usr_04.txt   Per Vim version 7.0. Ultima modifica: 2006 Giu 21

VIM USER MANUAL - di Bram Moolenaar
Traduzione di questo capitolo: Bartolomeo Ravera

Fare piccole modifiche

Questo capitolo mostra diversi modi di effettuare correzioni e spostare il testo. Vi insegnerà i tre metodi di base per modificare il testo: operatore-movimento, Visual_mode e oggetti di testo.

  Capitolo seguente: |usr_05.txt| Configurazioni personali
Capitolo precedente: |usr_03.txt| Muoversi nel file
            Indice: |usr_toc.txt|

==
04.1 Operatori e spostamenti

Nel capitolo 2 avete imparato il comando "x" per cancellare un singolo carattere. E usando un contatore: "4x" cancella quattro caratteri.
   Il comando "dw" cancella una parola. Potete ricordare il comando "w" come il comando di movimento di una parola. In effetti, il comando "d" può essere seguito da ogni comando di movimento, e cancella dalla posizione attuale fino a quella in cui il cursore viene spostato.
   Il comando "4w", per esempio, muove il cursore di 4 parole. Il comando d4w cancella quattro parole.

```
        To err is human. To really foul up you need a computer. ~
                         ------------------>
                                 d4w

        To err is human. you need a computer. ~
```

Vim cancella solamente dalla posizione successiva a quella da cui si trova il cursore. Questo perché Vim sa che probabilmente non volete cancellare il primo carattere di una parola. Se usate il comando "e" per muovervi alla fine di una parola, Vim penserà che vogliate includere l'ultimo carattere:

Vim cancella soltanto sopra la posizione su cui il movimento porta il cursore. Ciò avviene perché Vim sa che voi probabilmente non intendete cancellare il primo carattere di una parola. Usando il comando "e" per muovere il cursore alla fine di una parola, Vim immagina che vogliate includere l'ultimo carattere:

```
        To err is human. you need a computer. ~
                         -------->
                            d2e

        To err is human. a computer. ~
```

Se il carattere che sta sotto il cursore viene incluso o no, dipende dal comando usato per muovervi verso tale carattere. Il manuale di riferimento lo chiama comando "esclusivo" quando il carattere non è incluso, e "inclusivo" quando lo è.

Il comando "$" sposta il cursore alla fine della linea. Il comando "d$" cancella dalla posizione del cursore sino alla fine della linea. Questo è un movimento inclusivo, quindi l'ultimo carattere della linea è incluso nell'operazione di cancellazione:

```
        To err is human. a computer. ~
                         ------------>
                              d$

        To err is human ~
```

C'è uno schema qui: operatore-movimento. Voi prima scrivete un comando operatore. Per esempio, "d" è l'operatore di cancellazione. Allora voi scrivete un comando di movimento, come "4l" o "w". Così potete operare su qualsiasi testo su cui possiate spostarvi.

==
04.2 Cambiare il testo

Un altro operatore è "c", cambio. Questo agisce come l'operatore "d", ad eccezione del fatto che vi lascia nella Insert_mode.
  Per esempio, "cw" cambia una parola. O più precisamente, cancella una parola e vi pone in Insert_mode.

```
        To err is human ~
           ------->
             c2wbe<Esc>

        To be human ~
```

Questo "c2wbe<Esc>" è composto da queste parti:

```
        c       l'operatore di cambio
        2w      sposta avanti di due parole (vengono cancellate e parte il
                Insert_mode)
        be      inserisce questo testo
        <Esc>   torna in Normal_mode
```

Se avete fatto attenzione, avrete notato qualcosa di strano: lo spazio prima di "human" non è stato cancellato. C'è un proverbio che dice che per ogni problema esiste una risposta semplice, chiara e sbagliata. Come in questo caso, per questo esempio relativo al comando "cw". Questo in realtà lavora esattamente come "ce", cambia fino alla fine di una parola. Quindi lo spazio dopo la parola non è incluso. Questa è una eccezione che risale al vecchio Vi. Poiché molte persone ci si sono ormai abituate questa incoerenza è rimasta in Vim.

ULTERIORI MODIFICHE

Come "dd" cancella un'intera linea, "cc" cambia un'intera linea. Questo comando conserva il rientro esistente (aggiungendo spazi bianchi).

Esattamente come "d$" cancella fino alla fine della linea, "c$" sostituisce fino alla fine della linea. E' come scrivere "d$" per cancellare il testo e poi usare "a" per dare inizio alla Insert_mode ed aggiungere nuovo testo.

