Gilmation

RegExp - keeping parts of a pattern in sed

- **29 July, 2010**
- by Paco

Non-trivial use of regular expressions for pattern substitution often requires that a part of the pattern be kept after the substitution. The following sed examples show the basic usage of "variables" in regular expressions.

The **sed** *stream editor for filtering and transforming text* and **grep** *global regular expression print* are widely used tools because it's easy to get a good grasp of their basic functionality. Most use cases involve straightforward transformations but on occasions, like the ones described below, trickier transformations may prove to be necessary.

Referring to the matching pattern

In order to refer to the matching pattern the *&* variable is used. The following example adds "*the*" before each ordinal number:

```
$ echo "Is this 34th street or 35th then?" | sed 's/\([0-9][0-9]*th\)/the &/g'
Is this the 34th street or the 35th then?
```

"*[0-9][0-9]**" matches any number with at least one digit. Notice how neither "*this*" nor "*then*" is transformed.

Keeping part of the pattern

The following example finds words surrounded by the same number (*ab* is surrounded by *1* and *ef* is surrounded by *82*) and removes the surrounding number:

```
$ echo "4 9 1 ab 1 cd 13 82 ef 82" | sed 's/\([0-9]*\) \([a-z]*\) \1/\2 /g'
4 9 ab cd 13 ef
```

Notice that "*\1*" is placed in the pattern and refers to the expression "*([0-9]*)*". "*\2*" is placed in the substitution script and refers to "*([a-z]*)*".

Replace a given occurrence

Similarly, one might want to replace only one of the matching patterns. In that case, the index of the matching pattern can be placed in the third block of the **sed** expression. The following example aims to remove the second word of a sentence:

```
$ echo "Call me Ishmael" | sed 's/\([a-zA-Z]*\)//2'
Call  Ishmael
```

In this example "*[a-zA-Z]*" filters any word containing capital or non-capital letters.

One last note about sed

The most famous **sed** delimiter is the slash / character but notice that, according to the manual, the delimiter is the first character placed right after the command.

You can stick to slash usage because you can escape characters as shown in the following example:

```
$ echo "////Slashes / free text //" | sed 's/\///g'
Slashes  free text
```

But, if you know the nature of the text, it's easier to use a different delimiter:

```
$ echo "////Slashes / free text //" | sed 's:/::g'
Slashes  free text
```

Filed under: languages, tools

related articles

Contact

Gilmation S.L.

Calle 232, 66

La Cañada, Paterna

Valencia

46182

Spain

+34 637 250 751

hello@gilmation.com