usr_24.txt Per Vim version 7.3. Ultima modifica: 2006 Jul 23

VIM USER MANUAL - di Bram Moolenaar
Traduzione di questo capitolo: Giuliano Bordonaro

Inserzione rapida

Quando immettete del testo, Vim vi offre molti modi per ridurre il numero di caratteri immessi ed evitare errori di digitazione. Utilizzate il completamento in Insert mode per ripetere parole digitate in precedenza. Accorciate le parole lunghe in parole brevi. Scrivete caratteri che non sono presenti sulla tastiera.

Capitolo seguente: |usr_25.txt| Lavorare con testo formattato
Capitolo precedente: |usr_23.txt| Modifica di altri file
Indice: |usr_toc.txt|

==========

24.1 Effettuare correzioni

Del tasto **<BS>** abbiamo già parlato. Cancella i caratteri immediatamente precedenti il cursore. Il tasto **<Del>** fa la stessa cosa per i caratteri posti sotto (dopo) il cursore.
Avendo scritto qualche parola sbagliata potete usare **CTRL-W**:

```
The horse had fallen to the sky ~
                          CTRL-W
The horse had fallen to the ~
```

Se aveste sbagliato un'intera linea e voleste ripartire da capo, potreste usare **CTRL-U** per cancellarla. Ciò conserverebbe il testo posto dopo il cursore e il rientro. Solo il testo dal suo inizio alla posizione del cursore verrebbe cancellato. Con il cursore sulla "f" di "fallen" nella linea che segue premendo **CTRL-U** avverrebbe quanto segue:

```
The horse had fallen to the ~
               CTRL-U
fallen to the ~
```

Se trovaste un errore poche parole indietro bisognerebbe spostarvi il cursore per correggerlo. Ad esempio, avendo scritto così:

```
The horse had follen to the ground ~
```

Volendo cambiare "follen" in "fallen". Con il cursore a fine linea potreste scrivere così per correggerlo: >

```
                                    <Esc>4blraA

<       Uscire dall'Insert mode     <Esc>
        indietro di quattro parole       4b
        vai sulla "o"                      l
        sostituiscila con una "a"           ra
        riavvia l'Insert mode                 A
```

Un altro modo di farlo: >

```
            <C-Left><C-Left><C-Left><C-Left><Right><Del>a<End>

<       indietro di quattro parole      <C-Left><C-Left><C-Left><C-Left>
        vai sopra la "o"                   <Right>
```

```
        cancella la "o"                                  <Del>
        inserisci una "a"                                     a
        vai a fine linea                                       <End>
```

Questo impiega i tasti speciali per muoversi attorno, restando nell'Insert mode. Ciò assomiglia a quanto fareste con un editor non modale. È più facile da ricordare, ma richiede più tempo (dovreste spostarvi tra lettere e tasti cursore ed il tasto **<End>** risulta difficile da premere senza guardare la tastiera).

Questi tasti speciali sono molto utili per scrivere senza lasciare l'Insert mode. Allora il fatto di dover scrivere di più non costituisce un problema.

Una vista d'assieme dei tasti che potete usare nell'Insert mode:

<C-Home>	va all'inizio del file
<PageUp>	si sposta di uno schermo verso l'alto
<Home>	va all'inizio della linea
<S-Left>	si sposta di una parola a sinistra
<C-Left>	si sposta di una parola a sinistra
<S-Right>	si sposta di una parola a destra
<C-Right>	si sposta di una parola a destra
<End>	va alla fine della linea
<PageDown>	si sposta di uno schermo verso il basso
<C-End>	va alla fine del file

Ce ne sono alcuni in più, vedere |**ins-special-special**|.

==========

24.2 Evidenziare le corrispondenze

Quando scrivete un) potrebbe esservi utile vedere a quale (corrisponda. Per ottenere che Vim faccia ciò potete usare questo comando: >

```
:set showmatch
```

Quando scrivete un testo come "(example)", appena scritto) Vim sposterà per un attimo il cursore sulla corrispondente (, rimarrà lì per mezzo secondo, poi tornerà indietro dove stavate scrivendo.

