usr_25.txt Per Vim version 7.4. Ultima modifica: 2016 Mar 28

VIM USER MANUAL - di Bram Moolenaar
Traduzione di questo capitolo: Cristian Rigamonti

Elaborare testo formattato

Difficilmente un testo è composto da una frase per linea. Questo capitolo spiega come interrompere le frasi per adattarle alla pagina ed altre formattazioni. Vim ha anche utili funzioni per elaborare paragrafi di una sola linea e tabelle.

Capitolo seguente: |usr_26.txt| Ripetizione
Capitolo precedente: |usr_24.txt| Inserzione rapida
Indice: |usr_toc.txt|

==========

25.1 Interrompere le linee

Vim ha una serie di funzioni che facilitano il trattamento del testo. Di default, l'editor non interrompe le linee automaticamente. In altre parole, dovete premere **<Enter>** voi stessi. Ciò è utile se state scrivendo programmi e volete essere voi a decidere dove finisce ogni linea. Non va altrettanto bene se state scrivendo della documentazione e volete che il testo occupi al massimo 70 caratteri per linea.

Impostando l'opzione **'textwidth'**, Vim inserisce automaticamente le interruzioni di linea. Supponete, ad esempio, di volere una colonna molto stretta di soli 30 caratteri. Dovete eseguire il comando seguente: >

```
:set textwidth=30
```

Ora iniziate a scrivere (il righello è stato aggiunto qui per chiarezza)

```
          1         2         3
12345678901234567890123456789012345
Ho insegnato programmazione pe ~
```

Se ora scrivete la "r", la linea supererà il limite di 30 caratteri. Quando Vim se ne accorge, inserisce un'interruzione di linea e ottenete il seguente:

```
          1         2         3
12345678901234567890123456789012345
Ho insegnato programmazione ~
per un po' ~
```

Continuando, potete scrivere il resto del paragrafo:

```
          1         2         3
12345678901234567890123456789012345
Ho insegnato programmazione ~
per un po'. Una volta sono ~
stato fermato dalla polizia ~
perché davo dei compiti troppo ~
difficili. Storia vera. ~
```

Non dovete digitare il ritorno a capo: Vim lo inserisce automaticamente.

Nota:
Con l'opzione **'wrap'**, Vim interrompe le linee solo in fase di visualizzazione, non inserisce interruzioni di linea nel file.

RIFORMATTARE

Vim è un editor, non un word processor. In un word processor, se cancellate qualcosa all'inizio di un paragrafo, le interruzioni di linea vengono

rielaborate. In Vim non avviene, quindi se cancellate la parola "programmazione" dalla prima linea, vi ritrovate con una linea più corta:

```
                1         2         3
       12345678901234567890123456789012345
       Ho insegnato ~
       per un po'. Una volta sono ~
       stato fermato dalla polizia ~
       perché davo dei compiti troppo ~
       difficili. Storia vera. ~
```

Non è un bel vedere: per ridare forma al paragrafo, usate l'operatore "gq".
Usiamolo dapprima con una selezione visuale. Partendo dalla prima linea, scrivete: >

```
v4jgq
```

"v" per entrare in Visual mode, "4j" per muovervi alla fine del paragrafo e infine l'operatore "gq". Il risultato è:

```
                1         2         3
       12345678901234567890123456789012345
       Ho insegnato per un po'. Una ~
       volta sono stato fermato ~
       dalla polizia perché davo dei ~
       compiti troppo difficili. ~
       Storia vera. ~
```

Poiché "gq" è un operatore, potete usare uno dei seguenti tre modi per selezionare il testo su cui operare: con Visual mode, con un movimento e con un oggetto di testo.
Nell'esempio precedente avremmo potuto usare "gq4j". Che significa scrivere meno, ma bisogna conoscere il numero delle linee. Un comando di spostamento ancora più utile è "}". Questo va alla fine del paragrafo. Così "gq}" formatta il testo dalla posizione del cursore fino alla fine del paragrafo attuale.
Un oggetto di testo molto utile da usare con "gq" è il paragrafo. Provate: >

```
gqap
```

"ap" sta per "a paragraph". Ciò formatta il testo di un solo paragrafo (separato da linee vuote). Così la parte dopo il cursore.
Se i vostri paragrafi sono separati da linee vuote, potete formattare l'intero file scrivendo: >

```
gggqG
```

"gg" per spostarvi alla prima linea, "gqG" per formattare fino all'ultima linea.
Attenzione: se i paragrafi non sono opportunamente separati, verranno uniti. Un errore comune è quello di lasciare una linea con uno spazio o una Tabulazione **[Tab]**. Quella è una linea "bianca", ma non vuota.

