intro.txt Per **Vim version 8.0.** Ultima modifica: 2017 Jan 24

VIM REFERENCE MANUAL di Bram Moolenaar

Traduzione di questo testo: Antonio Colombo e Giuliano Bordonaro

Introduzione a Vim *ref* *reference*

1. Introduzione |intro|
2. Vim su internet |internet|
3. Ringraziamenti |credits|
4. Notazione |notation|
5. Modalità, introduzione |vim-modes-intro|
6. Cambiare modalità |mode-switching|
7. Contenuti della finestra |window-contents|
8. Definizioni |definitions|

==========

1. Introduzione *intro*

Vim sta per VI Migliorato. Era Vi IMitato, ma i miglioramenti sono tanti da giustificare un cambio di nome. Vim è un editore di testi dotati di quasi tutti i comandi del programma Unix "Vi", assieme a molti altri comandi nuovi. È molto utile per modificare programmi ed altri tipi di testo.
Tutti i comandi si possono dare usando la tastiera, e questo consente di mantenere le dita sopra la tastiera, e gli occhi rivolti verso lo schermo. Per chi lo desidera, è possibile usare il mouse, e c'è una versione GUI con barre di scorrimento e menù (vedere |**gui.txt**|).

Un sommario di questo manuale si può trovare nel file "help.txt", |**help.txt**|. Ci si può arrivare da dentro Vim con il tasto <Help> o <F1> e col comando |**:help**| (digitare ":help", senza '|' o '"').
L'opzione 'helpfile' si può impostare col nome del file di help, se questo non si trova nella posizione predefinita. Potete passare ad un argomento allo stesso modo con cui usate i tag: Usate CTRL-] per passare ad un argomento il cui nome sia sotto il cursore, usate CTRL-T per tornare indietro.

In tutto questo manuale le differenze tra Vi e Vim verranno riportate tra parentesi graffe, così: {Vi non ha un help online}. Vedere |**vi_diff.txt**| per un riassunto delle differenze tra Vim e Vi.

pronounce
Vim si pronuncia come una parola, come Jim, non vi-i-emme. È scritto con la maiuscola iniziale, trattandosi di un nome, sempre come Jim.

Questo manuale si riferisce a Vim su diverse macchine. Ci possono essere piccole differenze tra computer e terminali diversi. Oltre alle annotazioni date in questo documento, esiste un documento separato per ciascun sistema supportato, vedere |**sys-file-list**|.

Questo manuale è il riferimento per tutti i comandi e le opzioni di Vim. Non si tratta di un'introduzione all'uso di Vi o di Vim, risulta un tantino complesso, qua e là. Per principianti c'è un pratico |**tutor**|. Per imparare ad usare Vim, leggete il manuale per l'utente |**usr_toc.txt**|.

book
Ci sono molti libri su Vi che hanno una sezione dedicata ai principianti. Ci sono due libri che posso consigliare:

"Vim - Vi Improved" di Steve Oualline

Questo è il primissimo libro completamente dedicato a Vim. È ottimo per principianti. I comandi che vengono usati più spesso sono spiegati con figure ed esempi. Quelli che si usano meno sono spiegati lo stesso, le più avanzate caratteristiche sono riassunte. C'è un indice comprensivo ed un riferimento veloce. Parti di questo libro sono state incluse nel manuale dell'utente |**frombook**|.
Pubblicato da New Riders Publishing. ISBN: 0735710015
Per una maggiore informazione provate uno di questi:
http://iccf-holland.org/click5.html
http://www.vim.org/iccf/click5.html

"Learning the Vi editor" di Linda Lamb e Arnold Robbins

È un libro su Vi che comprende un capitolo su Vim (nella sesta edizione).
I primi passi con Vi sono spiegati assai bene. I comandi che Vim aggiunge
sono citati solo brevemente. C'è anche una traduzione in tedesco.
Pubblicato da O'Reilly. ISBN: 1-56592-426-6.

==
2. Vim su internet *internet*

www *WWW* *faq* *FAQ* *distribution* *download*
Le pagine Vim contengono l'informazione più recente su Vim. Contengono anche
dei link alle più recenti versioni di Vim. La FAQ è una lista delle domande
poste più frequentemente. Leggetele se avete dei problemi.

Vim home page: http://www.vim.org/
Vim FAQ: http://vimdoc.sf.net/
Downloading: ftp://ftp.vim.org/pub/vim/MIRRORS

Newsgroup Usenet dove si discute di Vim: *news* *usenet*
comp.editors
Questo gruppo è anche per altri editori. Se scrivete di Vim, non scordatevi
di specificarlo.

mail-list *maillist*
Ci sono molte Mailing List per Vim:
<vim@vim.org> *vim-use* *vim_use*
Per discussioni su come usare le versioni esistenti di Vim: Utili
mappature, domande, risposte, dove trovare una versione particolare,
etc. C'è un certo numero di persone che controlla questa lista e
risponde a domande, anche di principianti. Non esitate a formulare la
vostra domanda qui.
<vim-dev@vim.org> *vim-dev* *vim_dev* *vimdev*
Per discussioni su come cambiare Vim: Nuove caratteristiche, porting, patch,
versioni beta-test, etc.
<vim-announce@vim.org> *vim-announce* *vim_announce*
Annunci delle nuove versioni di Vim; anche per versioni beta-test
e porting a sistemi diversi. Questa lista è in sola lettura.
<vim-mac@vim.org> *vim-mac* *vim_mac*
Per discussioni su come usare e perfezionare la versione Macintosh di
Vim.