SCORCIATOIE

Alcuni comandi operatore-movimento sono di così frequente uso che sono stati attribuiti loro comandi di una sola lettera:

```
        x  equivale a   dl  (cancella il carattere sotto il cursore)
        X  equivale a   dh  (cancella il carattere a sinistra del cursore)
        D  equivale a   d$  (cancella i caratteri sino alla fine della linea)
        C  equivale a   c$  (sostituisce sino alla fine della linea)
        s  equivale a   cl  (sostituisce un solo carattere)
        S  equivale a   cc  (sostituisce un'intera linea)
```

DOVE POSIZIONARE IL CONTATORE

I comandi "3dw" e "d3w" cancellano tre parole. Cercando il pelo nell'uovo, il primo comando, "3dw" cancella una parola tre volte; il comando "d3w" cancella tre parole una volta sola. In pratica, non vi è differenza. Potete addirittura utilizzare due contatori. Per esempio, "3w2d" cancella due parole, ripetendo l'azione tre volte, per un totale di sei parole.

SOSTITUIRE UN CARATTERE

Il comando "r" non è un operatore. Questo attende che digitiate un carattere, per rimpiazzare con questo il carattere sotto il cursore. Potete ottenere lo stesso risultato con "cl" o con il comando "s", ma con "r" non dovete premere <Esc>

```
        there is somerhing grong here ~
```

```
        rT          rt    rw

        There is something wrong here ~
```

Usando un contatore con "r", molti caratteri saranno rimpiazzati con lo stesso carattere. Ad esempio:

```
        There is something wrong here ~
                            5rx

        There is something xxxxx here ~
```

Per sostituire un carattere con un'interruzione di linea, usate "r<Enter>". Ciò cancella un solo carattere ed inserisce una interruzione di linea. Usando un conto qui solo relativo al numero di caratteri cancellati: "4r<Enter>" sostituisce quattro caratteri con una interruzione di linea.

==

04.3 Ripetere una modifica

Il comando "." è uno dei più semplici ma potenti comandi in Vim. Tale comando ripete l'ultima modifica. Ad esempio, supponete di star lavorando su un file HTML e di voler cancellare tutte le etichette <B>. Posizionate il cursore sul primo < e cancellate <B> con il comando "df>". Poi spostatevi su < del prossimo </B> e cancellatelo usando il comando ".". Il comando "." esegue l'ultimo comando di modifica (in questo caso, "df>"). Per cancellare un'altra etichetta, posizionate il cursore su < e usate il comando ".".

```
                              To <B>generate</B> a table of <B>contents ~
        f<   find first <     --->
        df>  delete to >         -->
        f<   find next <           --------->
        .    repeat df>                  --->
        f<   find next <                    ------------->
        .    repeat df>                                 -->
```

Il comando "." funziona su ogni comando, ad eccezione di "u" (undo), CTRL-R (redo) e dei comandi che iniziano con un due punti (:).

Un altro esempio: si vuole sostituire la parola "quattro" con "cinque". "quattro" appare diverse volte nel vostro testo. Potete fare ciò più velocemente con questa sequenza di comandi:

```
        /quattro<Enter> trova la prima stringa "quattro"
        cwcinque<Esc>   cambia la parola in "cinque"
        n               trova il prossimo "quattro"
        .               ripete il cambiamento in "cinque"
        n               trova il prossimo "quattro"
        .               ripete il cambiamento
                        etc.
```

==

04.4 Visual_mode

Per cancellare voci semplici le sostituzioni degli operatori di movimento funzionano abbastanza bene. Ma spesso non è così semplice decidere quale comando vi sposti sul testo che volete cambiare. In questo caso potete usare il Visual_mode.

Accedete al Visual_mode premendo "v". Muovete il cursore sul testo su cui volete lavorare. Mentre fate ciò, il testo viene evidenziato. Infine digitate il comando operatore.
   Per esempio, per cancellare dalla metà di una parola alla metà di un'altra parola:

```
                This is an examination sample of visual_mode ~
                               ---------->
                                 velllld

                This is an example of visual_mode ~
```

Facendo questo, non dovrete conoscere esattamente quante volte digitare "l" per portarvi sulla giusta posizione. Potete immediatamente vedere la porzione di testo che sarà cancellata quando premerete "d".

Se ad un certo punto doveste decidere di non modificare il testo evidenziato, basterà premere <Esc> e il Visual_mode sarà concluso senza nessun cambiamento.

