usr_26.txt Per **Vim version 8.2.** Ultima modifica: 2006 Apr 24

VIM USER MANUAL - di Bram Moolenaar
Traduzione di questo capitolo: Ivan Morgillo

Ripetizione

Un lavoro di editing è sempre crudamente privo di struttura. Una modifica spesso necessita di venire eseguita molte volte. In questo capitolo saranno spiegati diversi modi utili per ripetere un cambiamento.

==========

26.1 Ripetizioni in modo Visual

Il modo Visual è molto pratico per effettuare correzioni su qualsiasi numero di righe. Potete vedere il testo evidenziato, così potete verificare se siano state modificate le linee giuste. Ma effettuare la selezione richiede la pressione di alcuni tasti. Il comando "gv" seleziona nuovamente la stessa area di testo. Questo vi permette di fare un'altra operazione sullo stesso testo.

Supponiamo che abbiate alcune righe in cui vogliate sostituire "2001" con "2002" e "2000" con "2001":

```
The financial results for 2001 are better ~
than for 2000.  The income increased by 50%, ~
even though 2001 had more rain than 2000. ~
                2000            2001 ~
income          45,403          66,234 ~
```

Per prima cosa sostituite "2001" con "2002". Selezionate le righe in modo Visual, e usate: >

```
:s/2001/2002/g
```

Ora usate "gv" per riselezionare lo stesso testo. Non importa dove si trovi il cursore. Usate ":s/2000/2001/g" per effettuare la seconda modifica.

Ovviamente, potete ripetere queste sostituzioni diverse volte.

==========

26.2 Aggiungere e sottrarre

Quando ripetete il cambiamento di un numero in un altro, spesso avete uno scostamento fisso. Nell'esempio sopra, è stato aggiunto uno ad ogni anno. Invece di scrivere un comando diverso per ogni anno, si può usare il comando CTRL-A.

Usando lo stesso testo di prima, cercate un anno: >

```
/19[0-9][0-9]\|20[0-9][0-9]
```

Ora premete CTRL-A. L'anno sarà incrementato di uno:

```
The financial results for 2002 are better ~
than for 2000.  The income increased by 50%, ~
even though 2001 had more rain than 2000. ~
                2000            2001 ~
income          45,403          66,234 ~
```

Usate "n" per trovare l'anno successivo, e premente "." per ripetere il CTRL-A ("." è un po' più veloce da scrivere). Ripetete "n" e "." per tutti gli anni che ci sono.

Suggerimento: impostate l'opzione 'hlsearch' per vedere i valori che state

per modificare, poi potrete guardare oltre e farlo più velocemente.

L'aggiunta di numeri maggiori di uno può essere fatta premettendo un numero a CTRL-A. Supponete di avere la seguente lista:

```
1.  item four ~
2.  item five ~
3.  item six ~
```

Spostate il cursore su "1." e scrivete: >

```
3 CTRL-A
```

L'"1." cambierà in "4,". Ancora, potete usare "." per ripetere l'operazione sugli altri numeri.

Altro esempio:

```
006     foo bar ~
007     foo bar ~
```

Usando CTRL-A su questi numeri risulterà:

```
007     foo bar ~
010     foo bar ~
```

7 più uno fa 10? Questo è accaduto perché Vim ha riconosciuto "007" come un numero in rappresentazione ottale, perché il primo numero è uno zero. Questa notazione è spesso usata nei programmi C. Se non volete che un numero che inizia per zero sia gestito come un numero ottale, usate questo: >

```
:set nrformats-=octal
```

Il comando CTRL-X esegue le sottrazioni nello stesso modo.

==

26.3 Fare una modifica in più file

Supponiamo che abbiate una variabile di nome "x_cnt" e che vogliate cambiarla in "x_counter". Questa variabile viene usata in parecchi dei vostri file C. Dovrete sostituirla in tutti i file. Ecco come fare.

Mettete tutti i file interessati nella lista degli argomenti: >

```
:args *.c
```

<

Questo comando trova tutti i file C e modifica il primo. Ora potete eseguire un comando di sostituzione su tutti gli altri file: >

```
:argdo %s/\<x_cnt\>/x_counter/ge | update
```

Il comando ":argdo" prende come argomento un altro comando. Questo comando verrà eseguito su tutti i file nella lista degli argomenti.

Il comando substitute "%s" che segue lavora su tutte le righe. Esso trova la parola "x_cnt" come "\<x_cnt\>". Le "\<" e "\>" vengono usate per trovare solo la parola esatta e non "px_cnt" o "x_cnt2".