Vim è in grado di formattare più che il solo testo semplice. Si veda |**fo-table**| in proposito. Si veda anche l'opzione **'joinspaces'** per cambiare il numero di spazi usati dopo un punto.
È possibile usare un programma esterno per formattare. Ciò è utile se il vostro testo non può venire correttamente formattato con in comandi disponibili in Vim. Si veda l'opzione **'formatprg'**.

==========

25.2 Allineare il testo

Per centrare un intervallo di linee, usate il comando seguente: >

```
:{range}center [width]
```

{range} è il solito intervallo da linea di comando. **[width]**, è un'opzionale larghezza di linea da usare per centrare il testo. Se **[width]** non viene specificata, assume automaticamente il valore di **'textwidth'** (se **'textwidth'** fosse 0, il valore predefinito è 80).
Per esempio: >

```
:1,5center 40
```

il risultato sarà il seguente:

```
     Ho insegnato per un po'. Una ~
       volta sono stato fermato ~
    dalla polizia perché davo dei ~
       compiti troppo difficili. ~
            Storia vera. ~
```

ALLINEAMENTO A DESTRA

In modo simile il comando ":right" allinea il testo a destra: >

```
:1,5right 37
```

produrrà:

```
         Ho insegnato per un po'. Una ~
             volta sono stato fermato ~
        dalla polizia perché davo dei ~
            compiti troppo difficili. ~
                        Storia vera. ~
```

ALLINEAMENTO A SINISTRA

Infine, c'è il comando: >

```
:{range}left [margin]
```

A differenza di ":center" e ":right", l'argomento di ":left" non è la lunghezza della linea. È invece il margine sinistro. Se viene omesso, il testo verrà allineato al margine sinistro dello schermo (lo stesso risultato si ottiene indicando un margine zero). Se vale 5, il testo sarà rientrato di 5 spazi. Provate ad esempio questi comandi: >

```
:1left 5
:2,5left
```

Il risultato è il seguente:

```
             Ho insegnato per un po'. ~
        Una volta sono stato fermato ~
        dalla polizia perché davo dei ~
        compiti troppo difficili. ~
        Storia vera. ~
```

GIUSTIFICARE IL TESTO

Vim non contiene comandi per giustificare il testo. Però c'è un bel pacchetto macro che fa proprio questo. Per usare questo pacchetto, eseguite il comando seguente: >

```
:packadd justify
```

Oppure inserite questa riga nel file |vimrc| in uso: >

```
packadd! justify
```

Questo script di Vim definisce un nuovo comando visuale: "_j". Per giustificare un blocco di testo, evidenziatelo in Visual mode ed eseguite "_j".
Maggiori spiegazioni sono contenute nel file. Per andare là, fate "gf" su questo nome: $VIMRUNTIME/pack/dist/opt/justify/plugin/justify.vim.

Un'alternativa è filtrare il testo attraverso un programma esterno. Ad esempio: >

```
:%!fmt
```

==
25.3 Rientro e tabulazione

Il rientro può venire usato per disallineare una parte del testo rispetto al resto. I testi di esempio in questo manuale, ad esempio, sono rientrati di otto spazi od un tab. Potrete normalmente ottenerlo digitando un tab all'inizio di ciascuna linea. Prendete questo testo:

```
la prima linea ~
la seconda linea ~
```

È stato creato scrivendo un tab, del testo, **<Enter>**, un tab e altro testo.
L'opzione **'autoindent'** attiva il rientro automatico: >

```
:set autoindent
```

Ogni nuova linea assume lo stesso rientro della precedente. Nell'esempio sopra, il tab dopo **<Enter>** non sarebbe stato necessario.

AUMENTARE IL RIENTRO

Per aumentare la quantità di rientro di una linea, usate l'operatore ">". Spesso questo viene usato come ">>", che aggiunge rientro alla linea corrente.

Il valore del rientro aggiunto è specificato con l'opzione **'shiftwidth'**, il cui valore predefinito è 8. Per far sì che ">>" inserisca un'ampiezza di rientro pari a quattro spazi, ad esempio, scrivete questo: >

```
:set shiftwidth=4
```

Se provate a usarlo sulla seconda linea del testo di esempio, ottenete:

```
la prima linea ~
    la seconda linea ~
```

"4>>", invece, aumenterà il rientro delle quattro linee successive.

