macro in `vi`

mobytrick

11 settembre 2014

L'editor `vi` è molto potente perché mette a disposizione molti comandi, forse non tutti conosciuti. Ma ci sono delle situazioni in cui per ottenere un certo risultato bisogna eseguire una serie di comandi. La cosa diventa improponibile se l'operazione deve essere ripetuta svariate volte. In questo caso risulta conveniente costruire una *macro*: i comandi vengono immessi una sola volta e viene dato loro un nome. Richiamandolo opportunamente si ottiene l'esecuzione dei comandi correlati.

Preventivamente è necessario apprendere le nozioni di modalità d'uso dell'editor e quelle inerenti l'uso dei buffer.

In `vi` esistono 3 modalità:

1. *Command*: è la modalità di default quando si richiama l'editor. Ci si può spostare ed eseguire limitati comandi. Serve a passare nelle successive due modalità

2. *Insert*: i caratteri battuti sulla tastiera vengono scritti nel file. Per passare nella modalità Insert da quella Command si usa un comando per l'inserzione di testo. Per uscire si preme il tasto ESC(ape) e ci si ritrova nella modalità Command

3. *Command Line*: si entra in questa modalità da quella Command premendo il tasto `:` (doppio punto). Il cursore si sposta in basso a sinistra. È in questa modalità che l'editor mette a disposizione tutta la sua potenza, a patto che si abbia il necessario *know-how*. Si esce da questa modalità premendo il tasto ESC(ape).

L'editor mette a disposizione 27 (ventisette) buffer interni. Uno, anonimo, è quello di default. Viene usato abbastanza spesso, talvolta senza la piena consapevolezza dell'utilizzatore di `vi`. I rimanenti 26 hanno come nome 1

lettera. L'uso è a discrezione dell'utente che è consapevole della *volatilità* del contenuto. A fine sessione, quanto presente nei buffer va perso senza possibilità di recupero.
Il comando per utilizzare un buffer è il seguente:

```
"†
```

ove " (apice doppio) è il comando `vi` che permette l'uso dei buffer e † è la lettera che lo individua.
Le operazioni possibili sono:

scrittura
: nel buffer vengono depositati dei caratteri, che sono da indicare tramite la *metrologia* propria di `vi`. Ad esempio, una o più righe oppure dal cursore sino ad un marker posizionato in precedenza e via discorrendo. Attenzione al caso della lettera che individua il buffer. Se è minuscola, l'eventuale contenuto viene sovrascritto, mente se è maiuscola si ha l'accodamento del nuovo contenuto a quanto già presente

uso
: il contenuto del buffer viene ricopiato nel file che si sta editando. Il caso della lettere che individua il buffer è ininfluente

esecuzione
: il contenuto del buffer viene eseguito. Si postula che il buffer contenga dei comandi `vi`. Però il comando da usare non è " (apice doppio) bensì `@` (chiocciolina)

Alcuni esempi per familiarizzare l'uso dei buffer.

`"qY`
: la riga corrente viene copiata nel buffer `q` con distruzione dell'eventuale precedente contenuto

`"S4yy`
: 4 righe, a partire dall'attuale, vengono accodate al buffer `s`

`"zp`
: il contenuto del buffer `z` viene ricopiato a destra del cursore

`"fP`
: il contenuto del buffer `f` viene ricopiato a sinistra del cursore

Quale esempio vogliamo costruire una macro che aggiunga all'inizio di una riga i caratteri `/*` (i caratteri d'inizio commento nel linguaggio C). Tralasciando da parte le macro, per fare quanto ci si propone si procede nel seguente modo, battendo i caratteri indicati nella prima colonna della sottostante tabella. Rimane sottinteso che il file da modificare è stato aperto con l'editor.