SELEZIONE DI LINEE

Se volete lavorare su un'intera linea, usate "V" per attivare il Visual_mode. Potete vedere come l'intera linea venga evidenziata, senza che il cursore sia mosso. Muovendovi verso destra o sinistra nulla cambia. Quando vi spostate nella riga in alto o in basso, la selezione si estende alle linee interessate, nella loro interezza.
   Per esempio, selezionate tre linee con "Vjj":

```
                        +------------------------+
                        | text more text         |
                     >> | more text more text    | |
     linee selezionate >> | text text text         | | Vjj
                     >> | text more              | V
                        | more text more         |
                        +------------------------+
```

SELEZIONE DI BLOCCHI

Se volete lavorare su un blocco rettangolare di caratteri, usate CTRL-V per attivare il Visual_mode. Ciò è veramente utile quando si lavora con delle tabelle.

```
               name            Q1      Q2      Q3
               pierre          123     455     234
               john            0       90      39
               steve           392     63      334
```

Per cancellare la colonna "Q2" posta in mezzo, spostate il cursore sulla "Q" di "Q2". Premete CTRL-V per attivare Visual_mode con modalità a blocco. Ora spostate il cursore tre linee in basso con "3j" e poi sulla prossima parola con "w". Potete vedere che viene incluso anche il primo carattere dell'ultima colonna. Per escluderlo, usate "h". Ora premete "d" e la colonna in mezzo è a posto.

ANDARE DALL'ALTRO LATO

Se avete già selezionato del testo in Visual_mode, e vi accorgete di dover fare un cambiamento dall'altra parte della selezione, usate il comando "o" (Suggerimento: "o" sta per "other", in inglese "altra" parte). Il cursore si posizionerà dall'altra parte, e potrete muovere il cursore per cambiare dove la selezione inizia. Premendo "o" nuovamente, sarete portati dall'altra parte.

Quando usate la selezione con modalità a blocchi, avete quattro angoli. "o" minuscolo vi porta sull'angolo opposto, in diagonale. Per spostarvi sull'altro angolo sulla stessa linea, si usa "O" maiuscolo.

Note: "o" e "O" in Visual_mode si comportano in modo differente dal Normal_mode, nel quale aprono una nuova linea sotto o sopra il cursore.

==
04.5 Muovere il testo

Quando cancellate qualcosa con "d", "x", o un altro comando, il testo viene memorizzato. Potete incollarlo nuovamente usando il comando "p". (Vim chiama ciò "put").
   Osservate come questo funziona. Prima potete cancellare un'intera linea, posizionando il cursore sulla linea che volete cancellare e digitando "dd", Ora spostate il cursore nella posizione dove volete incollare la linea e usate il comando "p" (put). La linea verrà inserita sulla linea sotto il cursore.

```
        una linea       una linea       una linea
        linea 2    dd   linea 3     p   linea 3
        linea 3                        linea 2
```

Poiché avete cancellato un'intera linea, il comando "p" ha posizionato la linea di testo sotto il cursore. Se cancellate parte di una linea (una parola, per esempio), il comando "p" la incollerà esattamente dopo il cursore.

```
        Some more boring try text to out commands. ~
                        ---->
                         dw

        Some more boring text to out commands. ~
```

```
                        ------->
                          welp

        Some more boring text to try out commands. ~
```

ALTRO SULL'INCOLLAGGIO

Il comando "P" incolla il testo come "p", ma prima del cursore. Quando avrete cancellato un'intera linea con "dd", "P" la incollerà prima del cursore. Quando avrete cancellato una parola con "dw", "P" la incollerà proprio prima del cursore.

Potete ripetere l'incollatura quante volte volete. Verrà usato sempre lo stesso testo.

Potete usare un contatore con "p" e "P". Il testo sarà ripetuto tante volte quante ne sono state specificate con il contatore. Così, "dd" e poi "3p" incolla tre copie della stessa linea cancellata.

