

\*usr\_20.txt\* Per Vim version 7.2. Ultima modifica: 2006 Apr 24

VIM USER MANUAL - di Bram Moolenaar  
Traduzione di questo capitolo: Fabio Teatini e Roberta Fedeli

Immissione rapida dei comandi sulla linea di comando

Vim ha alcune funzionalità generali che rendono più semplice l'immissione di comandi. I comandi possono essere abbreviati, modificati e ripetuti.

Il meccanismo del completamento è disponibile quasi sempre.

20.1	Elaborazione della linea di comando
20.2	Abbreviazioni dei comandi
20.3	Completamento automatico dei comandi
20.4	Cronologia dei comandi
20.5	Finestra della linea di comando

Capitolo seguente:	usr_21.txt	Andarsene e ritornare
Capitolo precedente:	usr_12.txt	Trucchi ingegnosi
Indice:	usr_toc.txt	

## \*20.1\* Elaborazione della linea di comando

Quando, per immettere un comando, digitate il carattere dei due-punti (:) o effettuate la ricerca di una stringa con / o ?, Vim pone il cursore in fondo allo schermo; questa è la posizione in cui inserire i comandi o le chiavi di ricerca, ed è chiamata Command line. Anche quando viene usata per inserire un comando di ricerca.

Il modo più semplice per modificare il comando digitato è quello di premere il tasto **<BS>**, così da cancellare il carattere a sinistra del cursore. Per cancellare un altro carattere precedentemente digitato, spostate il cursore coi tasti-freccia.

Per esempio, immaginate di aver già digitato il comando:

```
:s/col/pig/
```

Se ora, prima di premere **<Invio>**, voi voleste che "col" fosse trasformato in "cow", digitate **<Left>** cinque volte. Il cursore, ora, si trova proprio dopo "col". Per effettuare la correzione digitate **<BS>** e "w":

```
:s/cow/pig/
```

Ora potete premere **<Enter>** direttamente, senza spostare il cursore alla fine della linea prima di eseguire il comando.

Ecco i tasti utilizzati più spesso per spostarsi entro la riga di comando:

<b>&lt;Left&gt;</b>	un carattere a sinistra
<b>&lt;Right&gt;</b>	un carattere a destra
<b>&lt;S-Left&gt;</b> o <b>&lt;C-Left&gt;</b>	una parola a sinistra
<b>&lt;S-Right&gt;</b> o <b>&lt;C-Right&gt;</b>	una parola a destra
<b>CTRL-B</b> o <b>&lt;Home&gt;</b>	all'inizio della linea di comando
<b>CTRL-E</b> o <b>&lt;End&gt;</b>	alla fine della linea di comando

Note:

**<S-Left>** (tasto-freccia a sinistra con tasto delle Maiuscole premuto) e **<C-Left>** (tasto-freccia a sinistra con tasto Control premuto) non funzionano su tutte le tastiere. Ciò vale anche per le altre combinazioni ottenute coi tasti Maiuscola e Control.

Per muovere il cursore potete utilizzare anche il mouse.

## CANCELLAZIONE

Come già detto, **<BS>** cancella il carattere precedente al cursore. Per cancellare un'intera parola si usa **CTRL-W**.

```
/the fine pig ~
```

**CTRL-W**

/the fine ~

**CTRL-U** cancella tutto il testo, permettendo di ricominciare tutto da capo.

#### SOVRASCRITTURA

Esistono due possibilità per l'inserimento di nuovi caratteri: la prima permette di inserire nuovi caratteri senza cancellare quelli esistenti; la seconda fa sovrascrivere i nuovi caratteri su quelli esistenti. Il tasto **<Insert>** permette di passare dalla prima modalità alla seconda. Scrivete questo testo:

/the fine pig ~

Ora muovete il cursore all'inizio della parola "fine" premendo **<S-Left>** due volte (o **<Left>** otto volte, se **<S-Left>** non funziona). Quindi, premete **<Insert>** per passare alla funzione di sovrascrittura e digitate "great":

/the greatpig ~

Oops, è andato perso lo spazio. In questo caso non usate **<BS>**, perché cancellereste la "t" (diversamente dalla modalità di Sovrascrittura). Invece, premete **<Insert>** per passare dalla sovrascrittura all'inserimento, e digitate lo spazio:

/the great pig ~

#### ANNULLAMENTO

Quando volete annullare un comando : oppure / in fase di digitazione, premete **CTRL-C** o **<Esc>**.