Vedere http://www.vim.org/maillist.php

NOTA:
- Potete inviare messaggi a queste liste soltanto se vi siete registrati!
- Dovete spedire i messaggi dallo stesso posto da dove avete effettuato al
registrazione (per evitare lo spam).
- La massima dimensione di un messaggio è di 40000 caratteri.

subscribe-maillist
Se volete partecipare, spedite un messaggio a
<vim-subscribe@vim.org>
Accertatevi che il vostro indirizzo "From:" sia giusto. Così il server della
lista potrà aiutarvi su come registrarsi.

maillist-archive
Per ulteriori informazioni e per accedere agli archivi, potete consultare la
pagina dedicata alle maillist Vim:
http://www.vim.org/maillist.php

Rapporti di bug: *bugs* *bug-reports* *bugreport.vim*

Ci sono due modi per comunicare bug, ed entrambi sono funzionanti:
1. Spedire i rapporti sui bug a: Vim Developers <vim-dev@vim.org>
Questa è una Mailing List, è necessario diventare prima membri per poter
inviare messaggi alla lista e saranno in molti a ricevere il messaggio.
Se preferite evitarlo, p.es. perché il messaggio riguarda problemi di
sicurezza, speditelo a <bugs@vim.org>, che arriva solo al manutentore di
Vim (ovvero Bram).
2. Aprite una richiesta su GitHub: https://github.com/vim/vim/issues

   Il testo verrà inoltrato alla mailing list vim-dev.

Siate concisi; tutto il tempo impiegato a rispondere viene sottratto al tempo dedicato a migliorare Vim! Fornite sempre un esempio riproducibile e provate a trovare quali impostazioni od altre cose contribuiscono al manifestarsi del bug.

Preferibilmente, richiamate Vim con: >
```
vim --clean -u reproduce.vim
```
Dove "reproduce.vim" è uno script che riproduce il problema.
Provatelo su macchine diverse, se lo ritenete opportuno (si tratta forse di un problema che si manifesta solamente in MS-Windows?).

Mandatemi delle patch se possibile!

Sarà utile includere informazioni sulla versione di Vim che state utilizzando e sulle vostre personalizzazioni. Potete ottenere l'informazione necessaria con questo comando: >
```
:so $VIMRUNTIME/bugreport.vim
```
Ciò creerà un file "bugreport.txt" nella directory corrente, con un sacco di informazioni sul vostro ambiente. Prima di spedirlo, verificate che non contenga informazioni confidenziali!

Se Vim va in crash, cercate per piacere di trovare in che punto. Potete trovare aiuto a questo riguardo in |**debug.txt**|.

Se avete dei dubbi o vi domandate se il problema sia già stato risolto, ma non sapete in che modo, registratevi nella maillist vim-dev e formulate la vostra domanda in quella sede. |**maillist**|

year-2000 *Y2K*

Poiché Vim internamente non usa le date per lavorare, non c'è il problema dell'anno 2000 di cui preoccuparsi. Vim impiega il tempo sotto forma di secondi a partire dal primo gennaio 1970. Viene usato per verificare il time-stamp del file aperto e del file di swap, non è critico e può soltanto generare dei messaggi di avvertimento.

Ci potrebbe essere un problema dell'anno 2038, quando i secondi non staranno più entro un intero a 32 bit. Ciò dipende dal compilatore, dalle librerie e dal sistema operativo. Specificamente, vengono usate le funzioni time_t e ctime(). E la time_t viene posta in quattro byte nel file di swap. Ma ciò si usa soltanto per la stampa della data e dell'ora del file per il ripristino, non riguarderà mai il lavoro normale.

La funzione di Vim strftime() usa direttamente la funzione di sistema strftime(). localtime() usa la funzione di sistema time(). getftime() usa l'ora che gli viene ritornata dalla funzione di sistema stat(). Se le vostre librerie di sistema sono conformi all'anno 2000, lo è anche Vim.

L'utente può creare script per Vim che usano comandi esterni. Questi potrebbero introdurre problemi Y2K, ma non si tratta veramente di problemi di Vim.

==
3. Ringraziamenti *credits* *author* *Bram* *Moolenaar*

La maggior parte di Vim è stata scritta da Bram Moolenaar <Bram@vim.org>.