Se non ci fosse la corrispondente (, Vim emetterebbe un suono. Così sapreste di aver dimenticato la (da qualche parte, o scritto una) di troppo.

La corrispondenza viene cercata anche per le coppie [] e {}. Non dovete aspettare per scrivere la prossima lettera, appena Vim trova la corrispondenza il cursore tornerà indietro e l'inserzione continuerà come prima.

Potete modificare questo tempo di verifica con l'opzione **'matchtime'**. Ad esempio, per far sì che Vim attenda per un secondo e mezzo: >

```
:set matchtime=15
```

Il tempo va specificato in decimi di secondo.

==========

24.3 Completamento

Vim può completare automaticamente le parole durante l'inserzione. Scrivete la prima parte di una parola, premete **CTRL-P**, e Vim provvederà al resto.

Immaginate, ad esempio, di stare scrivendo un programma in C e di voler scrivere:

```
        total = ch_array[0] + ch_array[1] + ch_array[2]; ~
```

Inizierete scrivendo quanto segue:

```
        total = ch_array[0] + ch_ ~
```

Ora dite a Vim di completare la parola usando il comando **CTRL-P**.
Vim cercherà una parola che inizi con ciò che sta davanti al cursore. In questo caso, si tratta di "ch_", che corrisponde con la parola ch_array.
Così se digitate **CTRL-P** otterrete:

```
        total = ch_array[0] + ch_array ~
```

Dopo un minimo sforzo di digitazione otterrete (terminando con uno spazio):

```
        total = ch_array[0] + ch_array[1] +  ~
```

Se adesso digitate **CTRL-P** Vim cercherà ancora una parola che completi quella prima del cursore. Poiché non c'è nulla davanti al cursore, verrà trovata la prima parola cercando all'indietro, che è "ch_array". Digitando ancora **CTRL-P** si troverà la prossima parola corrispondente, in questo caso "total". Un terzo **CTRL-P** cercherà ancora all'indietro. Se non vi fosse altro, l'editor, avendo terminato le parole, tornerebbe al testo originale, che è nullo. Un quarto **CTRL-P** farà ripartire l'editor con "ch_array".

Per cercare in avanti usate **CTRL-N**. Quando la ricerca avrà raggiunto la fine del file, **CTRL-N** e **CTRL-P** troveranno le stesse corrispondenze, ma in sequenza diversa. Significato: **CTRL-N** vuol dire Next-match e **CTRL-P** significa Previous-match.

Vim tenterà di tutto per trovare parole da completare. Di default, cercherà nei seguenti posti:

1. File attuale
2. File in altre finestre
3. Altri file caricati (buffer nascosti)
4. File non caricati (buffers inattivi)
5. Tag file
6. Tutti i file #included dal file corrente

OPZIONI

Potete personalizzare l'ordine di ricerca con l'opzione **'complete'**.

Uso dell'opzione **'ignorecase'**. Quando viene impostata, vengono ignorate le differenze tra maiuscole e minuscole nella ricerca delle corrispondenze.

Un'opzione speciale per il completamento è **'infercase'**. Risulta utile per trovare corrispondenze ignorando la distinzione tra maiuscole e minuscole (**'ignorecase'** deve essere impostata) ma utilizzando l'attributo della parola precedentemente digitata. Così, se scriveste "For", Vim troverebbe la corrispondenza in "fortunately", il risultato sarà "Fortunately".

COMPLETAMENTO DI ELEMENTI SPECIFICI

Se sapete cosa state cercando, potete usare questi comandi per completare a mezzo di un certo tipo di elementi:

CTRL-X CTRL-F	nomi di file
CTRL-X CTRL-L	talune linee
CTRL-X CTRL-D	definizioni di macro (anche in file inclusi)
CTRL-X CTRL-I	file corrente ed inclusi
CTRL-X CTRL-K	parole da un dizionario
CTRL-X CTRL-T	parole da un thesaurus
CTRL-X CTRL-]	marcature
CTRL-X CTRL-V	linea di comando di Vim

Dopo ciascuno di essi **CTRL-N** può essere usato per trovare la prossima corrispondenza, **CTRL-P** per quella precedente.
Una maggiore informazione per ciascuno di questi comandi qui: |**ins-completion**|.