Note:

**<Esc>** è il tasto universalmente usato come «uscita». Sfortunatamente, nel buon vecchio Vi la digitazione di **<Esc>** in una riga di comando eseguiva il comando! Poiché ciò potrebbe essere considerato un bug, Vim usa **<Esc>** per cancellare il comando.

Ma usando l'opzione '**cpoptions**' si può ottenere la compatibilità con VI. **<Esc>** funziona in modalità compatibile con Vi anche quando si usa un mapping (scritto per Vi).

In definitiva, il metodo che funziona sempre è quello del **CTRL-C**.

Se siete all'inizio della linea di comando, premendo **<BS>** annullerete il comando. È equivalente a cancellare i caratteri ":" o "/" posti all'inizio della riga.

=====

**\*20.2\*** Scorciatoie per la linea di comando

Alcuni comandi ":" sono veramente lunghi. Si è già detto che ":substitute" può essere abbreviato come ":s". Questa è una regola generale: tutti i comandi ":" possono essere abbreviati.

Di quanto può essere abbreviato un comando? Ci sono 26 lettere, ma molti più comandi. Per esempio, anche ":set" inizia con ":s", ma ":s" non avvia un comando ":set". Invece ":set" può essere abbreviato con ":se".

Quando due comandi hanno la stessa forma abbreviata, in realtà essa funzionerà solo per uno di essi. Non c'è un metodo logico per individuarlo: è qualcosa che va imparato. Nei file di help viene indicata la forma abbreviata più corta che funziona. Per esempio: >

**:s[ubstitute]**

Significa che la forma più corta per ":substitute" è ":s". I caratteri successivi sono opzionali. Quindi funzionano anche ":su" e ":sub".

In questo manuale dell'utente si userà sia il nome intero del comando, sia una forma abbreviata comunque intelligibile. Per esempio, ":function" potrebbe essere abbreviato in ":fu", ma poiché molte persone non capirebbero cosa indica tale abbreviazione, useremo ":fun" (Vim non ha un comando ":funny", altrimenti anche ":fun" potrebbe ingenerare confusione).

Per quel che riguarda gli script di Vim, vi raccomando di scrivere il nome intero dei comandi. Ciò permette di interpretarli più facilmente quando si effettuano modifiche successive. Si può fare eccezione per alcuni comandi usati molto spesso, come `":w"` (`":write"`) e `":r"` (`":read"`).

Una forma particolarmente ambigua è `":end"`, che si adatta a `":endif"`, `":endwhile"` e `":endfunction"`. Perciò, usate sempre il nome intero.

#### NOMI BREVI DELLE OPZIONI

Nel manuale dell'utente sono illustrate le opzioni con i nomi lunghi. Per molte opzioni si possono usare anche nomi brevi. Contrariamente a quel che avviene per i comandi `":"`, c'è un solo nome breve funzionante. Ad esempio, il nome breve per `'autoindent'` è `'ai'`. Perciò, i due comandi seguenti fanno lo stesso lavoro: >

```
:set autoindent
:set ai
```

Un elenco dei nomi dei comandi (lunghi e brevi) si trova qui: `|option-list|`.

=====

**\*20.3\*** Completamento della linea di comando

Questa funzionalità di Vim è una di quelle che, da sola, giustifica l'adozione di Vim da parte degli utenti di Vi. Una volta che l'avrete usata, non potrete più farne a meno.

Supponete di avere una directory contenente questi file:

```
info.txt
intro.txt
corpodeltesto.txt
```

Per elaborare l'ultimo file, potete usare il comando: >

```
:edit corpodeltesto.txt
```

e noterete che è facile sbagliarne la digitazione. Un modo assai più rapido è:

```
:edit c<Tab>
```

e otterrete lo stesso effetto di prima. Che è successo? Il tasto `<Tab>` effettua il completamento della parola che precede il cursore. Nel nostro caso: `"b"`.

Vim cerca nella directory e trova soltanto un file il cui nome inizia per `"b"`; non essendoci ambiguità, Vim completa al posto vostro il nome del file e lo apre.

Ora digitate: >

```
:edit i<Tab>
```

Vim, ora, emetterà un segnale acustico e presenterà il comando: >

```
:edit info.txt
```

Il segnale acustico significa che Vim ha trovato più di una corrispondenza.

Per questo motivo aprirà il primo file (in ordine alfabetico) che corrisponde.