Parti della documentazione provengono da molti manuali di Vi, scritti da:
        W.N. Joy
        Alan P.W. Hewett
        Mark Horton

L'editor Vim è basato su Stevie ed include (idee da) altro software, elaborato dalle persone citate qui. Altra gente ha dato aiuto spedendomi delle patch, suggerimenti e dando riscontro per quanto di buono o di cattivo c'è in Vim.

Vim non sarebbe diventato quello che è, senza l'aiuto di questa gente!

Ron Aaron	modifiche GUI per Win32
Mohsin Ahmed	cifratura
Zoltan Arpadffy	lavoro di porting a VMS

Tony Andrews	Stevie
Gert van Antwerpen	modifiche per DJGPP in MS-DOS
Berkeley DB(3)	idee per l'implementazione del file di swap
Keith Bostic	Nvi
Walter Briscoe	aggiornamenti del Makefile, varie patch
Ralf Brown	libreria SPAWNO per MS-DOS
Robert Colon	molte annotazioni utili
Marcin Dalecki	porting a GUI GTK+, icone toolbar, gettext()
Kayhan Demirel	mi ha spedito notizie in Uganda
Chris & John Downey	xvi (idee per la versione multi-finestra)
Henk Elbers	primo porting a VMS
Daniel Elstner	porting a GTK+ 2
Eric Fischer	porting a Mac, 'cindent', ed altre migliorie
Benji Fisher	risposte a molte domande degli utenti
Bill Foster	porting della GUI di Athena
Google	lascia a Bram un giorno alla settimana per Vim
Loic Grenie	xvim (idee per la versione multi-finestra)
Sven Guckes	promotore di Vim ed ex-manutentore sito Vim
Darren Hiebert	Exuberant ctags
Jason Hildebrand	porting a GTK+ 2
Bruce Hunsaker	migliorie al porting a VMS
Andy Kahn	supporto per Cscope, porting GUI GTK+
Oezguer Kesim	manutentore Mailing List di Vim
Axel Kielhorn	lavora sul porting a Macintosh
Steve Kirkendall	Elvis
Roger Knobbe	porting originale a Windows NT
Sergey Laskavy	aiuto a Vim da Mosca
Felix von Leitner	precedente manutentore Mailing List di Vim
David Leonard	porting estensioni Python ad Unix
Avner Lottem	modificare testi scritti da destra a sinistra
Flemming Madsen	X11 client-server, varie funzioni e patch
Tony Mechelynck	risposta a molte domande poste da utenti
Paul Moore	estensione interfaccia Python, molte patch
Katsuhito Nagano	lavoro sulle versioni multi-byte
Sung-Hyun Nam	lavoro sulle versioni multi-byte
Vince Negri	GUI Win32 GUI e migliorie versione console
Steve Oualline	autore del primo libro su Vim |**frombook**|
Dominique Pelle	controlli con valgrind e molte patch
A.Politz	molti rapporti di bug e alcune patch
George V. Reilly	porting Win32, GUI Win32 iniziale
Stephen Riehm	collezionista di bug
Stefan Roemer	varie patch e supporto agli utenti
Ralf Schandl	porting IBM OS/390
Olaf Seibert	versioni DICE e BeBox, migliorie a regexp
Mortaza Shiran	patch Farsi (persiano)
Peter da Silva	termlib
Paul Slootman	porting a OS/2
Henry Spencer	espressioni regolari
Dany St-Amant	porting a Macintosh
Tim Thompson	Stevie
G. R. (Fred) Walter	Stevie
Sven Verdoolaege	interfaccia Perl
Robert Webb	completamento linea comando, versioni GUI, e molte patch
Ingo Wilken	interfaccia Tcl
Mike Williams	stampa in PostScript
Juergen Weigert	versione lattice, migliorie AUX, porting a UNIX e a MS-DOS, autoconf
Stefan 'Sec' Zehl	manutentore di vim.org
Yasuhiro Matsumoto	molti miglioramenti in ambiente MS-Windows
Ken Takata	correzioni e funzionalità
Kazunobu Kuriyama	GTK 3
Christian Brabandt	molte correzioni, funzionalità, supporto utenti, etc.

Voglio ringraziare tutta la gente che mi ha spedito rapporti sui bug e suggerimenti. La lista è troppo lunga per citarli tutti qui. Vim non avrebbe potuto essere lo stesso senza le idee di tutta questa gente: Mantengono in vita Vim!
love *peace* *friendship* *gross-national-happiness*

In questi documenti ci sono molti riferimenti alle altre versioni di Vi:

Vi *vi*

Vi "l'originale". Senza ulteriori annotazioni questa è la versione di Vi apparsa in Sun OS 4.x. ":version" restituisce "Version 3.7, 6/7/85". Talvolta altre versioni vi fanno riferimento. Gira soltanto sotto Unix. Il codice sorgente è disponibile soltanto con una licenza. Più informazioni su Vi possono venir trovate tramite:
http://vi-editor.org [attualmente non utilizzabile...]