Le flag per il comando includono "g" per sostituire tutte le "x_cnt" che si presentano nella stessa riga. La flag "e" è usata per evitare un messaggio di errore nel caso in cui "x_cnt" non appaia affatto nel file. Diversamente ":argdo" abortirebbe la sua esecuzione se non venisse trovata "x_cnt" nel primo file.

Le "|" separano i due comandi. Il comando "update" che segue salva il file solo se è cambiato il suo contenuto. Se nessuna "x_cnt" viene cambiata in "x_counter", non succede niente.

C'è anche il comando ":windo", che esegue il suo argomento in tutte le finestre. E ":bufdo" che esegue il suo argomento in tutti i buffer. Fate attenzione con quest'ultimo perché potreste avere più file di quanti pensiate nella lista dei buffer. Fate un controllo con il comando ":buffers" (o ":ls").

==

26.4 Usare Vim da uno shell script

Supponiamo che abbiate molti file in cui avete bisogno di cambiare la stringa "-person-" con "Jones" e poi stamparla. Come lo fate? Un modo è scrivere molto. L'altro modo è scrivere uno shell script che faccia il lavoro.
L'editor Vim fa un lavoro superbo come editor screen-oriented quando si usano i comandi del modo Normal. Nei processi di batch, tuttavia, i comandi del modo Normal non risultano chiari; così qui userete invece il modo Ex. Questo modo fornisce una gradevole interfaccia a linea di comando che facilita l'immissione dei comandi in un file batch. ("Ex command" è solo un altro nome per un comando (:) da linea di comando).
I comandi in modo Ex di cui avete bisogno sono i seguenti: >

```
%s/-person-/Jones/g
write tempfile
quit
```

Inserite questi comandi nel file "change.vim". Ora per lanciare l'editor in modalità batch, usate questo shell script: >

```
for file in *.txt; do
  vim -e -s $file < change.vim
  lpr -r tempfile
done
```

Il ciclo for-done è un costrutto della shell per ripetere le due righe nel mezzo, sino a che la variabile $file risulta ogni volta impostata con un nome di file diverso.
La seconda riga lancia l'editor Vim in modo Ex (argomento -e) sul file $file e legge i comandi dal file "change.vim". L'argomento -s dice a Vim di operare in modo silenzioso. In altre parole, non visualizza il :prompt, né alcun altro prompt.
Il comando "lpr -r tempfile" stampa il risultante "tempfile" e lo cancella (ciò che fa l'argomento -r).

LEGGERE DA STDIN

Vim può leggere testo dallo standard input. Dato che normalmente qui vengono letti i comandi, voi dovete dire a Vim di leggere invece del testo. Ciò è possibile passando l'argomento "-" al posto di un file. Esempio: >

```
ls | vim -
```

Ciò vi permette di modificare l'output del comando "ls", senza prima salvare il testo in un file.
Se usate lo standard input per leggere del testo, potete usare l'argomento "-S" per leggere uno script: >

```
producer | vim -S change.vim -
```

SCRIPT E MODO NORMAL

Se veramente volete usare i comandi del modo Normal in uno script, potete farlo così: >

```
vim -s script file.txt ...
```
<

Nota:
"-s" ha un significato diverso quando viene usato senza "-e". Qui significa che utilizza il contenuto di "script" come un insieme di comandi del modo Normal.
Quando è usato con "-e" significa che esegue in silenzio e non usa l'argomento seguente come nome di un file.

I comandi nello script vengono eseguiti nel modo in cui li avete scritti. Non dimenticate che una riga vuota viene interpretata come premere <Enter>. In modo Normal esso sposta il cursore sulla riga successiva.
Per realizzare lo script potete creare il file script e scrivervi i comandi. Avete bisogno di immaginare quale possa essere il risultato, il che può essere un po' difficile. Un altro modo consiste nel registrare i comandi mentre li eseguite manualmente. Questo è quanto dovete fare: >

```
vim -w script file.txt ...
```

Tutti i tasti che verranno premuti saranno scritti nello "script". Se fate un piccolo sbaglio potete pure continuare e ricordarvi di modificare lo script in seguito.
L'argomento "-w" aggiunge tutto ad uno script esistente. Va bene quando volete registrare lo script un po' alla volta. Se voleste iniziare dal niente e scrivere tutto dall'inizio, usate l'argomento "-W". Sovrascriverà ogni file esistente.

==

Capitolo seguente: |**usr_27.txt**| Comandi di ricerca ed espressioni

Per segnalazioni scrivere a vimdoc.it at gmail dot com
oppure ad Antonio Colombo azc100 at gmail dot com