usr_03.txt   Per Vim version 7.0. Ultima modifica: 2006 Giu 21

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

Muoversi nel file

Prima di inserire o cancellare del testo, il cursore deve essere portato sul punto giusto. Vim ha un grande numero di comandi per posizionare il cursore.
   Questo capitolo vi mostrerà come usare quelli principali. Potete trovare una lista di questi comandi in |Q_lr|.

  Capitolo seguente: |usr_04.txt| Fare piccole modifiche
Capitolo precedente: |usr_02.txt| I primi passi con Vim
              Indice: |usr_toc.txt|

==
03.1 Movimenti di parola

Per muovere il cursore avanti di una parola, usate il comando "w". Come molti comandi di Vim, potete usare un prefisso numerico per muovervi di più parole.
   Per esempio, "3w" muove avanti di tre parole. Questa figura mostra come questo accada:

```
        Questa è una linea di esempio~
          ----->->------------>
             w   w     3w
```

Notate che se il cursore è già all'inizio di una parola, "w" muove all'inizio della parola seguente.
   Il comando "b" muove indietro all'inizio della parola precedente:

```
        Questa è una linea di esempio ~
        <------<-<---------<--<---
            b   b     2b     b  b
```

C'è anche il comando "e" che muove alla fine della parola seguente, e "ge" che muove alla fine della parola precedente:

```
        Questa è una linea di esempio ~
             <-    <---- --->   ---->
             ge      ge   e      e
```

Se siete posizionati sull'ultima parola di una linea, il comando "w" vi porterà alla prima parola della riga seguente. Potete usare "w" per spostarvi all'interno di un paragrafo molto più rapidamente che usando "l". "b" si comporta allo stesso modo, ma in direzione opposta.

Una parola termina al primo carattere che non ricorre in una parola, ad esempio ".", "-" oppure ")". Per modificare la lista dei caratteri che fanno parte di una parola, si veda l'opzione 'iskeyword'.
   E' anche possibile muoversi fra PAROLE (WORDS) separate da spazi bianchi. Queste non sono parole nel senso di cui prima. In questo caso si usano le lettere maiuscole. Guardate il seguente esempio:

```
             ge       b        w                                  e
           <---      <-       --->                             ----->
    Questa è-una linea, con parole/separate/stranamente (e altro ancora). ~
         <------ <------       ------------------------->        ------>
            gE       B                   W                        E
```

Con questi vari comandi minuscoli e maiuscoli, potete muovervi rapidamente avanti e indietro all'interno di un paragrafo.

==
03.2 Spostarsi all'inizio o alla fine di una riga

Il comando "$" sposta il cursore alla fine della linea. Se la vostra tastiera ha un tasto <Fine>, premendolo otterrete lo stesso effetto.

Il comando "^" muove il cursore al primo carattere non-blank della linea. Il comando "0" (zero) muove esattamente al primo carattere della linea. Il tasto <Home> fa la stessa cosa. In una immagine:

```
                ^
           <---------
     .....Questa è una linea di esempio~
     <--------------      ------------>
            0                $
```

(i puntini "....." in questo esempio indicano degli spazi)

   Il comando "$" può essere preceduto da un numero, come molti comandi di movimento. Poiché muoversi più volte alla fine della linea non ha senso, l'editor muoverà il cursore alla fine di un'altra linea. Per esempio, "1$" vi sposterà alla fine della prima linea, (quella dove siete), "2$" alla fine della linea seguente, e così via.
   Il comando "0" non può essere preceduto da alcun numero, perché lo "0" farebbe parte del numero. Contrariamente alle aspettative, l'uso di un numero davanti a "^" non ha alcun effetto.