Posix

Posix Dallo standard IEEE 1003.2, Parte 2: Shell e programmi di utilità. Generalmente conosciuto come "Posix". Descrizione testuale di come Vi dovrebbe funzionare.
Vedere |**posix-compliance**|.

Nvi

Nvi Il "Nuovo" Vi. La versione di Vi che nasce con BSD 4.4 e FreeBSD. Ottima compatibilità con il Vi originale, con alcune estensioni. La versione usata è la 1.79. ":version" risponde "Version 1.79 (10/23/96)". Non ci sono stati rilasci negli ultimi anni, sebbene la versione 1.81 sia in fase di sviluppo.
Il codice sorgente è disponibile liberamente.

Elvis

Elvis Un altro clone di Vi, scritto da Steve Kirkendall. Assai compatto ma non così flessibile come Vim.
La versione usata è 2.1. È ancora in sviluppo. Il codice sorgente è disponibile liberamente.

==========

4. Notazione *notation*

Quando l'evidenziazione della sintassi viene usata per leggerlo, quel testo che non sia stato digitato letteralmente viene spesso evidenziato con il gruppo Speciale. Questi sono elementi racchiusi fra [], {} e <>, e CTRL-X.

Nota Vim usa tutti i caratteri possibili nei comandi. Talvolta le [], {} e <> sono parte di quel che immettete, il contesto dovrebbe renderlo chiaro.

[] I caratteri tra parentesi quadre sono facoltativi.

count *[count]*

[count] Un numero può venire posto davanti ad un comando per moltiplicarlo od iterarlo. Se non vi fosse alcun numero, viene usato l'uno, se non specificato diversamente. Si noti che in questo manuale il [count] non viene menzionato nella descrizione del comando, ma soltanto nella sua spiegazione. Ciò è stato fatto per rendere più semplice l'osservazione dei comandi. Se l'opzione 'showcmd' è abilitata, il numero (parzialmente) inserito vien mostrato in fondo alla finestra. Potete usare <Del> per cancellare l'ultima cifra (|**N<Del>**|).

[quotex]

["x] Specifica facoltativa di un registro nel quale il testo possa essere immagazzinato.
Vedere |**registers**|. La x è un solo carattere tra 'a' e 'z' o 'A' e 'Z' o '"', ed in alcuni casi (con il comando put) tra '0' e '9', '%', '#', od altri. Le lettere maiuscole o minuscole indicano lo stesso registro, ma le minuscole vengono usate per sovrascrivere i contenuti precedenti del registro, mentre le maiuscole vengono usate per accodare ai contenuti precedenti del registro. Senza la ""x" o con """" il testo immagazzinato viene messo entro il registro senza nome.

{}

{} Le parentesi graffe denotano parti del comando che vanno specificate, ma che possono prendere molti valori diversi. Le differenze tra Vim e Vi vengono pure date entro parentesi graffe (ciò sarà evidente dal contesto).

{char1-char2}

{char1-char2} Un solo carattere compreso tra char1 e char2. Ad esempio: {a-z} è una lettera minuscola. Molti campi possono essere concatenati. Ad esempio, {a-zA-Z0-9} corrisponde a qualsiasi carattere alfanumerico.

{motion} *movement*

{motion} Un comando che muove il cursore. Ciò viene spiegato in |**motion.txt**|. Esempi:

w	all'inizio della prossima parola
b	all'inizio della parola corrente
4j	quattro linee sotto
/The<CR>	alla prossima occorrenza di "The"

Questo viene impiegato dopo un comando |**operator**| per muoversi oltre il testo su cui si era operato.
- Se il movimento include un count e l'operatore ha a sua volta un count, i due count verrebbero moltiplicati. Ad esempio: "2d3w" cancella sei parole.
- Il movimento può essere all'indietro, ad es. "db" per cancellare dall'inizio della parola corrente.
- il movimento può anche essere un clic di mouse. Il mouse però non viene supportato su tutti i terminali.
- Il comando ":omap" può venire usato per la mappatura di caratteri sino a quando si deve ancora inserire un operatore.
- I comandi Ex possono venire usati per spostare il cursore. Ciò può servire per chiamare una funzione che svolga qualche movimento complicato.
  Il movimento è sempre esclusivamente di caratteri, qualsiasi comando ":" venga adoperato. Ciò significa che è impossibile includere l'ultimo carattere di una linea escludendo il carattere di fine linea (a meno che 'virtualedit' non sia impostato).
  Se il comando Ex modifica il testo prima del momento in cui l'operatore inizi (o passi ad) un altro buffer il risultato non è prevedibile. È possibile modificare il testo immediatamente successivo. Il passaggio ad un altro buffer è possibile se il buffer corrente non viene scaricato.

