Regular expressions

Character	BRE / ERE	Meaning in a pattern
\	Both	Usually, turn off the special meaning of the following character. Occasionally, enable a special meaning for the following character, such as for \(...\) and \{...\}.
.	Both	Match any single character except NUL. Individual programs may also disallow matching newline.
*	Both	Match any number (or none) of the single character that immediately precedes it. For EREs, the preceding character can instead be a regular expression. For example, since . (dot) means any character, **.*** means "match any number of any character." For BREs, * is not special if it's the first character of a regular expression.
^	Both	Match the following regular expression at the beginning of the line or string. BRE: special only at the beginning of a regular expression. ERE: special everywhere.

Regular Expressions (cont'd)

$	Both	Match the preceding regular expression at the end of the line or string. BRE: special only at the end of a regular expression. ERE: special everywhere.
[…]	Both	Termed a bracket expression, this matches any one of the enclosed characters. A hyphen (-) indicates a range of consecutive characters. (Caution: ranges are locale-sensitive, and thus not portable.) A circumflex (^) as the first character in the brackets reverses the sense: it matches any one character not in the list. A hyphen or close bracket (]) as the first character is treated as a member of the list. All other metacharacters are treated as members of the list (i.e., literally). Bracket expressions may contain collating symbols, equivalence classes, and character classes (described shortly).
\{*n*,*m*\}	BRE	Termed an *interval expression*, this matches a range of occurrences of the single character that immediately precedes it. \{*n*\} matches exactly n occurrences, \{*n*,\} matches at least n occurrences, and \{*n*,*m*\} matches any number of occurrences between n and m. n and m must be between 0 and RE_DUP_MAX (minimum value: 255), inclusive.
\(\)	BRE	Save the pattern enclosed between \(and \) in a special *holding space*. Up to nine subpatterns can be saved on a single pattern. The text matched by the subpatterns can be reused later in the same pattern, by the escape sequences \1 to \9. For example, **\(ab\).*\1** matches two occurrences of ab, with any number of characters in between.

Regular Expressions (cont'd)

\n	BRE	Replay the nth subpattern enclosed in \(and \) into the pattern at this point. n is a number from 1 to 9, with 1 starting on the left.
{n,m}	ERE	Just like the BRE \{*n*,*m*\} earlier, but without the backslashes in front of the braces.
+	ERE	Match one or more instances of the preceding regular expression.
?	ERE	Match zero or one instances of the preceding regular expression.
|	ERE	Match the regular expression specified before or after.
( )	ERE	Apply a match to the enclosed group of regular expressions.

POSIX Bracket Expressions

Class	Matching characters	Class	Matching characters
[:alnum:]	Alphanumeric characters	[:lower:]	Lowercase characters
[:alpha:]	Alphabetic characters	[:print:]	Printable characters
[:blank:]	Space and tab characters	[:punct:]	Punctuation characters
[:cntrl:]	Control characters	[:space:]	Whitespace characters
[:digit:]	Numeric characters	[:upper:]	Uppercase characters
[:graph:]	Nonspace characters	[:xdigit:]	Hexadecimal digits

Backreferences

- Match whatever an earlier part of the regular expression matched
  - Enclose a subexpression with \(and \).
  - There may be up to 9 enclosed subexpressions and may be nested
  - Use \digit, where digit is a number between 1 and 9, in a later part of the same pattern.

Pattern	**Matches**
\(ab\)\(cd\)[def]*\2\1	abcdcdab, abcdeeecdab, abcdddeeffcdab, ...
\(why\).*\1	A line with two occurrences of why
\([[:alpha:]_][[:alnum:]_]*\) = \1;	Simple C/C++ assignment statement

Matching Multiple Characters with One Expression

*	Match zero or more of the preceding character
\{*n*\}	Exactly n occurrences of the preceding regular expression
\{*n*,\}	At least n occurrences of the preceding regular expression
\{*n*,*m*\}	Between n and m occurrences of the preceding regular expression

Anchoring text matches

Pattern	Text matched (in bold) / Reason match fails
ABC	Characters 4, 5, and 6, in the middle: abc**ABC**defDEF
^ABC	Match is restricted to beginning of string
def	Characters 7, 8, and 9, in the middle: abcABC**def**DEF
def$	Match is restricted to end of string
[[:upper:]]\{3\}	Characters 4, 5, and 6, in the middle: abc**ABC**defDEF
[[:upper:]]\{3\}$	Characters 10, 11, and 12, at the end: abcDEFdef**DEF**
^[[:alpha:]]\{3\}	Characters 1, 2, and 3, at the beginning: **abc**ABCdefDEF

Operator Precedence (High to Low)

Operator	Meaning
[. .] [= =] [: :]	Bracket symbols for character collation
metacharacter	Escaped metacharacters
[ ]	Bracket expressions
\(\) *digit*	Subexpressions and backreferences
* \{ \}	Repetition of the preceding single-character regular expression
no symbol	Concatenation
^ $	Anchors

Examples

Match a string that has **ab followed by one or more c**

Answer : abc+

Match a string that has **ab followed by 2 c**

Answer : abc{2}

Match a string that has **ab followed by 2 or more c**

Answer : abc{2,}

Match a string that has **ab followed by at least 2 and at most 5 c**

Answer : abc{2,5}

Match a string that has **a followed by at least 2 at most 5 copies of the sequence bc**

Answer : a(bc){2,5}

Examples

Match a string that has **a followed by b or c**

Answer: a(b|c)

or: a[bc]

Match a string that has a character **from 0 to 9 before a % sign**

Answer : [0-9]%

Match a string that **does not have a letter from a to z or from A to Z**

Answer : [^a-zA-Z]

Examples

Difference between [.] and (.)

[.] will match exactly the ‘.’ character.

eg: .

(.) will match any one character.

eg: a, b, 1

sed

- Now you can extract, but what if you want to replace parts of text?
- Use sed!

```
sed 's/regExpr/replText/'
```

- Example

```
sed 's/:.*//' /etc/passwd      Remove everything
                after the first colon
```