What is the \& pattern in Vim's Regex

Ask Question

I recently came across the branch specifier in Vim regex builtins. Vim's help section on \& contains this:

11 this

4

```
A branch is one or more concats, separated by concat, but only if all the preceding concats position. Examples:
    "foobeep\&..." matches "foo" in "foobeep
    ".*Peter\&.*Bob" matches in a line conta
```

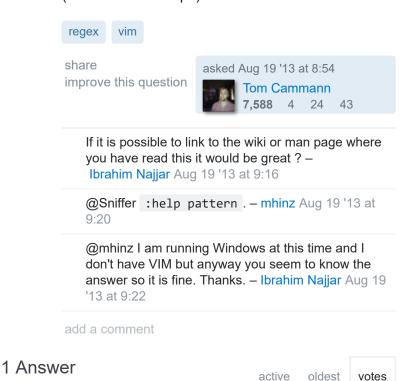
It's not clear how it is used and what it is used for. A good explanation of what it does and how it is used would be great.

To be clear this is **not** the & (replace with whole match) used in a substitution, this is the \& used in a pattern.

Example usage:

```
/\c\v([^aeiou]&\a){4}
```

Used to search for 4 consecutive consonants (Taken from vim tips).



Explanation:

14

\& is to \| , what the *and* operator is to the *or*operator. Thus, both concats have to match, but only the last will be highlighted.

Example 1:

(The following tests assume :setlocal hlsearch .)

Imagine this string:

foo foobar

Now, /foo will highlight foo in both words. But sometimes you just want to match the foo in foobar. Then you have to use /foobar\&foo.

That's how it works anyway. Is it often used? I haven't seen it more than a few times so far. Most people will probably use zero-width atoms in such simple cases. E.g. the same as in this example could be done via /foo\zebar.

Example 2:

 $/\c\v([^aeiou]&\a){4}$.

\c - ignore case

\v - "very magic" (-> you don't have to escape
the & in this case)

(){4} - repeat the same pattern 4 times

[^aeiou] - exclude these characters

\a - alphabetic character

Thus, this, rather confusing, regexp would match xxxx, XXXX, wXyZ or wxYz but not AAAA or xxx1. Putting it in simple terms: Match any string of 4 alphabetic characters that doesn't contain either 'a', 'e', 'i', 'o' or 'u'.

share

edited Aug 19 '13 at 10:09

improve this answer

answered Aug 19 '13 at 9:15



2,288 13 18

_

@Sniffer Note: in all cases concat may be replaced with zero-width positive look-ahead (a\&b\&c is always \%(a\)\@=\%(b\)\@=c , wondering why you did not mention this, only have a few words about zero-width atoms). Look-aheads/look-behinds are more powerful then concats and it makes sense to get used to use only them because when you learn new regexp engine it is much more likely that it would support neither look-aheads nor concats or would support only look-aheads rather then have any support for concats. – ZyX Aug 19 '13 at 14:34 \bullet

Also note that \zs used as zero-width is buggy: try searching for \zso with foo in buffer and compare the result with \@<=o . Do not know a bug for \ze though. - ZyX Aug 19 '13 at 14:39

@ZyX Thanks for the clarification. I don't work with VIM and don't know what it supports but mhinz explanation of operator seemed pretty logical so I up-voted his answer. – Ibrahim Najjar Aug 19 '13 at 14:43

@mhinz After some thought I do not think that behavior is a bug.
But .\@<=o matches two o's, .\zso , matches one o. This seems to be because one o is taken by the previous match and, unlike lookbehind, \zs is applied after both zero-width and non-zero-width parts matched. And it is impossible to have non-zero-width match at the position already taken by previous match. – ZyX Aug 19 '13 at 14:53</p>