SCAMBIARE DUE CARATTERI

Capita spesso che mentre si digita, le dita corrano più velocemente del cervello (o viceversa...). Il risultato è una cosa simile a questa: "teh" invece di "the". Vim rende più facile correggere questo tipo di problema. Posizionate il cursore sulla "e" di "teh" ed eseguite il comando "xp". Questo lavora in questo modo: "x" cancella il carattere e lo pone in un registro. "p" incolla il testo dopo il cursore, che è dopo la "h".

```
        teh     th     the ~
         x       p
```

==

04.6 Copiare il testo

Per copiare del testo da un posto ad un altro, potete cancellarlo, usare "u" per annullare la cancellazione e poi usare "p" per incollarlo dove si vuole. C'è un metodo più semplice: usare lo "yank" (in inglese: "strappare"). L'operatore "y" copia il testo in un registro. Poi, il comando "p" può essere usato per incollarlo.
   "Strappare" è solo un sinonimo che Vim usa al posto di "copiare". Questo perché la lettera "c" era già stata usata per l'operatore di cambiamento, mentre la lettera "y" era ancora disponibile. Chiamando questo operatore "yank", è più facile ricordare l'uso del tasto "y".

Poiché "y" è un operatore, si usa "yw" per copiare una parola. Come al solito, è possibile usare un contatore. Per copiare due parole, usate "y2w". Per esempio:

```
        let sqr = LongVariable * ~
                  -------------->
                       y2w

        let sqr = LongVariable * ~
                               p

        let sqr = LongVariable * LongVariable ~
```

Notate che "yw" include lo spazio bianco dopo una parola. Se non desiderate ciò, usate "ye".

Il comando "yy" copia un'intera linea, come "dd" cancella un'intera linea. Contrariamente alle aspettative, mentre "D" cancella dal cursore fino alla fine della linea, "Y" funziona come "yy", cioè copia l'intera linea.

```
        a text line   yy       a text line           a text line
        line 2                 line 2          p     line 2
        last line              last line             a text line
                                                     last line
```

==

04.7 Usare la clipboard

Se state usando la versione GUI di Vim (gvim), potete trovare la voce "Copia" nel menu "Edit". Prima selezionate del testo con il Visual_mode, poi usate il menu Modifica/Copia. Il testo selezionato è ora stato copiato nella clipboard (NdT: clipboard=parte della memoria in cui è temporaneamente

memorizzato un testo). Potete incollare il testo in altri programmi.
Ovviamente, anche nello stesso Vim.

Se avete copiato, da un'altra applicazione, del testo nella clipboard, potete
incollarlo in Vim con il menu Modifica/Incolla. Questo funziona in
Normal_mode e in Insert_mode. In Visual_mode il testo selezionato è
sostituito con il testo incollato.

La voce "Taglia" del menù cancella il testo prima di porlo nella clipboard.
Le voci "Copia", "Taglia" e "Incolla" sono anche disponibili nel menù
a discesa.

Se il vostro Vim ha una barra degli strumenti, potete trovare queste voci
anche lì.

Se non state usando la GUI, oppure se non vi piace usare il menù, dovete
utilizzare un altro metodo. Usate i soliti comandi "y" (yank) e "p" (put), ma
premettete loro "* (virgolette asterisco). Per copiare una linea nella
clipboard: **>**

```
"*yy
```

Per trasferire del testo dalla clipboard di nuovo nel testo: **>**

```
"*p
```

Ciò funziona solo per le versioni di Vim che includono il supporto per la
clipboard. Potete trovare maggiori informazioni sulla clipboard nella sezione
|09.3**|** e qui: **|**clipboard**|**.

==
04.8 Oggetti di testo

Se il cursore è a metà di una parola che volete cancellare, dovete tornare
indietro all'inizio di questa parola prima di usare il comando "dw".
Esiste un modo più rapido: "daw".

```
this is some example text. ~
                daw

this is some text. ~
```

La "d" di "daw" è l'operatore di cancellazione. "aw" è un oggetto di testo.
Suggerimento: "aw" ricorda "A Word", cioè "Una Parola" in inglese. Per la
precisione, viene cancellato anche lo spazio bianco che segue la parola (e lo
spazio bianco prima della parola se ci si trova alla fine della linea).

L'uso degli oggetti di testo rappresenta un terzo modo per effettuare dei
cambiamenti in Vim. C'erano già operatore-movimento e Visual_mode; ora si
aggiunge alla nostra lista anche l'operatore-oggetti di testo.
   E' molto simile all'operatore-movimento, ma invece di operare sul testo fra
la posizione del cursore prima e dopo il comando di spostamento, l'oggetto di
testo è usato come un blocco. Non importa dove si trova il cursore
all'interno dell'oggetto.

