matching

If a repeating operator (above) is immediately followed by a question mark (?), the repeating operator stops at the smallest number of repetitions that can complete the rest of the match.

Lookahead

Lookahead refers to the ability to match part of an expression without consuming any of the input text. There are two variations to this:

Expression	Description
(?=*foo*)	Matches at any position where *foo* would match, but does not consume any characters of the input.
(?!*foo*)	Matches at any position where *foo* would not match, but does not consume any characters of the input.

Unsupported Syntax

Some flavors of regular expression utilities support additional escape sequences. The following is not meant to be an exhaustive list. In the future, gnu.regexp might support some or all of the following:

Expression	Description
(?mods)	Inlined compilation/execution modifiers (Perl5).
\G	End of previous match (Perl5).
[.*symbol*.]	Collating symbol in class expression (POSIX).
[=*class*=]	Equivalence class in class expression (POSIX).
s/foo/bar	Style expressions as in sed and awk.