{Visual}

{Visual} Un'area di testo selezionata. Si avvia coi comandi "v", "V", o CTRL-V, poi ogni movimento del cursore può venire usato per modificare la fine del testo selezionato.
Viene usato prima di un comando |**operator**| per evidenziare il testo su cui si intende operare.
Vedere |**Visual-mode**|.

<character>

<character> Un carattere speciale dalla tabella qui sotto, anche con modificatori, od unico carattere ASCII con modificatori.

'character'

'c' Un solo carattere ASCII.

CTRL-{char}

CTRL-{char} {char} digitato come carattere di controllo; cioè, digitando {char} mentre si tiene premuto il tasto CTRL. È ininfluente che {char} sia maiuscolo o minuscolo; quindi CTRL-A e CTRL-a sono equivalenti. Ma su certi terminali, l'uso del tasto SHIFT genera un altro codice, in questo caso non va usato.

'option'

'option' Un'opzione, o parametro, che può venire impostato ad un valore, è compreso tra virgolette singole. Vedere |**options**|.

quotecommandquote

"command" Il riferimento ad un comando che potete digitare è racchiuso fra virgolette doppie.
`command` Nuovo stile per indicare un comando, distinguendolo da un testo virgolettato o da una stringa di caratteri.

key-notation *key-codes* *keycodes*

Questi nomi per i tasti vengono usati nella documentazione. Possono anche venire usati con il comando ":map" (inserite il nome del tasto premendo CTRL-K e poi il tasto che corrisponde a quel nome).

notazione	significato	equivalente	valore/i decimali ~
<Nul>	zero	CTRL-@	0 (inserito come 10) *<Nul>*

```
<BS>            backspace                CTRL-H    8     *backspace*
<Tab>           tab                      CTRL-I    9     *tab* *Tab*
                                                         *linefeed*
<NL>            linefeed [a capo]        CTRL-J   10 (usato per <Nul>)
<FF>            formfeed [pagina nuova]  CTRL-L   12     *formfeed*
<CR>            ritorno carrello         CTRL-M   13     *carriage-return*
<Return>        come <CR>                                *<Return>*
<Enter>         come <CR>                                *<Enter>*
<Esc>           escape                   CTRL-[   27     *escape* *<Esc>*
<Space>         spazio                            32     *space*
<lt>            minore                   <        60     *<lt>*
<Bslash>        backslash                \        92     *backslash* *<Bslash>*
<Bar>           barra verticale          |       124     *<Bar>*
<Del>           cancella                        127
<CSI>           Command Sequence Intro   ALT-Esc 155     *<CSI>*
<xCSI>          CSI se introdotta in GUI                 *<xCSI>*

<EOL>           fine-linea (può essere <CR>, <LF> o <CR><LF>,
                dipende dal sistema e da 'fileformat')  *<EOL>*

<Up>            cursore-su                          *cursor-up* *cursor_up*
<Down>          cursore-giù                         *cursor-down* *cursor_down*
<Left>          cursore-sinistra                    *cursor-left* *cursor_left*
<Right>         cursore-destra                      *cursor-right* *cursor_right*
<S-Up>          shift-cursore-su
<S-Down>        shift-cursore-giù
<S-Left>        shift-cursore-sinistra
<S-Right>       shift-cursore-destra
<C-Left>        control-cursore-sinistra
<C-Right>       control-cursore-destra
<F1> - <F12>    tasti funzione da 1 a 12            *function_key* *function-key*
<S-F1> - <S-F12> shift-tasti funzione da 1 a 12     *<S-F1>*
<Help>          tasto help
<Undo>          tasto undo
<Insert>        tasto insert
<Home>          home                                *home*
<End>           end                                 *end*
<PageUp>        pagina-su                           *page_up* *page-up*
<PageDown>      pagina-giù                          *page_down* *page-down*
<kHome>         keypad home (alto a sinistra)       *keypad-home*
<kEnd>          keypad end (basso a destra)         *keypad-end*
<kPageUp>       keypad pag.-su (alto a sinistra)    *keypad-page-up*
<kPageDown>     keypad pag.-giù (basso a destra)    *keypad-page-down*
<kPlus>         keypad +                            *keypad-plus*
<kMinus>        keypad -                            *keypad-minus*
<kMultiply>     keypad *                            *keypad-multiply*
<kDivide>       keypad /                            *keypad-divide*
<kEnter>        keypad Invio                        *keypad-enter*
<kPoint>        keypad punto decimale               *keypad-point*
<k0> - <k9>     keypad 0 a 9                        *keypad-0* *keypad-9*
<S-...>         tasto maiuscolo (shift)             *shift* *<S-*
<C-...>         tasto control                       *control* *ctrl* *<C-*
<M-...>         tasto alt o meta                    *meta* *alt* *<M-*
<A-...>         come <M-...>                        *<A-*
<D-...>         tasto comando (solo Macintosh)      *<D-*
<t_xx>          tasto designato come "xx" in termcap
------------------------------------------------------------------------------
```

Nota: I tasti di cursore maiuscoli, il tasto di help, ed il tasto undo sono disponibili soltanto su alcuni terminali. Sull'Amiga, il tasto di funzione 10 produce un codice (CSI) che può venire usato per le sequenze di tasti. Verrà riconosciuto soltanto dopo avere premuto un altro tasto.