==
03.3 Spostarsi verso un carattere

Uno dei più utili comandi di movimento è il comando di ricerca di un singolo carattere. Il comando "fx" cerca avanti nella linea la prima occorrenza del carattere x. Suggerimento: "f" sta per "Find", ovvero "Cerca" in inglese.
   Supponiamo che nel seguente esempio vogliate andare verso la u della parola umano. Eseguendo il comando "fu" il cursore sarà posizionato sopra la u:

```
        Errare è umano. Per fare un vero disastro ci vuole un computer. ~
        --------->-------------------------------------------------->
           fu                           fp
```

Nello stesso esempio, il comando "fp" vi sposta al centro della parola computer.
   Potete specificare un numero; in questo modo, potete ad esempio andare alla "o" di "disastro" con "3fo":

```
        Errare è umano. Per fare un vero disastro ci vuole un computer. ~
                  ------------------------------>
                              3fo
```

Il comando "F" cerca verso sinistra:

```
        Errare è umano. Per fare un vero disastro ci vuole un computer. ~
                  <----------------------
                            Fm
```

Il comando "tx" lavora allo stesso modo di "fx", ad eccezione del fatto che si ferma un carattere prima del carattere ricercato. Suggerimento: "t" sta per "To", ovvero "Verso" in inglese. Per andare a ritroso usare "Tx".

```
        Errare è umano. Per fare un vero disastro ci vuole un computer. ~
                   <-----------   ---------------------->
                        Tm                 tl
```

Questi quattro comandi possono essere ripetuti con ";". "," ripete nella direzione opposta. Il cursore non è mai spostato su un'altra linea. Neppure quando la frase continua.

Può capitare di iniziare una ricerca, e di accorgersi di aver usato il comando sbagliato. Per esempio, digitate "f" per cercare in avanti, mentre volevate usare "F". Per annullare una ricerca, premete <Esc>. Così "f<Esc>" terminerà la ricerca e non farà nient'altro. Note: <Esc> annulla molte operazioni, non solo le ricerche.

==
03.4 Spostarsi sulla parentesi corrispondente

Quando si scrive un programma, spesso si utilizzano dei costrutti con parentesi () annidate. Il comando "%" può allora tornare utile: sposta il cursore sulla parentesi corrispondente. Se il cursore è su una "(", si porterà sulla corrispondente ")". Se è su una ")", si porterà sulla

corrispondente "(".

```
                        %
                     <----->
               if (a == (b * c) / d) ~
                  <---------------->
                        %
```

Questo funziona anche con le parentesi [] e {}. (Questa lista si può modificare con l'opzione 'matchpairs').

Quando il cursore non è su un carattere adatto, "%" cercherà in avanti per trovarne uno. Così se il cursore si trova all'inizio della linea del precedente esempio, "%" cercherà avanti e troverà il primo "(". Poi muoverà il cursore alla parentesi corrispondente:

```
               if (a == (b * c) / d) ~
               ---+---------------->
                  %
```

==
03.5 Spostarsi sulla linea desiderata

Se siete un programmatore C o C++, avrete già visto un messaggio di errore simile al seguente:

```
        prog.c:33: j undeclared (first use in this function) ~
```

Questo vi dice che dovete correggere qualcosa alla riga 33. Come trovare la linea 33? Un metodo è quello di digitare "9999k" per andare all'inizio del file, e poi "32j" per muovervi verso il basso di 32 linee. Non è il massimo, ma funziona. E' comunque meglio usare il comando "G".
   Associato a un numero, questo comando vi posiziona sulla linea specificata dal numero. Per esempio, "33G" vi sposta sulla linea numero 33. (Per un metodo migliore per scandire la lista degli errori di compilazione, si veda |usr_30.txt|, che contiene informazioni sul comando :make).
   Se usato senza argomenti, il comando "G" vi posiziona alla fine del file. Un metodo veloce per andare all'inizio del file è usare "gg". "1G" si comporta allo stesso modo, ma bisogna premere qualche tasto in più.

```
            |   prima linea di un file  ^
            |   testo testo testo testo |
            |   testo testo testo testo | gg
        7G  |   testo testo testo testo |
            |   testo testo testo testo
            |   testo testo testo testo
            V   testo testo testo testo |
                testo testo testo testo | G
                testo testo testo testo |
                ultima linea di un file V
```

Un altro modo per muoversi verso una linea è usare il comando % con un numero.
   Per esempio "50%" muove alla metà (50 %) del file. "90%" muove verso la fine del file.