Se premete ancora `<Tab>`, otterrete: >

```
:edit intro.txt
```

Così, se il primo `<Tab>` non evidenzia il file che stavate cercando, premetelo ancora. Se esistono più nomi corrispondenti, li vedrete tutti, uno per uno.

Se premete `<Tab>` sull'ultimo nome corrispondente, tornerete a quello che avevate digitato inizialmente: >

```
:edit i
```

Allora si ricomincia daccapo. Così Vim effettua un ciclo attraverso l'elenco delle corrispondenze.

Con **CTRL-P** potrete scorrere la lista nel verso opposto:

```

<-----<Tab>-----+
|
:edit i      <Tab> -->      :edit info.txt      <Tab> -->      :edit intro.txt
      <-- CTRL-P      <-- CTRL-P
|
+-----CTRL-P----->

```

#### CONTESTO

Quando digitate ":set i" invece che ":edit i" e premete **<Tab>**, si ottiene: >

```
:set icon
```

Ehi! Perché non compare ":set info.txt"? Ciò avviene perché il completamento di Vim è sensibile al contesto. Il tipo di parole cercate da Vim dipende dal comando iniziale. Vim sa che subito dopo un comando ":set" non potete usare un nome di file; è consentito solo un nome d'opzione.

Anche qui, se digitate più volte il tasto **<Tab>**, Vim effettuerà una scansione ciclica attraverso tutti i valori corrispondenti. Ce ne sono parecchi, per cui è meglio digitare altri caratteri: >

```
:set isk<Tab>
```

Diventa: >

```
:set iskeyword
```

Ora digitate "=" e premete **<Tab>**: >

```
:set iskeyword=@,48-57,_,192-255
```

Ciò che accade è che Vim inserisce il vecchio valore dell'opzione. Adesso potete modificarla.

L'uso di **<Tab>** porta al completamento con ciò che Vim si aspetta debba comparire nella posizione corrispondente. Provate semplicemente a vedere come funziona. In certe situazioni otterrete qualcosa di diverso da quanto desiderato. Ciò è dovuto al fatto che Vim non conosce i vostri desideri; oppure il completamento potrebbe non essere stato implementato per quella determinata situazione. In quel caso otterrete il semplice inserimento di un **<Tab>** (visualizzato come ^I).

#### ELENCO DI CORRISPONDENZE

Quando esistono molte corrispondenze, vorrete vederle tutte insieme. Potete farlo premendo **CTRL-D**. Per esempio, premendo **CTRL-D** dopo aver digitato >

```
:set is
```

si ottiene: >

```

:set is
incsearch  isfname  isident  iskeyword  isprint
:set is

```

Vim elenca le corrispondenze e ritorna al testo da voi digitato, così che possiate passare in rassegna l'elenco e scegliere ciò che volete. Se non ci sono corrispondenze, potete usare **<BS>** per correggere la parola. Se le corrispondenze sono molte, prima di premere **<Tab>** digitate qualche altro carattere di ciò che vi interessa.

Se ci avete fatto caso, noterete che "incsearch" non inizia con "is". In questo caso, "is" è il nome breve di "incsearch". (Molte opzioni hanno nomi sia brevi che lunghi.) Vim è sufficientemente scaltro da sapere che voi potreste aver voluto espandere il nome breve dell'opzione nel suo nome lungo.

C'E' DI PIU'

Il comando **CTRL-L** completa la parola trasformandola nella stringa più lunga e non ambigua. Se digitate ":edit i" ed esistono i file "info.txt" e "info\_backup.txt" otterrete ":edit info".

Per modificare il meccanismo di completamento, potete usare l'opzione **'wildmode'**.

L'opzione **'wildmenu'** permette di produrre un elenco corrispondenze in forma di menù.

L'opzione **'suffixes'** permette di precisare i file meno importanti che possono essere relegati alla fine dell'elenco.

L'opzione **'wildignore'** permette di indicare i file da non elencare affatto.

Altre informazioni in merito si trovano qui: |[cmdline-completion](#)|

#### =====

#### \*20.4\* Cronologia dei comandi

La cronologia dei comandi (o history) è stata brevemente citata nel capitolo 3. La cosa più semplice a riguardo è l'utilizzo del tasto **<Up>** per richiamare i precedenti comandi; il tasto **<Down>** permette di scorrere i comandi in avanti.