Nota: Ci sono due codici per il tasto Cancella [delete]. 127 è il valore ASCII decimale per il tasto Cancella, che viene sempre riconosciuto. Alcuni tasti Cancella inviano un codice diverso, nel qual caso questo valore è ottenuto dalla linea in termcap "kD". Entrambi i valori hanno lo stesso effetto. Vedere anche |**:fixdel**|.

Nota: I tasti del tastierino ["keypad" nella tabella sopra - NdT] vengono usati nello stesso modo dei corrispondenti tasti "normali". Ad esempio, <kHome> ha lo stesso effetto che ha <Home>. Se un tasto del tastierino invia lo stesso codice tasto di riga come l'equivalente della tastiera "normale",

verrà riconosciuto come il codice di tale tastiera. Ad esempio, quando <kHome> invia lo stesso codice di <Home>, quando viene premuto <kHome> Vim penserà che <Home> sia stato premuto. La mappatura di <kHome> in questo caso non verrà attivata.

<>
Gli esempi vengono sovente dati nella notazione <>. Talvolta questo serve solo per chiarire cosa occorre digitare, ma spesso il tasto in questione può venire inserito scrivendolo proprio così, ad esempio, con il comando ":map". Le regole sono:

1. Ogni carattere stampabile viene digitato direttamente, eccetto '\' e '<'
2. Un backslash viene rappresentato con "\\", doppia backslash, o con "<Bslash>".
3. Un vero '<' viene rappresentato con "\<" o "<lt>". Quando non è possibile una ambiguità, può venire usato direttamente un '<'.
4. "<key>" significa il tasto speciale digitato. Questa è la notazione spiegata nella tabella più sopra. Qualche esempio:

<Esc>	Tasto Escape
<C-G>	CTRL-G
<Up>	Tasto cursore su
<C-LeftMouse>	Control-clic del tasto sinistro del mouse
<S-F11>	Tasto di funzione 11, maiuscolo
<M-a>	Meta- a ('a' con impostato il bit 8)
<M-A>	Meta- A ('A' con impostato il bit 8)
<t_kd>	Tasto "kd" di termcap (tasto cursore giù)

Volendo usare la notazione piena <> in Vim, dovete essere sicuri che la flag '<' sia esclusa da 'cpoptions' (quando 'compatible' non è impostato, lo è sempre di default). >
```
:set cpo-=<
```
La notazione <> usa <lt> per aggirare il significato speciale dei nomi dei tasti. Usare un backslash funziona pure, ma solo quando 'cpoptions' non include la flag 'B'.

Esempi per mappare CTRL-H alla parola di sei caratteri "<Home>": >
```
:imap <C-H> \<Home>
:imap <C-H> <lt>Home>
```
La prima funziona solo se la flag 'B' non è nelle 'cpoptions'. La seconda funziona sempre.
Per mappare la stringa letterale di quattro caratteri "<lt>": >
```
:map <C-L> <lt>lt>
```

Per le mappature, le abbreviazioni ed i comandi del menù potete copiare ed incollare gli esempi ed usarli direttamente. O scriverli a mano, includendo i caratteri '<' e '>'. Ciò NON funzionerebbe per altri comandi come ":set" e ":autocmd"!

==========
5. Modalità, introduzione *vim-modes-intro* *vim-modes*

Vim ha sette modi DI BASE:

Normal *Normal-mode* *command-mode*

Normal mode	In Normal mode potete digitare tutti i comandi del normale editor. Se avviate l'editor vi trovate in questo modo (se non avete impostato l'opzione 'insertmode', vedere sotto). Questo è noto anche come Command mode.
Visual mode	È come il Normal mode, ma i comandi di movimento determinano l'estensione dell'area evidenziata. Quando viene usato un comando non di movimento, viene eseguito per l'area evidenziata. Vedere |**Visual-mode**|. Se l'opzione 'showmode' è attiva il messaggio "-- VISUALE --" sarà visibile in fondo a sinistra nella finestra.
Select mode	Assomiglia molto al modo selezione di MS-Windows. L'immissione di un carattere stampabile termina la selezione e fa passare ad Insert mode. Vedere |**Select-mode**|. Se l'opzione 'showmode' è attiva il messaggio "-- SELEZIONA --" sarà visibile in fondo a sinistra

nella finestra.

Insert mode — In Insert mode il testo che scrivete viene inserito nel buffer. Vedere |**Insert-mode**|.
Se l'opzione 'showmode' è attiva il messaggio "-- INSERISCI --" sarà visibile in fondo a sinistra nella finestra.