I comandi precedenti presumono che vogliate muovervi verso una linea del file, e non importa se la linea è presente sulla schermata o no. Cosa fare se volete muovervi su una delle linee che vedete? Questa figura vi mostra tre comandi che potete utilizzare:

```
                    +--------------------------+
               H -->| testo esempio testo      |
                    | esempio testo            |
                    | testo esempio testo      |
                    | esempio testo            |
               M -->| testo esempio testo      |
                    | esempio testo            |
                    | testo esempio testo      |
                    | esempio testo            |
               L -->| testo esempio testo      |
                    +--------------------------+
```

Suggerimento: "H" sta per "Home" (in questo caso "in Alto"), "M" per "Middle" ("in Mezzo") e "L" per "Last" (in questo caso "in Basso").

==
03.6 Sapere dove siete

Per vedere dove siete in un file, ci sono tre metodi:

1. Usare il comando CTRL-G. Ottenete un messaggio come questo (assumendo che l'opzione 'ruler' non sia attiva):

        "usr_03.txt" line 233 of 650 --35%-- col 45-52 ~

   Questo messaggio mostra il nome del file che state editando, il numero di linea dove si trova il cursore, il numero totale delle linee, in che posizione percentuale siete rispetto a tutto il file e la colonna su cui è posizionato il cursore.
   A volte potreste vedere un doppio numero di colonna. Per esempio, "col 2-9". Questo indica che il cursore è posizionato sul secondo carattere, ma poiché il primo carattere è una tabulazione, che occupa otto spazi, la colonna sullo schermo è la 9.

2. Attivare l'opzione 'number'. Sarà visualizzato il numero della riga davanti a ogni riga: **>**

        :set number
<
   Per disattivare questa opzione: **>**

        :set nonumber
<
   Poiché 'number' è una opzione binaria, premettendo un "no" all'opzione si ottiene la sua disattivazione. Una opzione binaria ha solo due valori: on e off.
   Vim ha molte opzioni. Oltre a quelle binarie, ci sono opzioni con valore numerico e stringhe. Si vedranno altri esempi di opzione dove tornerà utile usarle.

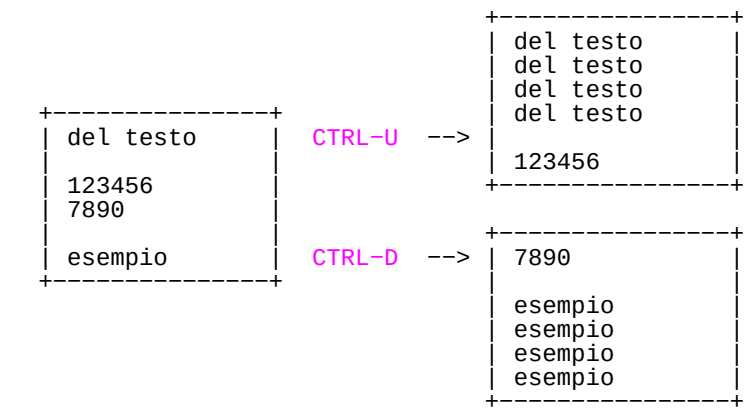
3. Attivare l'opzione 'ruler'. Sarà visualizzata la posizione del cursore nell'angolo in basso a destra della finestra di Vim: **>**

        :set ruler

L'uso dell'opzione 'ruler' ha il vantaggio di non occupare molto spazio, cosicché c'è più spazio per il vostro testo.

==
03.7 Paginazione

Il comando CTRL-U fa "scendere" una mezza schermata di testo. Pensate di guardare il testo attraverso una finestra, e di spostare questa finestra verso l'alto, per una altezza pari a metà della altezza della finestra. In questo modo la finestra di sposta all'insù, verso il testo che si trova più indietro nel file. Non preoccupatevi se fate fatica a ricordare quale parte è più in alto. Succede così alla maggior parte degli utilizzatori.
   Il comando CTRL-D sposta la finestra di visualizzazione verso il basso, e quindi sposta il vostro testo verso l'alto di una mezza schermata.

