Regular expressions

“Regular expressions” are a powerful tool in Linux for searching for patterns – like wild card characters for filenames (eg *.txt means all files with a .txt extension) but much more powerful. See http://www.regular-expressions.info/quickstart.html for more information, and links to very detailed explanations. One complicating element in the use of regular expressions is that many different programs use them, but the syntax can vary from one program to another, so the meaning of some regular expressions can be different in one situation versus another.

In a regular expression, some characters have special meanings: ^, $, \, ., [], {}, ?, *, +, |, (). These are called metacharacters, because they have meaning for the search functions. In the paragraphs below, each of these is highlighted in bold font and the special meaning is explained. Other characters are taken literally in the search query – an “a” in the search query means “search for an a” – but metacharacters are not taken literally unless special steps are taken.

The ^ symbol means the beginning of a line, so ^@ is a regular expression meaning “search for the @ character at the beginning of a line”. The $ symbol means the end of a line.

The \ symbol is used to specify some non-printing characters as well as classes of printing characters. For example, \t means a tab character, \r means a line return, \n means a newline, and \s means any whitespace character (space, tab, return, or newline). For character classes, \d means any digit, while \w means a word (any number of consecutive letters, numbers, or underscore characters with a whitespace character before and after) and \b is a word boundary (any character that is not part of a word). A capital letter means the opposite of the lower-case letter for these character classes, so \D is any character that is not a digit and \S is any character that is not whitespace. These special symbols can be used in character sets (see below), but be careful with the upper-case versions. For example, [\d\s] means any character that is either a digit or a whitespace character, but [\D\S] means any character that is either “not a digit” or “not a whitespace”. A digit is not a whitespace, and whitespace is not a digit, so [\D\S] matches any character. NOTE: Windows uses \r\n to denote end-of-line, Unix and Linux use \n, and Mac uses \r, so text files created on one operating system often fail to work as expected on a different type of system.

The . symbol matches any character except a newline character (\n) under most circumstances.

Square brackets [] are used to denote character sets – multiple characters, any one of which may be present. For example, [aeiou] matches a single character that can be any one of the letters a, e, i, o, or u. A hyphen can be used to specify a range, so [a-z] means any lower-case letter. Ranges and single characters can be combined in a character set, so [a-zA-Z:_;] means any one lower-case or upper-case letter or :, _, or ;. If a metacharacter is to be included in a character set or search string, it must be preceded by \, to tell the search engine to look for the literal character instead of using the metacharacter meaning. For example, to look for all punctuation marks that might occur at ends of sentences (period, question mark, or exclamation point), the search string would be [\.\?!], because . and ? are metacharacters that would be interpreted differently without the “escape” character (backslash). So how do we search for a backslash? \\ A ^ symbol after the opening square bracket means "everything except the specified character set", so if we want to find an occurrence of any non-alphanumeric character (not a letter or a number), we can use [^a-zA-Z0-9]. To include a hyphen in a character set, it must occur first – for example, [-a-zA-Z] would match any single character that is either a hypen, a lower-case letter or an upper-case letter.

Curly brackets {} containing a number following a literal character, character set, or group (see below) specifies the number of consecutive occurrences of that character or group, or characters from the character set. To search for a string of any three digits, we can use [0-9]{3} to match character strings from 000 to 999. Including two numbers in the curly brackets sets a minimum and maximum number of occurrences, so [0-9]{1,3} would match one, two, or three digit strings. One number followed by a comma inside the curly brackets specifies a minimum number but no maximum number, so [0-9]{3,} would match strings of at least three digits, with no limit on the length of the string matched. **The + symbol** means "one or more of the preceding character", so [0-9]+ will find strings consisting of one or more digits, up to the length of a line. **The ? symbol** means "zero or one occurrence of the previous character", so it makes a particular character optional in the search string. **The * symbol** means "zero or more occurrences of the previous character". *NOTE: regular expressions use "greedy matching" for repetition, which means the search engine will match as much text as possible by default. For example, using the expression <.+> to search for any opening HTML tag (such as <u><B></u> in the string <B>this text is bold</B>), the matched string will be the entire string <u><B>this text is bold</B></u>, because it begins and ends with < and >, the specified characters. The .+ symbol matched as much text as possible, so it did not stop with the first >, but included everything up to the second >. To prevent this, you can specify the regular expression as <[^>]+> to tell the search engine to find a < character, followed by any number of characters that are not >, followed by a > character.*

The pipe symbol | is a logical OR – the search engine will search for each expression in a series of expressions separated by | symbols, from left to right, and return a match to the first expression that works. For example, \b(cat|dog|fish)\b would search for the words 'cat', 'dog', or 'fish'. The expression fish|fishing would match 'fish' , even if the string is 'fishing', because 'fish' occurs first in the left-to-right sequence of expressions.

Parentheses () are grouping symbols that can be used to indicate a set of characters that are to be treated as a unit, and can also be used to specify characters for replacement. For example, the expression Set(Value)? would match either Set or SetValue, because the ? symbol makes the group of characters Value optional (zero or one occurrence) in the search. In functions that allow replacement, up to nine sets of parentheses can be specified and the strings matched can be used in replacement by specifying \1 through \9. For example, the expression (XYZ\d+)(:[A-Z]+) will find the string XYZ followed by one or more digits followed by a colon and one or more upper-case letters. In a replace function, the expression \2\1 would insert the colon and upper-case letters first, followed by the XYZ and digits.