`I`	inserzione ad inizio riga
`/*`	il testo che si desidera inserire

Per uscire dalla modalità Input premere il tasto ESC(ape). Si tenga presente che il comando `I` porta il cursore ad inizio riga, ovvero nel posto ove poi viene inserito il testo.
Per creare la macro si segue pressapoco lo stesso schema. Si aggiunge una riga di lavoro nella quale si riportanto sia i comandi che il testo già visti (ricordarsi che si è in modalità Insert). L'unica difficoltà consiste nel mimare a caratteri l'azione del testo ESC(ape). Si procede nel seguente modo. Premere la combinazione di tasti CTRL/V: si ottiene il carattere `^` (accento circonflesso, che appare di colore azzurrino). Quindi premere la combinazione di tasti CTRL/ESC(ape) e si ottiene il carattere `[` (parentesi quadra aperta, pure lei di colore azzurrino). Premere il tasto ESC(ape) per uscire dalla modalità Input. Sino ad ora i comandi sono stati solo scritti. Bisogna memorizzarli e dare loro un nome. Per fare ciò battere in modalità Command `"aY` seguìto da Invio. Poiché apparentemente non succede nulla ci vuole fede (o fiducia). Spiegazione:

`"`	comando `vi` che permette l'uso dei buffer
`a`	indica il buffer
`Y`	indica la riga corrente

L'allestimento è finito e la riga di appoggio va eliminata. La macro è stata memorizzata nel buffer `<a>`.
Per eseguirla portarsi su una riga qualsiasi e battere `@a` che significa: esegui i comandi inseriti nel buffer interno `<a>`. Ad inizio riga si potrà notare che è stato aggiunto quanto convenuto.
La macro testé creata è *volatile*, poiché ha validità solo nella corrente sessione di editing. Uscendo dall'editor non c'è modo di salvare il contenuto dei buffer. Per renderla permanente la si scrive nel file `~/.exrc`, che viene eseguito all'inizio di ogni sessione di `vi`[1].

[1] attenzione a non appesantire inutilmente il file di configurazione dell'editor

Una macro ha la seguente struttura:

- `map` parola chiave che introduce la macro
- `lhs` (**l**eft **h**and **s**ide). È il nome della macro: 1 solo carattere. Può essere:
  1. una lettera. Pratica sconsigliata, perché così quella lettera viene *bruciata*
  2. un carattere di controllo (ovvero una lettera preceduta da CTRL). Il carattere di controllo viene immesso premendo dapprima la combinazione di tasti CTRL/V (che fa comparire l'accento circonflesso (`^` in azzurrino) e poi la combinazione di tasti CTRL/† ove quest'ultimo rappresenta il carattere prescelto
  3. un tasto funzionale. Il numero prescelto viene preceduto dal cancelletto (`#`). Evitare il tasto funzionale 1. Sia Linux che Windows interpretano il suo uso come richiesta di aiuto prioritaria rispetto alla macro
- `rhs` (**r**ight **h**and **s**ide). È il *body* della macro, ovvero i comandi previsti

La macro predecentemente immessa in un buffer esterno viene modificata in modo da poterla richiamare tramite il tasto funzionale F2 in modalità Command (macro *offline*). Viene pure estesa, in modo da inserire a fine riga la fine del comment. Nel file `~./exrc` inserire in un'**unica** riga quanto compare nella prima colonna. Come separatore usare lo spazio.

`map`	introduce la macro
`#2`	indica il tasto funzione F2
`I/* ^[A */^[`	stesso significato della macro volatile

Ora in modalità Command premendo il tasto funzionale F2 la riga verrà commentata (linguaggio C).

È possibile creare delle macro utilizzabili anche quando si è nella modalità Insert (*inline* macro). È sufficiente aggiungere un `!` (punto escalamativo) alla parola chiave `map`.

Per elencare le macro disponibili, in modalità Command Line battere `map` (`map!` per quelle *inline*). Una macro può essere disattivata tramite il comando `unmap` seguito dal nome. L'operazione **non** è reversibile. Per *resuscitare* una macro disattivata si esce dall'editor e poi si rientra.