COMPLETAMENTO DEI NOMI DEI FILE

Prendiamo ad esempio **CTRL-X CTRL-F**. Serve a trovare nomi di file. Esamina la directory corrente e mostra ciascun file che corrisponda con la parola davanti al cursore.
Ad esempio, immaginate di avere i seguenti file nella directory corrente:

```
main.c  sub_count.c  sub_done.c  sub_exit.c
```

Entrate in Insert mode ed iniziate a scrivere:

```
The exit code is in the file sub ~
```

A questo punto digitate il comando **CTRL-X CTRL-F**. Vim completa ora la parola

"sub" osservando i file nella directory attuale. La prima corrispondenza è sub_count.c. Ciò non è quello che volevate, così provate il prossimo file con CTRL-N. La corrispondenza è sub_done.c. Scrivendo ancora una volta CTRL-N vi darà sub_exit.c. Il risultato:

```
The exit code is in the file sub_exit.c ~
```

Se il nome del file iniziasse con / (Unix) o C:\ (MS-Windows) potreste trovare tutti i file del file system. Ad esempio, digitate "/u" e CTRL-X CTRL-F. Ciò troverà corrispondenza in "/usr" (ciò in Unix):

```
the file is found in /usr/ ~
```

Se adesso premete CTRL-N tornate a "/u". Invece, per accettare "/usr/" ed andare ad una directory di livello più basso, usate ancora CTRL-X CTRL-F:

```
the file is found in /usr/X11R6/ ~
```

I risultati dipenderanno da cosa si trova nel vostro file system, ovviamente. Le corrispondenze vengono ordinate alfabeticamente.

COMPLETAMENTO DI CODICE SORGENTE

I file di codice sorgente sono ben strutturati. Questo rende possibile effettuare dei completamenti in maniera intelligente. In Vim questa è stata definita completamento Omni. In altri programmi di videoscrittura è detta intellisense, ma la parole è un marchio commerciale.

La chiave al completamento Omni è CTRL-X CTRL-O. Ovviamente la O sta per Omni qui, per aiutare a ricordarlo. Vediamo un esempio di modifica ad un sorgente in C:

```
{ ~
    struct foo *p; ~
    p-> ~
```

Il cursore è posizionato dopo "p->". Ora digitate CTRL-X CTRL-O. Vim vi offrirà una lista di scelte, ossia gli elementi contenuti in "struct foo". Ciò è piuttosto differente dall'usare CTRL-P, che completerebbe qualsiasi parola, mentre solo elementi di "struct foo" sono validi qui.

Per far funzionare il completamento Omni può essere necessaria qualche preparazione. Assicuratevi almeno che i plugin per un tipo_file siano abilitati. Il vostro file vimrc dovrebbe contenere una linea come questa: >

```
filetype plugin on
```

Oppure: >

```
filetype plugin indent on
```

Per il sorgente C dovrete creare un file di tag ed impostare l'opzione 'tags'. La spiegazione si trova in |ft-c-omni|. Per altri tipi di file dovrete procedere analogamente, guardate qui sotto |compl-omni-filetypes|. Questo funziona solo per tipi di file specifici. Controllate il valore dell'opzione 'omnifunc' per trovare se l'opzione potrebbe funzionare per il file che state modificando.

==========

24.4 Ripetizione di un inserimento

Se premete CTRL-A, l'editor inserisce il testo scritto l'ultima volta che vi trovavate nell'Insert mode.
Ad esempio, supponete di avere un file che inizi con i seguenti:

```
"file.h" ~
/* Main program begins */ ~
```

Modificate questo file inserendo "#include " all'inizio della prima linea:

```
#include "file.h" ~
/* Main program begins */ ~
```

Scendete all'inizio della linea seguente con il comando "j^". Inserite una

nuova linea "#include". Così scrivete: >

```
i CTRL-A
```

Ne deriverà quanto segue:

```
#include "file.h" ~
#include /* Main program begins */ ~
```

È stato incluso "#include " perché **CTRL-A** inserisce il testo dell'inserimento precedente. Ora scrivete "main.h"**<Enter>** per completare la linea:

```
#include "file.h" ~
#include "main.h" ~
/* Main program begins */ ~
```

Il comando **CTRL-@** produce un **CTRL-A** ed esce dall'Insert mode. È un modo veloce per fare lo stesso inserimento un'altra volta.