Command-line mode
Cmdline mode — In Command-line mode (chiamato anche Cmdline mode) potete inserire una linea di testo in fondo alla finestra. Ciò vale per i comandi Ex, ":", i comandi di ricerca di un'espressione, "?" e "/", ed il comando di filtro, "!". |**Cmdline-mode**|

Ex mode — Simile al Command-line mode, ma dopo avere digitato un comando rimanete in Ex mode. Poche possibilità di modifica della linea di comando. |**Ex-mode**|

Terminal-Job mode — Interazione con un job in una finestra di terminale. keys go to the job and the job output is displayed in
I caratteri immessi sono passati al job, e l'output del job è visualizzato nella finestra di terminale.
Vedere |**terminal**| per come passare ad altri modi.

Ci sono sette modi ADDIZIONALI. Sono varianti dei modi BASE:

Operator-pending *Operator-pending-mode*

Operator-pending mode — È come il Normal mode, ma dopo avere avviato un comando di operatore, e Vim attende un {movimento} per specificare il testo su cui vuole lavorare l'operatore.

Replace mode — Replace mode è un caso speciale dell'Insert mode. Potete fare le stesse cose come in Insert mode, ma per ogni carattere che digitare, un carattere del testo esistente viene cancellato. Vedere |**Replace-mode**|.
Se l'opzione 'showmode' è attiva il messaggio "-- SOSTITUISCI --" sarà visibile in fondo a sinistra nella finestra.

Insert Normal mode — Attivato quando CTRL-O viene dato in Insert mode. E' come Normal mode, ma dopo avere eseguito un solo comando Vim ritorna nell'Insert mode.
Se l'opzione 'showmode' è attiva il messaggio "-- (inserisci) --" sarà visibile in fondo a sinistra nella finestra.

Terminal Normal mode — Utilizzo del Normal mode in una finestra di terminale. Non è possibile effettuare modifiche. Usare un comando di inserimento, come "a" oppure "i" per ritornare in Terminal-Job mode.

Insert Visual mode — Attivato quando si inizia una selezione Visual da Insert mode, ad esempio, usando CTRL-O e poi "v", "V" o CTRL-V. Quando termina la selezione Visual, Vim ritorna all'Insert mode.
Se l'opzione 'showmode' è attiva il messaggio "-- (inserisci) VISUALE --" sarà visibile in fondo a sinistra nella finestra.

Virtual Replace mode — Virtual Replace mode è simile a Replace mode, ma invece di caratteri nel file state sostituendo aree dello schermo. Vedere |**Virtual-Replace-mode**|.
Se l'opzione 'showmode' è attiva il messaggio "-- V-SOSTITUISCI --" sarà visibile in fondo a sinistra nella finestra.

Insert Select mode — Attivato quando si inizia Select mode da Insert mode. Ad esempio, trascinando il mouse o <S-Right>.
Quando termina il Select mode, Vim ritorna in Insert mode.
Se l'opzione 'showmode' è attiva il messaggio "-- (inserisci) SELEZIONA --" sarà visibile in fondo a

sinistra nella finestra.

==========

6. Cambiare modalità *mode-switching*

Se per qualunque ragione non sapeste in quale modo vi troviate, potete sempre ritornare al Normal mode digitando <Esc> due volte. Questo non funziona in Ex mode, dove potete usare ":visual".
Saprete di essere ritornati in Normal mode vedendo lo schermo lampeggiare od udendo la campana suonare dopo aver digitato <Esc>. Comunque, quando premete <Esc> dopo avere usato CTRL-O in Insert mode otterrete un beep ma sarete sempre in Insert mode, dovete digitare ancora <Esc>.

i_esc

Dal modo \ AL modo	Normal	Visual	Select	Insert	Replace	Cmd-line	Ex
Normal		v V ^V	*4	*1	R gR	: / ? !	Q
Visual	*2		^G	c C	--	:	--
Select	*5	^O ^G		*6	--	--	--
Insert	<Esc>	--	--		<Insert>	--	--
Replace	<Esc>	--	--	<Insert>		--	--
Command-line	*3	--	--	:start	--		--
Ex	:vi	--	--	--	--	--	

-- non possibile

*1 Passate dal Normal mode all'Insert mode dando il comando "i", "I", "a", "A", "o", "O", "c", "C", "s" o S".
*2 Passate dal Visual mode al Normal mode dando un comando non di movimento, che causa l'esecuzione del comando, o schiacciando <Esc> "v", "V" o "CTRL-V" (vedere |**v_v**|), che arrestano semplicemente il Visual mode senza altri effetti.
*3 Passate da Command-line mode al Normal mode così:
- Premete <CR> o <NL>, ciò causa l'esecuzione del comando inserito.
- Cancellate l'intera linea (ad esempio, con CTRL-U) e dando un <BS> finale.
- Premete CTRL-C o <Esc>, uscendo da Command-line mode senza eseguire il comando.

