REGULAR EXPRESSIONS

This page is an excerpt of the UNIX manual page **regexp**(5) that defines the syntax and semantics of regular expressions used by the **seq**(LOCAL) format of CADP, the **exhibitor**(LOCAL) tool, and the **OPEN/CAESAR** application programming interfaces **caesar_hide_1**(LOCAL) and **caesar_rename_1**(LOCAL).

REGULAR EXPRESSIONS

A regular expression specifies a set of character strings. A member of this set of strings is said to be matched by the regular expression. Some characters have special meaning when used in a regular expression; other characters stand for themselves.

The following *one-character RE*s match a *single* character:

1.1 An ordinary character (*not* one of those discussed in 1.2 below) is a one-character RE that matches itself.

1.2 A backslash (****) followed by any special character is a one-character RE that matches the special character itself. The special characters are:

a. **.**, *, **[** , and **** (period, asterisk, left square bracket, and backslash, respectively), which are always special, *except* when they appear within square brackets (**[ ]**; see 1.4 below).

b. ^ (caret or circumflex), which is special at the *beginning* of an *entire* RE (see 4.1 and 4.3 below), or when it immediately follows the left of a pair of square brackets (**[ ]**) (see 1.4 below).

c. **$** (dollar sign), which is special at the **end** of an *entire* RE (see 4.2 below).

d. The character used to bound (that is, delimit) an entire RE, which is special for that RE (for example, see how slash (**/**) is used in the **g** command, below.)

1.3 A period (**.**) is a one-character RE that matches any character except new-line.

1.4 A non-empty string of characters enclosed in square brackets (**[ ]**) is a one-character RE that matches *any one* character in that string. If, however, the first character of the string is a circumflex (^), the one-character RE matches any character *except* new-line and the remaining characters in the string. The ^ has this special meaning *only* if it occurs first in the string. The minus (**–**) may be used to indicate a range of consecutive characters; for example, **[0–9]** is equivalent to **[0123456789]**. The **–** loses this special meaning if it occurs first (after an initial ^, if any) or last in the string. The right square bracket (**]**) does not terminate such a string when it is the first character within it (after an initial ^, if any); for example, **[ ]a–f]** matches either a right square bracket (**]**) or one of the ASCII letters **a** through **f** inclusive. The four characters listed in 1.2.a above stand for themselves within such a string of characters.

The following rules may be used to construct REs from one-character REs:

2.1 A one-character RE is a RE that matches whatever the one-character RE matches.

2.2 A one-character RE followed by an asterisk (*) is a RE that matches **0** or more occurrences of the one-character RE. If there is any choice, the longest leftmost string that permits a match is chosen.

2.3 A one-character RE followed by **\{***m***\}**, **\{***m,***\}**, or **\{***m,n***\}** is a RE that matches a *range* of occurrences of the one-character RE. The values of *m* and *n* must be non-negative integers less than 256; **\{***m***\}** matches *exactly m* occurrences; **\{***m,***\}** matches *at least m* occurrences; **\{***m,n***\}** matches *any number* of occurrences *between m* and *n* inclusive. Whenever a choice exists, the RE matches as many occurrences as possible.

2.4 The concatenation of REs is a RE that matches the concatenation of the strings matched by each component of the RE.

2.5 A RE enclosed between the character sequences **\(** and **\)** is a RE that matches whatever the unadorned RE matches.

2.6 The expression *****n* matches the same string of characters as was matched by an expression enclosed between **\(** and **\)** *earlier* in the same RE. Here *n* is a digit; the sub-expression specified is that beginning with the *n*-th occurrence of **\(** counting from the left. For example, the expression **^\(.*\)\1$** matches a line consisting of two repeated appearances of the same string.

A RE may be constrained to match words.

3.1 **\<** constrains a RE to match the beginning of a string or to follow a character that is not a digit, underscore, or letter. The first character matching the RE must be a digit, underscore, or letter.

3.2 **\>** constrains a RE to match the end of a string or to precede a character that is not a digit, underscore, or letter.

An *entire RE* may be constrained to match only an initial segment or final segment of a line (or both).

4.1 A circumflex (**^**) at the beginning of an entire RE constrains that RE to match an *initial* segment of a line.

4.2 A dollar sign (**$**) at the end of an entire RE constrains that RE to match a *final* segment of a line.

4.3 The construction **^***entire RE***$** constrains the entire RE to match the entire line.

The null RE (for example, **//**) is equivalent to the last RE encountered.

CHARACTERS WITH SPECIAL MEANING

Characters that have special meaning except when they appear within square brackets (**[ ]**) or are preceded by **** are: **.**, *****, **[**, ****. Other special characters, such as **$** have special meaning in more restricted contexts.

The character **^** at the beginning of an expression permits a successful match only immediately after a newline, and the character **$** at the end of an expression requires a trailing newline.

Two characters have special meaning only when used within square brackets. The character **-** denotes a range, **[***c***-***c***]**, unless it is just after the open bracket or before the closing bracket, **[-***c***]** or **[***c***-]** in which case it has no special meaning. When used within brackets, the character **^** has the meaning *complement of* if it immediately follows the open bracket (example: **[^***c***]**); elsewhere between brackets (example: **[***c***^]**) it stands for the ordinary character **^**.

The special meaning of the **** operator can be escaped only by preceding it with another ****, for example ****.