==========

24.5 Copiare da un'altra linea

Il comando **CTRL-Y** inserisce il carattere prima del cursore. Risulta utile dovendo duplicare una linea precedente. Ad esempio, posta questa linea di codice C:

```
b_array[i]->s_next = a_array[i]->s_next; ~
```

Adesso dovete scrivere la stessa linea, ma con "s_prev" invece di "s_next". Andate a capo e premete **CTRL-Y** 14 volte, sino a quando giungerete alla "n" di "next":

```
b_array[i]->s_next = a_array[i]->s_next; ~
b_array[i]->s_ ~
```

Adesso scrivete "prev":

```
b_array[i]->s_next = a_array[i]->s_next; ~
b_array[i]->s_prev ~
```

Continuate premendo **CTRL-Y** sino al prossimo "next":

```
b_array[i]->s_next = a_array[i]->s_next;~
b_array[i]->s_prev = a_array[i]->s_ ~
```

Adesso scrivete "prev;" per terminare.

Il comando **CTRL-E** fa come **CTRL-Y** ad eccezione del fatto che inserisce il carattere dopo il cursore.

==========

24.6 Inserire un registro

Il comando **CTRL-R {register}** inserisce i contenuti del registro. Ciò risulta utile per evitare di dover scrivere una parola lunga. Ad esempio, se volete scrivere:

```
r = VeryLongFunction(a) + VeryLongFunction(b) + VeryLongFunction(c) ~
```

Il nome della funzione è definito entro un file diverso. Aprite questo file e muovete il cursore all'inizio del nome della funzione, copiatelo ora nel registro v: >

```
"vyiw
```

"v è la specificazione del registro, "yiw" sta per yank-inner-word. Ora aprite il file in cui volete inserire la nuova linea e premete le prime lettere:

```
r = ~
```

Adesso, con **CTRL-R** v verrà inserito il nome della funzione:

```
        r = VeryLongFunction ~
```

Continuate a scrivere caratteri entro il nome della funzione ed usate ancora due volte **CTRL-R** v.
   Potreste fare lo stesso per il completamento. Usare un registro è utile se ci fossero troppo parole inizianti con lo stesso carattere.

Se il registro contenesse caratteri come **<BS>** od altri caratteri speciali, essi verrebbero interpretati come se fossero stati digitati dalla tastiera. Se non volete che ciò accada (volete che venga inserito davvero nel testo il **<BS>**), usate il comando **CTRL-R CTRL-R {register}**.

==
24.7 Abbreviazioni

Un'abbreviazione è una parola breve che prende il posto di una lunga. Ad esempio, "ad" sta per "advertisement". Vim permette di scrivere un'abbreviazione e la espanderà automaticamente.
   Per dire a Vim di espandere "ad" in "advertisement" ogni volta che venga inserita, usate il comando seguente: >

```
        :iabbrev ad advertisement
```

Adesso, scrivendo "ad", tutta la parola "advertisement" verrà inserita nel testo. Lo otterrete scrivendo un carattere che non fa parte della parola, ad esempio uno spazio:

```
        What Is Entered         What You See
        I saw the a             I saw the a ~
        I saw the ad            I saw the ad ~
        I saw the ad<Space>     I saw the advertisement<Space> ~
```

L'espansione non avviene scrivendo solo "ad". Ciò vi permette di scrivere una parola come "add", che non deve essere espansa. Solo le parole intere vengono prese in esame per le abbreviazioni.