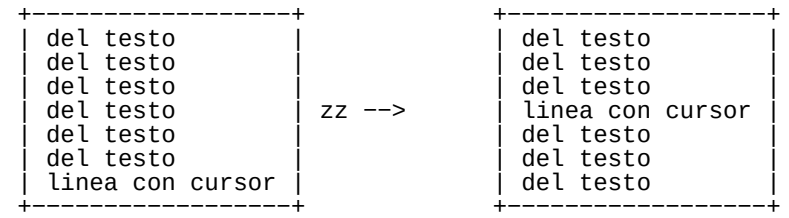


Per scendere di una linea per volta usate CTRL-E (pagina all'insù) e CTRL-Y (pagina all'ingiù). Pensate a CTRL-E come il modo per vedere una linea Extra. (Se utilizzate una mappatura dei tasti simile a MS-Windows, CTRL-Y serve per rifare una modifica (REDO), invece che per paginare).

Per andare in avanti di una intera schermata (meno un paio di linee), usate CTRL-F. Nella direzione opposta, il comando da usare è CTRL-B. Fortunatamente [...per gli inglesi] CTRL-F va "Forward" (in avanti), e CTRL-B

va "Backward" (all'indietro).

Una situazione comune è che dopo esservi mossi all'ingiù di parecchie linee con "j", il vostro cursore è in fondo allo schermo. Se volete vedere nel suo contesto la linea dove si trova il cursore, basta usare il comando "zz".



Il comando "zt" porta la linea su cui si trova il cursore in cima ("top") allo schermo, "zb" la posta a fondo schermata ("bottom"). Ci sono alcuni altri comandi per paginare, si veda |Q_sc|. Per lasciare sempre alcune linee di contesto visibili attorno alla linea del cursore, usate l'opzione 'scrolloff'.

==
03.8 Ricerche semplici

Per cercare una stringa, si usa il comando "/stringa". Per trovare la parola include, per esempio, usate il comando: >

```
/include
```

Potete notare che quando digitate "/" il cursore salta all'ultima linea della finestra di Vim, come quando usate il comando due punti. Su questa riga digiterete la parola da ricercare. Potete premere il tasto backspace ("freccia all'indietro" o <BS>) per fare delle correzioni. Usate i tasti cursore <Left> ("freccia sinistra") e <Right> ("freccia destra") se necessario.
   Il comando viene eseguito quando premete <Invio>.

```
Note:
I caratteri .*[]^%/\?~$ hanno un significato speciale.  Se volete
usarli come caratteri in una ricerca, dovete premettere una \
davanti ad essi.  Vedere più sotto.
```

Per trovare l'occorrenza successiva della stessa stringa si usa il comando "n". Per trovare il successivo #include dopo il cursore usare: >

```
/#include
```

E poi digitare "n" diverse volte. Raggiungerete ogni #include nel testo. Potete anche usare un numero se sapete verso quale occorrenza spostarvi. Così "3n" cerca la terza occorrenza. L'uso di un contatore con "/" non funziona.

Il comando "?" ha la stessa funzione di "/", ma ricerca all'indietro:

```
?parola
```

Il comando "N" ripete l'ultima ricerca nell'opposta direzione. Così "N" dopo un comando "/" cerca all'indietro, "N" dopo "?" cerca in avanti.