LUNGHEZZA DELLA TABULAZIONE

Se volete ottenere rientri multiple di 4, basta impostare **'shiftwidth'** a 4; tuttavia, premendo **<Tab>** ottenete ancora un rientro di 8 caratteri. Per modificare questo comportamento, impostate l'opzione **'softtabstop'**: >

```
:set softtabstop=4
```

Ciò farà sì che il tasto **<Tab>** inserisca un rientro larga 4 caratteri. Se ci fossero già quattro spazi, verrà usato un carattere **<Tab>**, (risparmiando così 7 caratteri nel file). (Se invece volete sempre spazi e non caratteri tab, impostate l'opzione **'expandtab'**).

```
Nota:
Potreste impostare l'opzione 'tabstop' a 4. Tuttavia, se aprite il
file un'altra volta, con 'tabstop' impostato al valore predefinito di
8, il file sarà visualizzato in modo scorretto. In altri programmi, e
nella stampa, il rientro risulterà sbagliato. Per questo motivo è
raccomandabile tenere 'tabstop' sempre ad 8. Questo è il valore
standard ovunque.
```

MODIFICARE LA TABULAZIONE

Se elaborate un file che è stato scritto con una tabulazione 3, in Vim apparirà terribile, visto che Vim usa il valore standard di 8 per la tabulazione. Potreste risolvere impostando **'tabstop'** a 3, ma dovreste farlo ogni volta che lavorate su questo file.
Vim può cambiare l'uso delle tabulazioni nel vostro file. Per prima cosa impostate **'tabstop'** in modo che il rientro risulti corretto, quindi usate il comando ":retab": >

```
:set tabstop=3
:retab 8
```

Il comando ":retab" imposterà 'tabstop' a 8, modificando al contempo il testo in modo che il suo aspetto rimanga inalterato: le sequenze di spazi bianchi saranno trasformate opportunamente in sequenze di tab e spazi per questo. Potete ora salvare il file. La prossima volta che lo aprirete, il rientro risulterà corretto senza dover impostare alcuna opzione.

Attenzione: usando ":retab" su un programma, potreste modificare gli spazi in una costante di stringa. Per questo è buona norma usare "\t" invece del vero tab.

==

25.4 Trattare le linee lunghe

Qualche volta aprirete un file più largo del numero di colonne della finestra. Quando ciò avviene, Vim spezza le linee cosicché tutto stia sullo schermo.

Se disattivate l'opzione 'wrap', ogni linea del file verrà mostrata su una linea dello schermo. Allora la fine delle linee lunghe scomparirà sulla destra dello schermo.

Quando spostate il cursore su un carattere che non può essere visto, Vim farà scorrere il testo fino a mostrarlo. Ciò è come spostare una finestra sul testo in senso orizzontale.

Di default Vim non mostra una barra di scorrimento nella GUI. Se volete abilitarne una, usate il comando seguente: >

```
:set guioptions+=b
```

Una barra di scorrimento orizzontale apparirà in basso nella finestra di Vim.

Se non avete, o non volete usare, una barra di scorrimento, usate i seguenti comandi per far scorrere il testo. Il cursore resterà nello stesso posto, ma verrà spostato sul testo visibile se necessario.

zh	scorre (il testo) a destra
4zh	scorre a destra di quattro caratteri
zH	scorre a destra di metà finestra
ze	scorre a destra finché il cursore è a fine riga
zl	scorre (il testo) a sinistra
4zl	scorre a sinistra di quattro caratteri
zL	scorre a sinistra di metà finestra
zs	scorre a sinistra finché il cursore è a fine riga

Proviamo a mostrarlo con una linea di testo. Il cursore è sulla "e" di "del". La "finestra attuale" sopra la linea indica la parte di testo attualmente visibile. Le "finestre" sotto il testo indicano la parte di testo visibile dopo avere eseguito il comando scritto sulla sinistra.

```
                          |<--finestra attuale-->|
          un testo lungo, parte del quale è visibile nella finestra ~
ze         |<--    finestra    -->|
zH          |<--    finestra    -->|
4zh               |<--    finestra    -->|
zh                   |<--    finestra    -->|
zl                     |<--    finestra    -->|
4zl                       |<--    finestra    -->|
zL                              |<--    finestra    -->|
zs                            |<--    finestra    -->|
```