ABBREVIARE DIVERSE PAROLE

È possibile definire un'abbreviazione che si sviluppi in diverse parole. Ad esempio, per definire "JB" come "Jack Benny", usate il seguente comando: >

```
        :iabbrev JB Jack Benny
```

Come programmatore, uso due abbreviazioni abbastanza insolite: >

```
        :iabbrev #b /****************************************
        :iabbrev #e <Space>****************************************/
```

Servono per generare commenti incorniciati. Il commento parte con #b, che disegna la linea sopra. Poi si scrive il testo del commento e si usa #e per disegnare la linea sotto.
   Attenzione al fatto che l'abbreviazione #e inizia con uno spazio. In altre parole, i primi due caratteri sono spazio-asterisco. Di solito Vim ignora gli spazi tra l'abbreviazione e l'espansione. Per evitare questo problema, sillabo "space" come sette caratteri: <, S, p, a, c, e, >.

```
        Nota:
        ":iabbrev" è una parola lunga da scrivere. ":iab" va meglio.
        Ciò significa abbreviare il comando abbreviate!
```

CORREZIONE DEGLI ERRORI DI SCRITTURA

È comunissimo ripetere spesso lo stesso errore di digitazione. Ad es., scrivere "teh" invece di "the". Potete rimediare con un'abbreviazione: >

```
        :abbreviate teh the
```

Potete aggiungerne un'intera lista. Aggiungetene una ogni volta che scoprite un errore comune.

ELENCARE LE ABBREVIAZIONI

Il comando ":abbreviate" elenca le abbreviazioni:

```
:abbreviate
i  #e             ************************************/
i  #b            /************************************
i  JB            Jack Benny
i  ad            advertisement
!  teh           the
```

La lettera "i" nella prima colonna indica l'Insert mode. Queste abbreviazioni sono attive soltanto nell'Insert mode. Altri possibili caratteri sono:

```
c       Command-line mode                       :cabbrev
!       Entrambi, Insert e Command-line mode    :abbreviate
```

Poiché le abbreviazioni non sono utili spesso nel Command-line mode, userete preferibilmente il comando ":iabbrev". Ciò eviterà, ad esempio, che "ad" venga espanso quando state scrivendo un comando come: >

```
:edit ad
```

CANCELLARE LE ABBREVIAZIONI

Per eliminare un'abbreviazione usate il comando ":unabbreviate". Supponete di avere la seguente abbreviazione: >

```
:abbreviate @f fresh
```

La potete rimuovere con il seguente comando: >

```
:unabbreviate @f
```

Sino a quando non farete ciò @f verrà espanso in "fresh". Non preoccupatevene, Vim lo capisce comunque (eccetto se aveste un'abbreviazione per "fresh", ma essa fosse molto differente).
Per eliminare tutte le abbreviazioni: >

```
:abclear
```

":unabbreviate" ed ":abclear" sono delle varianti per l'Insert mode ("iunabbeviate ed ":iabclear") e per il Command-line mode (":cunabbreviate" e "cabclear").

RIMAPPATURA DELLE ABBREVIAZIONI

C'è una cosa da tenere in considerazione quando definite un'abbreviazione: La stringa risultante non può essere mappata. Ad esempio: >

```
:abbreviate @a adder
:imap dd disk-door
```

Se adesso scriveste @a, otterreste "adisk-doorer". Non è quanto volevate. Per evitarlo, impiegate il comando ":noreabbrev". Fa come ":abbreviate", ma evita che la stringa risultante venga usata per la mappatura: >

```
:noreabbrev @a adder
```

Fortunatamente è raro che il risultato di un'abbreviazione venga mappato.

==========

24.8 Inserimento di caratteri speciali

Il comando **CTRL-V** serve ad inserire letteralmente il prossimo carattere. In altre parole, qualsiasi significato speciale il carattere abbia verrà ignorato. Ad esempio: >

```
CTRL-V <Esc>
```

Inserisce un carattere di escape. Così non dovrete lasciare l'Insert mode.

(Non dovete mettere lo spazio dopo **CTRL-V**, è solo per renderlo più leggibile).

> Nota:
> In MS-Windows **CTRL-V** viene usato per incollare del testo. Usate **CTRL-Q** invece di **CTRL-V**. Su Unix, d'altronde, **CTRL-Q** non funziona su alcuni terminali perché ha un significato speciale.

