A Regular Expression Primer

This table briefly compares the regular expression capability of Perl, grep, egrep, and lex. This table does not include minimal matching, casefolding, and pattern substitution features of the various regular expression languages in each tool. Any pattern matching of lex also works in flex.

chars	action	Perl	grep	egrep	lex/flex
abc...	Match that character (metacharacters excluded)	Perl	grep	egrep	lex/flex
\\\.*...	Match that metacharacter following the back slash	Perl	uses sh	uses sh	lex/flex
""	De-meta any chars inside quotes				lex/flex
\t,\n,\r,\f	tab, newline, return, form feed	Perl	grep	egrep	lex/flex
.	Match any character	Perl	grep	egrep	lex/flex (not \n)
[]	Character class	Perl	grep	egrep	lex/flex
[^]	Inverse Character class	Perl	grep	egrep	lex/flex
[-]	Character ranges	Perl	grep	egrep	lex/flex
\w	Match a "word" character (alphanumeric plus "_")	Perl			
\W	Match a non-word character	Perl			
\s	Match a whitespace character	Perl			
\S	Match a non-whitespace character	Perl			
\d	Match a digit character	Perl			
\D	Match a non-digit character	Perl			
anchors	**action**	**Perl**	**grep**	**egrep**	**lex/flex**
^	Match the beginning of the line	Perl	grep	egrep	lex/flex
$	Match the end of the line	Perl	grep	egrep	lex/flex
\b	Match a word boundary	Perl			
\B	Match a non-(word boundary)	Perl			

operators	action	Perl	grep	egrep	lex/flex
|	Alternation	Perl		egrep	lex/flex
()	Grouping	Perl		egrep	lex/flex
multiplicity	**action**	**Perl**	**grep**	**egrep**	**lex/flex**
*	Greedy match 0 or more times	Perl	grep	egrep	lex/flex
+	Greedy match 1 or more times	Perl		egrep	lex/flex
?	Greedy match 1 or 0 times	Perl	grep	egrep	lex/flex
{n}	Greedy match exactly n times	Perl		egrep	
{n,}	Greedy match at least n times	Perl		egrep	
{n,m}	Greedy match at least n but not more than m times	Perl		egrep	lex/flex
lookAhead	**action**	**Perl**	**grep**	**egrep**	**lex/flex**
/	Look ahead predicate				lex/flex
(?=)	Look ahead predicate	Perl			
(?!)	NOT Look ahead predicate	Perl			

Greedy means that it trys to find the longest pattern that matches from the point where the previous pattern left off.

Some Examples of Regular Expressions

dog	matches the string "dog"
[dog]	matches matches one character: a "d" an "o" or a "g"
[dog]*	matches matches a string of zero or more characters from the set {"d" an "o" or a "g"}
(dog|cat)	matches the string "dog" or the string "cat"
dog.*cat	matches the string "dog" followed by the string "cat" somewhere later in the string

x(dog|cat)x	matches the string "dog" or the string "cat" between two "x"s
xx*	matches a string of one or more "x"s
x+	matches a string of one or more "x"s
x(dog|cat)?x	matches two "x"s with optionally the string "dog" or the string "cat" between the "x"'s
[aeiou]	matches a single vowel
[A-Z]+	matches a string of one or more uppercase characters
[az-]+	matches a string of one characters from the set or three characters "a", "z", "-"
[^a-z]+	matches a string of one or more characters that are **not** lowercaase letters
"[a-z]"	in **flex** matches exactly the five character string "[a-z]"
[a-zA-Z][a-zA-Z0-9]*	matches a letter optionally followed by letters or digits
[1-9][0-9]*|0	matches a positive integer with no leading zero except when the number is zero
[+-]?[0-9]+	matches an integer with optional sign (note that leading zeroes are allowed
([0-9].)*	matches an even number of characters where every odd numbered character is a digit
[+-]?[1-9][0-9]*|0	matches an integer with no leading zero except when the number is zero. The number may have an optional sign
[\^\+\-\:*\]]	matches one of the 6 characters: "^", "+", "-", ":", "*", "]"