SPOSTARSI CON L'INTERRUZIONE DI LINEA DISATTIVATA

Quando 'wrap' è disattivata ed il testo è stato fatto scorrere orizzontalmente, potete usare i seguenti comandi per spostare il cursore su uno dei caratteri visibili, ignorando il testo al di fuori della finestra, a destra e a sinistra. Questi comandi non fanno mai scorrere il testo:

g0	al primo carattere visibile della linea
g^	al primo carattere "non bianco" visibile della linea
gm	a metà della linea
g$	all'ultimo carattere visibile della linea

```
        |<--    finestra   -->|
un testo     lungo, parte del quale è visibile ~
         g0  g^    gm        g$
```

INTERROMPERE LE LINEE SENZA SPEZZARE LE PAROLE *edit-no-break*

Quando preparate un testo che dovrà essere usato da un altro programma, potreste dover fare dei paragrafi senza linea di interruzione. Uno svantaggio di usare 'nowrap' è che non potreste vedere l'intera frase su cui state lavorando; d'altra parte, quando 'wrap' è attivo, le parole vengono spezzate a metà, rendendone difficile la lettura.
Una buona soluzione per elaborare questo tipo di paragrafi consiste nell'impostare l'opzione 'linebreak'. Vim allora interromperà le linee nel punto giusto mostrando la linea. Il testo nel file rimarrà inalterato.
Senza 'linebreak' un testo potrebbe apparire così:

```
+---------------------------------+
|programma per creare automaticame|
|nte delle lettere. Volevano spedi|
|re una lettera personalizzata ai |
|loro 1000 clienti più ricchi. Sfo|
|rtunatamente per il programmatore|
+---------------------------------+
```

Dopo: >

```
:set linebreak
```

apparirebbe così:

```
+---------------------------------+
|programma per creare             |
|automaticamente delle lettere.   |
|Volevano spedire una lettera     |
|personalizzata ai loro 1000      |
|clienti più ricchi.              |
+---------------------------------+
```

Opzioni collegate:
'breakat' specifica i caratteri ove un'interruzione può venire inserita.
'showbreak' specifica una stringa da mostrare all'inizio di una linea interrotta.
Impostate 'textwidth' a zero per evitare che i paragrafi vengano interrotti.

SPOSTARSI SULLE LINEE VISIBILI

I comandi "j" e "k" spostano il cursore alla prossima od alla precedente linea del file. Quando usati su una linea lunga ciò significa spostare molte linee dello schermo alla volta.
Per spostare solo una linea dello schermo, usate i comandi "gj" e "gk". Quando una linea non viene interrotta essi fanno come "j" e "k". Quando la linea viene interrotta, spostano il carattere indicato una linea sopra o sotto.
Potrebbe piacervi usare le seguenti mappature, che assegnano questi comandi di movimento ai tasti cursore: >

```
:map <Up> gk
:map <Down> gj
```

TRASFORMARE UN PARAGRAFO IN UNA LINEA *edit-paragraph-join*

Se volete importare del testo in un programma come MS-Word, ogni paragrafo deve essere formato da una sola linea. Se i vostri paragrafi sono attualmente separati da linee vuote, ecco come trasformare ogni paragrafo in una linea singola: >

```
:g/./,/^$/join
```

Sembra complicato. Dividiamolo in parti:

```
:g/./          Un comando ":global" che trova tutte le linee che
               contengono almeno un carattere.
    ,/^$/      Un intervallo che inizia dalla linea attuale (la linea
               non vuota) e termina con una linea vuota.
```

join Il comando ":join" unisce l'intervallo di linee formandone una sola.

Partendo da questo testo, che contiene otto linee interrotte alla colonna 30:

```
+---------------------------------+
|Un programma per creare          |
|automaticamente delle lettere.   |
|Volevano spedire una lettera     |
|personalizzata.                  |
|                                 |
|Ai loro 1000 clienti più         |
|ricchi. Sfortunatamente per il   |
|programmatore,                   |
+---------------------------------+
```

Otterreste queste due linee:

```
+---------------------------------+
|Un programma per creare automatica|
|mente delle lettere. Volevano sped|
|ire una lettera personalizzata.  |
|Ai loro 1000 clienti più ricchi. S|
|fortunatamente per il programmator|
+---------------------------------+
```

Nota: il tutto non funziona se la linea che separa i paragrafi è "bianca" ma non vuota, ossia se contiene spazi e/o tab. Questo comando funziona con le linee "bianche": >

```
:g/\S/,/^\s*$/join
```

Ciò richiede almeno una linea "bianca" o vuota alla fine del file perché venga unito anche l'ultimo paragrafo.