Potete anche usare il comando **CTRL-V {numeri}** per inserire un carattere contenente numeri decimali **{numeri}**. Ad esempio, il carattere numero 127 è il carattere **<Del>** (ma non necessariamente il tasto **<Del>**!). Per inserire **<Del>** scrivete: >

```
CTRL-V 127
```

In questo modo potete inserire caratteri in numero superiore a 255. Quando scrivete meno di due cifre, un carattere non cifra dovrà ultimare il comando. Per evitare di scrivere il carattere non cifra anteponete uno o due zeri per fare tre cifre.
Tutti i seguenti comandi inseriscono un **<Tab>** seguito da un punto:

```
CTRL-V 9.
CTRL-V 09.
CTRL-V 009.
```

Per inserire un carattere in esadecimale, usate una "x" dopo il **CTRL-V**: >

```
CTRL-V x7f
```

Ciò va anche oltre i 255 caratteri (**CTRL-V** xff). Potete usare "o" per scrivere un carattere come numero ottale ed altri due metodi vi consentiranno di scrivere numeri a 16 o 32 bit (e.g., per un carattere Unicode): >

```
CTRL-V o123
CTRL-V u1234
CTRL-V U12345678
```

==========

24.9 Digrammi

Taluni caratteri non esistono sulla tastiera. Ad esempio, il carattere di copyright (©). Per scrivere questi caratteri con Vim, utilizzerete dei digrammi, ove due caratteri ne rappresentano uno. Per inserire un ©, ad esempio, premerete tre tasti: >

```
CTRL-K Co
```

Per sapere quali digrammi siano disponibili usate il seguente comando: >

```
:digraphs
```

Vim farà vedere la tabella dei digrammi. Eccone tre linee:

```
AC ~_ 159  NS |  160  !I ¡  161  Ct ¢  162  Pd £  163  Cu ¤  164  Ye ¥  165 ~
BB ¦  166  SE §  167  ': ¨  168  Co ©  169  -a ª  170  << «  171  NO ¬  172 ~
-- -  173  Rg ®  174  'm ¯  175  DG °  176  +- ±  177  2S ²  178  3S ³  179 ~
```

Ciò mostra, per esempio, che il carattere-digramma che otterrete premendo **CTRL-K** Pd è il carattere (£). Si tratta del carattere numero 163 (decimale).
Pd è l'abbreviazione di Pound. Molti digrammi sono fatti in modo da darvi un suggerimento circa il carattere che produrranno. Guardando la lista ne capirete la logica.
Si può scambiare il primo col secondo carattere, se non esiste un altro digramma con la stessa combinazione. Così funzionerà anche **CTRL-K** dP. Se non vi fosse un digramma per "dP" Vim cercherà anche un digramma per "Pd".

> Nota:
> I caratteri-digramma dipendono dall'assieme di caratteri che Vim pensa usiate. In MS-DOS è diverso che in MS-Windows. Usate sempre ":digraphs" per trovare quali caratteri-digramma sono attualmente disponibili.

Potete definire i vostri digrammi. Esempio: >

```
:digraph a" ä
```

Definisce che **CTRL-K** a" inserisca un carattere ä. Potete anche specificare il carattere con un numero decimale. Ciò definisce il medesimo digramma: >

```
:digraph a" 228
```

Troverete ulteriori informazioni circa i caratteri digrammi in: |digraphs|
Un altro modo per inserire caratteri speciali è con una keymap. Di più sull'argomento: |45.5|

==========

24.10 Comandi in Normal mode

L'Insert mode offre un numero limitato di comandi. In Normal mode è disponibile assai di più. Se ne voleste usare uno, normalmente abbandonereste l'Insert mode con **<Esc>**, eseguireste il comando in Normal mode, e rientrereste in Insert mode con "i" od "a".
C'è una via più rapida. Con **CTRL-O {comando}** potete eseguire tutti i comandi Normal mode restando nell'Insert mode. Ad esempio, per cancellare dalla posizione del cursore alla fine della linea: >

```
CTRL-O D
```

Potete eseguire un solo comando Normal mode in questo modo. Ma potete specificare un registro od un conto. Un esempio più complicato: >

```
CTRL-O "g3dw
```

Cancella a partire dalla terza parola entro il registro g.

==========

Capitolo seguente: |usr_25.txt| Lavorare con testo formattato

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