Per modificare un'intera frase, usate "cis". Considerate questo testo:

```
Hello there.  This ~
is an example.  Just ~
some text. ~
```

Spostatevi all'inizio della seconda linea, su "is an". Ora usate "cis":

```
Hello there.    Just ~
some text. ~
```

Il cursore è posizionato fra gli spazi bianchi nella prima linea. Ora
digitate la nuova frase "Another line.":

```
Hello there.  Another line.  Just ~
some text. ~
```

"cis" è composto dall'operatore "c" (change) e dall'oggetto di testo "is".
Questo sta per "Inner Sentence", ovvero "Frase interna", "Frase in cui è il
cursore". Esiste anche l'oggetto "as" ("a sentence", "una frase"). La
differenza consiste nel fatto che "as" include lo spazio bianco dopo la frase,
mentre "is" non lo fa. Se volete cancellare una frase, contemporaneamente
desiderate cancellare lo spazio bianco, quindi usate "das". Se volete

digitare del nuovo testo, lo spazio bianco può rimanere, quindi usate "cis".

Potete usare oggetti di testo anche in Visual_mode. L'oggetto di testo sarà incluso nella selezione Visuale. Il Visual_mode continuerà a permanere, cosicchè potete fare questa operazione diverse volte. Per esempio, date inizio al Visual_mode con "v" e selezionate una frase con "as". Ora potete ripetere "as" per includere altre frasi. Infine, usate un operatore per fare ciò che desiderate con le frasi selezionate.

Potete trovare una lunga lista di oggetti di testo qui: |text-objects|.

==
04.9 Replace_mode

Il comando "R" fa entrare Vim in Replace_mode. In questa modalità, ogni carattere digitato sostituisce quello sotto il cursore. Questo comportamento permane fino a che non si digita <Esc>.
   In questo esempio, date inizio al Replace_mode sulla prima "t" di "testo":

        This is text. ~

                Rinteresting.<Esc>

        This is interesting. ~

Avrete notato come questo comando abbia sostituito 5 caratteri nella linea con altri dodici. Il comando "R" estende automaticamente la lunghezza della linea se questa è troppo corta per contenere i caratteri da sostituire.
  Non prosegue sulla linea successiva.

Potete passare alternativamente fra Insert_mode e Replace_mode con il tasto <Insert>.

Quando usate <BS> (backspace) per fare delle correzioni, potete notare come venga nuovamente immesso sulla riga il vecchio testo. Ciò funziona come un comando "undo" ("annulla") per l'ultimo carattere digitato.

==
04.10 Conclusioni

Gli operatori, i comandi di movimento e gli oggetti di testo vi offrono la possibilità di creare molte combinazioni. Ora che conoscete il loro funzionamento, potete usare N operatori combinati con M comandi di movimento per ottenere N * M comandi!

Potete trovare una lista di operatori qui: |operator|

Per esempio, ci sono molti altri modi per cancellare porzioni di testo. La lista seguente ne illustra alcuni fra i più usati:

```
x       cancella il carattere sotto il cursore (abbreviazione per "dl")
X       cancella il carattere prima del cursore (abbreviazione per "dh")
D       cancella dal cursore alla fine della linea (abbreviazione per "d$")
dw      cancella dal cursore all'inizio della prossima parola
db      cancella dal cursore al precedente inizio di parola.
diw     cancella la parola sotto il cursore (esclusi gli spazi bianchi)
daw     cancella la parola sotto il cursore (inclusi gli spazi bianchi)
dG      cancella fino alla fine del file
dgg     cancella fino all'inizio del file
```

Usando "c" invece di "d", i comandi precedenti diventeranno comandi di cambiamento. E così via.

Ci sono poi alcuni altri comandi usati frequentemente, che non è possibile classificare facilmente:

```
        ~       cambia da maiuscolo a minuscolo (e viceversa) il carattere
                sotto il cursore, e sposta il cursore al prossimo carattere.
                Questo non è un operatore (a meno che no sia impostato
                'tildeop'), quindi non è possibile usarlo in combinazione con
                un comando di movimento.  Funziona in Visual_mode e modifica
                il maiuscolo/minuscolo per tutto il testo selezionato.

        I       Avvia l'Insert_mode dopo aver spostato il cursore al primo
                carattere non-blank nella linea.

        A       Avvia l'Insert_mode dopo aver spostato il cursore alla fine
                della linea.
```

==

Capitolo seguente: |usr_05.txt| Configurazioni personali

Copyright: vedere |manual-copyright| vim:tw=78:ts=8:ft=help:norl:

Segnalare refusi a Bartolomeo Ravera - E-mail: barrav at libero.it
oppure ad Antonio Colombo - E-mail: azc100 at gmail.com