IGNORARE IL MINUSCOLO/MAIUSCOLO

Normalmente dovete digitare esattamente quello che volete cercare. Se non vi interessa distinguere le maiuscole dalle minuscole in una parola, impostate l'opzione 'ignorecase': >

```
:set ignorecase
```

Se ora cercate "parola", troverete sia "Parola" che "PAROLA". Per tornare all'impostazione precedente: >

```
:set noignorecase
```

CRONOLOGIA

Supponiamo che abbiate fatto queste tre ricerche: >

```
/uno
/due
/tre
```

Ora iniziate una ricerca con un semplice "/", senza premere <Invio>. Se premete <Up> (tasto "freccia in sù"), Vim mette "/tre" sulla linea di comando. A questo punto, premendo <Invio> cerca tre. Se non premete <Invio>, ma nuovamente <Up>, Vim cambia il prompt in "/due". Se premete ancora <Up> ottenete "/uno".
Potete ovviamente anche usare il tasto cursore <Down> ("freccia in giù") per muovervi attraverso la cronologia dei comandi di ricerca nell'altra direzione.

Se sapete come inizia una precedente espressione di ricerca che avete usato, e volete utilizzarlo nuovamente, potete inserire tale lettera prima di premere <Up>. Nel precedente esempio, potete digitare "/u<Up>" e Vim metterà "/uno" sulla linea di comando.

I comandi che iniziano con ":" hanno anch'essi una cronologia. Ciò vi permette di richiamare un precedente comando e di eseguirlo nuovamente. Queste due cronologie sono separate.

RICERCA DI UNA PAROLA NEL TESTO

Supponiamo di vedere la parola "FunzioneConNomeLungo" nel testo e di voler cercare la prossima occorrenza di tale parola. Potete digitare "/FunzioneConNomeLungo", ma dovete scrivere molto.
C'è un metodo più semplice: posizionare il cursore sulla parola e usare il comando "*". Vim cattura la parola sotto il cursore e la usa come stringa di ricerca.
Il comando "#" fa la stessa cosa nell'altra direzione. Potete premettere un numero: "3*" cerca la terza occorrenza della parola sotto il cursore.

RICERCA DI PAROLE INTERE

Se digitate "/sono" trovate anche "sonoro". Per trovare solamente le parole che finiscono con "sono" digitate: **>**

```
/sono\>
```

La voce "\>" è una speciale marcatura che indica la fine di una parola. Similmente "\<" indica l'inizio di una parola. Così per cercare esattamente la parola "sono" si usa: **>**

```
/\<sono\>
```

Questo non trova "sonoro" o "consono". Notate che i comandi "*" e "#" usano questi marcatori di fine-parola e inizio-parola per cercare unicamente le parole complete (si possono usare "g*" e "g#" per trovare parole parziali).

EVIDENZIARE I RISULTATI DELLE RICERCHE

Immaginiamo di editare un programma e di vedere una variabile chiamata "nr", e di voler controllare dove è usata. Potete posizionare il cursore su "nr" e usare il comando "*" e poi premere "n" per visualizzare tutte le corrispondenze.
C'è un altro metodo. Digitate questo comando: **>**

```
:set hlsearch
```

Se ora ricercate "nr", Vim evidenzierà tutte le corrispondenze. Questo è un ottimo metodo per vedere dove è usata una variabile, senza dover digitare altri comandi.
Per annullare questa impostazione: **>**

```
:set nohlsearch
```

Ora dovrete riattivare l'impostazione se volete usarla per il prossimo comando di ricerca. Se volete solo rimuovere l'evidenziazione, usate questo comando: **>**

```
:nohlsearch
```

Questo non disattiva l'opzione, ma disabilita l'evidenziazione. Subito dopo che avrete eseguito la ricerca, l'evidenziazione sarà usata di nuovo. Questo

vale anche per i comandi "n" e "N".

AFFINAMENTO DELLE RICERCHE

Ci sono alcune opzioni che modificano il comportamento delle ricerche. Queste sono quelle essenziali:
>

```
:set incsearch
```

Questo fa sì che Vim visualizzi i risultati della ricerca mentre state ancora digitando. Usate questa opzione per controllare se verrà trovata la corrispondenza che cercate. Poi premete <Invio> per spostarvi realmente nel posto evidenziato. Oppure digitate altre lettere per modificare la stringa di ricerca.
>

```
:set nowrapscan
```

Questa opzione interrompe la ricerca alla fine del file. Oppure, se state cercando all'indietro, la interrompe all'inizio del file. L'opzione 'wrapscan' è attivata per default, e quindi le ricerche proseguono ad anello, passando dalla fine all'inizio del file (o viceversa).