==========

25.5 Elaborare tabelle

Supponete di lavorare su una tabella con quattro colonne:

```
tabella          test 1     test 2     test 3 ~
input A          0.534 ~
input B          0.913 ~
```

Dovete inserire dei numeri nella terza colonna. Potreste spostarvi sulla seconda linea, usare "A", inserire un sacco di spazi e scrivere il testo.
Per questo tipo di elaborazione esiste un'opzione speciale: >

```
:set virtualedit=all
```

Ora potete muovere il cursore in posizioni dove non vi sia alcun testo. Questo si chiama "spazio virtuale". Elaborare tabelle risulta molto più facile in questo modo.
Muovete il cursore ricercando l'intestazione dell'ultima colonna: >

```
/test 3
```

Ora premete "j" e vi ritrovate esattamente dove dovete inserire il valore per "input A". Scrivendo "0.693" ottenete:

```
tabella          test 1     test 2     test 3 ~
input A          0.534                 0.693 ~
input B          0.913 ~
```

Vim ha riempito automaticamente lo spazio che precede il nuovo testo. Ora, per riempire il campo successivo in questa colonna usate "Bj". "B" vi sposta all'indietro, all'inizio di una parola separata da spazio bianco. Poi "j" vi sposta nella posizione dove può essere inserito il valore per il prossimo campo.

Nota:
Potete spostare il cursore ovunque sullo schermo, anche oltre la fine di una linea, ma Vim non inserirà spazi là, finché non inserirete un

carattere in quella posizione.

COPIARE UNA COLONNA

Volete aggiungere una colonna, che deve essere la copia della terza colonna e posta prima della colonna "test 1". Potete farlo in sette passi:
1. Spostate il cursore sull'angolo superiore sinistro di questa colonna, ad es. con "/test 3".
2. Premete **CTRL-V** per entrare in Visual mode a blocchi.
3. Spostate il cursore di due linee verso il basso con "2j". Siete ora in "spazio virtuale": la linea "input B" della colonna "test 3".
4. Spostate il cursore a destra, per includere l'intera colonna nella selezione, più lo spazio che volete tra le colonne. "9l" dovrebbe farlo.
5. Copiate il rettangolo selezionato con "y".
6. Spostate il cursore su "test 1", dove va inserita la nuova colonna.
7. Premete "P".

Il risultato dovrebbe essere:

```
tabella          test 3     test 1     test 2     test 3 ~
input A          0.693      0.534                 0.693 ~
input B                     0.913 ~
```

Notate che l'intera colonna "test 1" è stata spostata a destra, compresa la linea in cui la colonna "test 3" non contiene testo.

Tornate ai movimenti del cursore non virtuali con: >

```
:set virtualedit=
```

MODALITÀ DI SOSTITUZIONE VIRTUALE

Lo svantaggio di usare **'virtualedit'** è che viene avvertito diverso. Non potete riconoscere tab o spazi oltre la fine delle linee quando spostate il cursore. Può essere usato un altro metodo: il Virtual Replace mode.
Supponete di avere una linea in una tabella che contenga sia tab che altri caratteri. Usate "rx" sul primo tab:

```
inp     0.693   0.534   0.693 ~

          |
     rx   |
          V

inpx0.693   0.534       0.693 ~
```

L'allineamento viene perduto. Per evitare ciò, usate il comando "gr":

```
inp     0.693   0.534   0.693 ~

          |
    grx   |
          V

inpx    0.693   0.534   0.693 ~
```

Ciò che avviene è che il comando "gr" si assicura che il nuovo carattere prenda la giusta quantità di spazio dello schermo. Vengono inseriti spazi o tab in più per riempire lo spazio vuoto. Così, ciò che accade ora è che un tab viene sostituito da "x" e che vengono inseriti spazi bianchi per fare sì che il testo dopo di esso mantenga la sua posizione. In questo caso viene inserito un tab.
Se dovete sostituire più di un carattere e usate il comando "R" per entrare in modalità Sostituzione (si veda |04.9|), rovinerete l'allineamento e sostituirete i caratteri sbagliati:

```
inp     0       0.534   0.693 ~

          |
 R0.786   |
          V
```

```
        inp     0.78634 0.693 ~
```

Il comando "gR" invece usa il Virtual Replace mode. Ciò preserva
l'allineamento:

```
        inp     0       0.534   0.693 ~

                |
        gR0.786 |
                V

        inp     0.786   0.534   0.693 ~
```

==

Capitolo seguente: |usr_26.txt| Ripetizione

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