In realtà di cronologie ne esistono quattro. Qui tratteremo quella dei comandi ":", e quelle dei comandi di ricerca "/" e "?". I comandi "/" e "?" condividono la stessa cronologia, essendo entrambi comandi di ricerca. Le altre due cronologie sono quelle delle espressioni e delle linee di input per la funzione input().

|[cmdline-history](#)|

Supponiamo di aver immesso un comando ":set", di aver digitato più di dieci comandi ":" e poi di voler ripetere il comando ":set" di prima. Potreste premere ":" e dieci volte **<Up>**, ma c'è un modo più rapido: >

```
:se<Up>
```

Vim tornerà al precedente comando che inizia con "se". Avete buone probabilità che si tratti del comando ":set" che cercavate. Se non altro, vi sarà risparmiato di premere tante volte il tasto **<Up>** (a meno che non abbiate dato solo comandi di tipo ":set").

Il tasto **<Up>** userà il testo digitato fino a quel momento e lo confronterà con le linee presenti nella cronologia. Solo le linee corrispondenti saranno usate.

Se non trovate le linee che cercavate, usate **<Down>** e tornerete a quelle digitate inizialmente, per modificarle. Oppure usate **CTRL-U** per ricominciare tutto da capo.

Per vedere tutte le linee della cronologia: >

```
:history
```

Questa è la cronologia dei comandi ":". La cronologia dei comandi di ricerca viene mostrata con il comando: >

```
:history /
```

**CTRL-P** ha lo stesso effetto di **<Up>**, tranne per il fatto che non importa quel che avete già scritto. Lo stesso vale per **CTRL-N** e **<Down>**. **CTRL-P** sta per "precedente", **CTRL-N** per "next" (successivo).

#### =====

#### \*20.5\* Finestra della linea di comando

La digitazione del testo nella linea di comando funziona in modo diverso dalla digitazione in Insert mode: a molti comandi non è permesso di modificare il testo. Per la maggior parte dei programmi questo non è un problema, ma a volte vi troverete a digitare comandi complessi. E' in questo caso che torna utile la finestra della linea di comando.

La finestra della linea di comando si apre con questo comando: >

**q:**

Con esso, Vim apre una (piccola) finestra in basso, che contiene la cronologia della linea di comando e una linea vuota in fondo:

```
+-----+
| other window                               |
| ~                                           |
| file.txt=====                           |
| :e c                                       |
| :e config.h.in                           |
| :set path=.,/usr/include,,                |
| :set iskeyword=@,48-57,_,192-255          |
| :set is                                    |
| :q                                         |
| :                                           |
| command-line=====                       |
+-----+
```

Ora siete in Normal mode. Qui potete usare i tasti "hjkl" per spostarvi col cursore. Per esempio, spostatevi verso l'alto con "5k" fino alla linea con ":e config.h.in". Digitate "\$h" per muovervi fino alla "i" di "in" e digitate "cwout". Ora la linea è stata modificata in:

**:e config.h.out ~**

Premete **<Enter>** e questo comando verrà eseguito, dopodiché la finestra della linea di comando si chiuderà da sé.

Il comando **<Enter>** eseguirà la linea su cui si trova il cursore, senza badare se Vim si trova in Insert mode o Normal mode.

Le modifiche, effettuate in finestra della linea di comando, saranno perdute; non saranno reperibili nella cronologia dei comandi, tranne per il comando da voi eseguito (ciò vale per tutti i comandi mandati in esecuzione).

La finestra della linea di comando è utilissima per ottenere una panoramica della cronologia dei comandi, cercare nei comandi qualcosa di simile a quello che serve, fare le dovute modifiche ed eseguire il tutto. Per trovare quel che si vuole si può utilizzare esplicitamente un comando di ricerca.

Nell'esempio precedente avreste potuto usare il comando di ricerca "?config" per trovare il comando più recente contenente "config". C'è un che di strano nell'usare un comando di ricerca per effettuare ricerche nella finestra dei comandi.

Quando digitate questo comando di ricerca non potrete aprire un'altra finestra della linea di comando, perché ne viene supportata solo una.

=====

Capitolo seguente: |[usr\\_21.txt](#)| Andarsene e ritornare

Copyright: vedere |[manual-copyright](#)| vim:tw=78:ts=8:ft=help:norl:

Per segnalazioni scrivere a vimdoc.it at gmail dot com  
oppure ad Antonio Colombo azcl00 at gmail dot com