INTERMEZZO

Se gradite una delle opzioni appena menzionate, e volete attivarla ogni volta che usate Vim, potete inserire il comando nel file di configurazione di Vim.
Editate il file, come menzionato in |not-compatible|. Oppure usate questo comando per trovare dove sia tale file: >

```
:scriptnames
```

Editate il file, per esempio con: >

```
:edit ~/.vimrc
```

Poi aggiungete una linea con il comando per impostare l'opzione, esattamente come avreste fatto in Vim. Esempio: >

```
Go:set hlsearch<Esc>
```

"G" vi posiziona alla fine del file. "o" inizia una nuova riga, dove digitate il comando ":set". Infine uscite dalla modalità di inserimento con <Esc>. Ora salvate il file: >

```
ZZ
```

Se avviate nuovamente Vim, l'opzione 'hlsearch' sarà impostata.

==
03.9 Modelli semplici di ricerca

L'editor Vim usa delle espressioni regolari per specificare cosa si vuole cercare. Le espressioni regolari sono un mezzo estremamente compatto e potente per specificare una espressione da cercare. Sfortunatamente, questa potenza ha un prezzo, perché le espressioni regolari devono essere specificate con molta pignoleria...
In questa sezione menzioneremo solo le più essenziali. Potete trovare maggiori informazioni sulle espressioni e sui comandi di ricerca nel capitolo 27 |usr_27.txt|. Potete trovare una spiegazione esauriente qui: |pattern|.

INIZIO E FINE DI LINEA

Il carattere ^ indica l'inizio di una linea. Ad esempio, l'espressione "include" trova la parola include ovunque sulla linea.
L'espressione "^include" invece trova la parola include solo se questa è all'inizio di una linea.
Il carattere $ indica la fine di una linea. Così, "was$" trova la parola was solo se questa si trova alla fine di una linea.

In questa linea di esempio, sono indicate con delle "x" le posizioni dove è stata trovata la stringa "the":

```
the solder holding one of the chips melted and the ~
xxx                     xxx                 xxx
```

Usando "/the$" si trova soltanto:

```
        the solder holding one of the chips melted and the ~
                                                        xxx
```

E con "/^the" si trova soltanto:

```
        the solder holding one of the chips melted and the ~
        xxx
```

Se si prova a cercare con "/^the$", si troveranno solo le linee che consistono unicamente della parola "the". Gli spazi bianchi in questo caso hanno importanza, quindi se una linea contiene uno spazio dopo la parola, come "the ", l'espressione non sarà trovato.

TROVARE OGNI SINGOLO CARATTERE

Al carattere . (punto) corrisponde ogni possibile carattere. Per esempio, l'espressione "c.m" trova una stringa in cui il primo carattere è una c, il secondo carattere è un carattere qualunque, e il terzo carattere è una m. Esempio:

```
        We use a computer that became the cummin winter. ~
                 xxx             xxx      xxx
```

TROVARE CARATTERI SPECIALI

Se volete davvero trovare il carattere "." (punto), dovete "avvertire" Vim, mettendo un backslash (\) prima del punto stesso.
   Se cercate "ter.", troverete questi risultati:

```
        We use a computer that became the cummin winter. ~
                      xxxx                          xxxx
```

Cercando "ter\." si trova invece solo il secondo risultato.

==
03.10 Marcare il testo

Quando saltate in una posizione con il comando "G", Vim ricorda la posizione occupata prima di questo salto. Questa posizione è chiamata marcatore. Per tornare dove eravate partiti, usate questo comando: **>**

```
        ``
```

Il carattere ` è un backtick, cioè una virgoletta singola (un accento grave).
   (Nota del traduttore: sulle tastiere italiane si ottiene con "<AltGr>'" nei sistemi Linux, e con "<Alt>96" nei sistemi Windows).
   Se usate lo stesso comando una seconda volta, tornerete dove eravate. Questo perché il comando ` è un salto a se stesso, e la posizione prima del salto viene memorizzata.

Generalmente, ogni comando che muove il cursore in una linea che non sia la stessa linea di partenza, è considerato un salto. Questo include i comandi di ricerca "/" e "n" (non importa quanto distante sia la corrispondenza), ma non le ricerche effettuate con "fx" e "tx" o i movimenti di parola "w" e "e".
   Ugualmente, "j" e "k" non sono considerati un salto. Neppure quando si usa un contatore per muovere il cursore in una posizione molto lontana.

Il comando `` salta avanti e indietro, fra due punti. Il comando CTRL-O salta verso la precedente posizione (Suggerimento: O sta per "Older", ossia "più vecchio" in inglese). CTRL-I salta alla posizione più recente (Suggerimento: I è immediatamente vicino a O sulla tastiera). Considerate questa sequenza di comandi: **>**

```
        33G
        /^Qui
        CTRL-O
```

Prima saltate alla linea 33, poi cercate una linea che inizia con "Qui". Poi CTRL-O vi porta indietro alla linea 33. Un altro CTRL-O vi riporta dove avete iniziato. Se ora usate CTRL-I, tornerete nuovamente alla linea 33. E usando un altro CTRL-I salterete alla parola "Qui" precedentemente trovata.

```
        |   esempio testo     ^           |
    33G |   esempio testo     | CTRL-O    | CTRL-I
        |   esempio testo     |           |
        V   linea 33 testo    ^           V
        |   esempio testo     |           |
  /^Qui |   esempio testo     | CTRL-O    | CTRL-I
        V   Qui voi siete     |           V
            esempio testo
```

Note:

CTRL-I funziona allo stesso modo di <Tab>.

Il comando ":jumps" fornisce una lista delle posizioni verso le quali siete saltati. La posizione che avete usato per ultima è segnata con ">".

MARCATORI CON NOME *bookmark*

Vim vi permette di posizionare i vostri personali marcatori nel testo. Il comando "ma" marca la posizione sotto il cursore come il marcatore a. Potete posizionare 26 marcatori (usando le lettere dalla a alla z) nel vostro testo. Non potete vederli, sono solo posizioni che Vim memorizza.
Per andare su un marcatore, usate il comando `{marcatore}, dove con {marcatore} si intende la lettera prescelta. Così per muoversi sul marcatore a si usa:
>

```
`a
```

Il comando 'marcatore (virgoletta semplice, o apostrofo) vi posiziona invece all'inizio della linea che contiene il marcatore. Questa è la differenza fondamentale rispetto al comando `marcatore, il quale muove sulla colonna marcata.

I marcatori possono essere veramente utili quando si lavora su due parti collegate di un file. Supponete di avere del testo che dovete avere sott'occhio vicino all'inizio del file, mentre state lavorando su del testo vicino alla fine del file.
Muovetevi all'inizio del testo e posizionate qui il marcatore i (inizio): **>**

```
mi
```

Poi muovetevi sul testo dove volete lavorare e posizionate qui il marcatore f (fine): **>**

```
mf
```

Ora potete muovervi avanti e indietro, e quando volete vedere l'inizio del file, usate questo comando per saltare lì: **>**

```
'i
```

Quindi potete usare '' per saltare indietro dove eravate, oppure 'f per saltare sul testo dove state lavorando alla fine.
La scelta di i per inizio ed f per fine poteva essere diversa, queste lettere sono solo più facili da ricordare.

Potete usare questo comando per ottenere una lista dei marcatori: **>**

```
:marks
```

Potete notare alcuni marcatori speciali. Fra questi:

```
'    Posizione del cursore prima di effettuare un salto
"    Posizione del cursore quando avete editato il file l'ultima volta
[    Inizio dell'ultimo cambiamento
]    Fine dell'ultimo cambiamento
```

==========

Capitolo seguente: |usr_04.txt| Fare piccole modifiche

Copyright: si veda |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