Nell'ultimo caso <Esc> può essere il carattere definito con l'opzione 'wildchar', nel qual caso partirà il completamento della linea di comando. Potete ignorarlo e digitare ancora <Esc>. {Vi: quando si preme <Esc> viene eseguita la linea di comando. Ciò è una sorpresa per molti; perciò questo è stato cambiato in Vim. Ma quando <Esc> è parte di una mappatura, la linea di comando viene eseguita. Se volete il comportamento di Vi anche quando digitate <Esc>, usate ":cmap ^V<Esc> ^V^M"}
*4 Passate dal Normal al Select mode così:
- usate il mouse per selezionare del testo quando 'selectmode' contiene "mouse"
- usate un comando non stampabile per spostare il cursore tenendo premuto il tasto Shift, quando l'opzione 'selectmode' contiene "key"
- usate "v", "V" o "CTRL-V" quanto 'selectmode' contiene "cmd"
- usate "gh", "gH" o "g CTRL-H" |**g_CTRL-H**|

*5 Passate dal Select mode al Normal mode usando un comando non stampabile per spostare il cursore, senza tenere premuto il tasto Shift.
*6 Passate dal Select mode all'Insert mode digitando un carattere stampabile. La selezione viene cancellata ed il carattere inserito.

Se l'opzione 'insertmode' è impostata, la modifica di un file partirà in Insert mode.

CTRL-_CTRL-N *i_CTRL-_CTRL-N* *c_CTRL-_CTRL-N* *v_CTRL-_CTRL-N*
Inoltre il comando CTRL-\ CTRL-N o <C-\><C-N> può venire usato per passare al Normal mode da ogni altro modo. Ciò può venire usato per essere certi che Vim sia in Normal mode, senza causare un beep come farebbe <Esc>. Comunque, ciò non funziona in Ex mode. Quando usato dopo un comando che richiede un argomento, come ad es. |**f**| o |**m**|, si applica il ritardo consentito da 'ttimeoutlen'.
Quando il cursore si trova in una finestra di terminale, CTRL-\ CTRL-N porta in Normal mode per un solo comando, vedere |**t_CTRL-_CTRL-N**|.

CTRL-_CTRL-G *i_CTRL-_CTRL-G* *c_CTRL-_CTRL-G* *v_CTRL-_CTRL-G*
Il comando CTRL-\ CTRL-G o <C-\><C-G> può venire usato per andare in Insert mode quando sia impostato 'insertmode'. Altrimenti va in Normal mode. Ciò

può venire usato per accertarsi che Vim si trovi nel modo indicato da 'insertmode', senza sapere in che modo si trovi attualmente Vim.

Q *mode-Ex* *Ex-mode* *Ex* *EX* *E501*

Q Passa a "Ex" mode. È un po' come digitare dei comandi ":" uno dopo l'altro, eccetto:
- Non dovete continuare a scrivere ":".
- Lo schermo non viene aggiornato dopo ciascun comando.
- Non è disponibile la normale modifica della linea di comando.
- Non vengono usate mappature ed abbreviazioni.

Effettivamente, state modificando le linee con i comandi di modifica "standard" da linea di comando (<Del> o <BS> per cancellare, CTRL-U per eliminare l'intera linea).
Vim entrerà in questo modo per default se viene invocato come "ex" dalla linea di comando.
Usate il comando ":vi" |**:visual**| per uscire dall'"Ex" mode.
Nota: Nelle versioni più vecchie di Vim "Q" formattava il testo, cosa che ora viene fatta con |**gq**|. Ma se usate lo script |**vimrc_example.vim**| "Q" funzionerà come "gq".

gQ

gQ Passa a '"Ex" mode, come con "Q", ma si comporta come se aveste digitato dei comandi ":" l'uno dopo l'altro. Tutte le modifiche da linea di comando, il completamento etc. sono disponibili.
Usate il comando ":vi" |**:visual**| per uscire dall'"Ex" mode.
{non in Vi}

==========

7. Contenuti della finestra *window-contents*

In Normal mode e nell'Insert/Replace mode la finestra dello schermo mostra l'attuale contenuto del buffer: What You See Is What You Get [WYSIWYG - Quel che vedete è quel che avete - NdT]. Ci sono due eccezioni:
- Quando l'opzione 'cpoptions' contiene '$', e la modifica è all'interno di una sola linea, il testo non viene direttamente cancellato, ma un '$' viene messo sull'ultimo carattere cancellato.
- Inserendo del testo entro una sola finestra, altre finestre contenenti lo stesso testo non vengono aggiornate sino a che l'immissione non sia terminata.

{Vi: Lo schermo non viene sempre aggiornato su terminali lenti}

Linee più lunghe della larghezza della finestra andranno a capo, a meno che l'opzione 'wrap' non sia disattivata (vedere sotto). L'opzione 'linebreak' può venire impostata per andare a capo in corrispondenza di un carattere spazio.

Se la finestra ha spazio dopo l'ultima linea del buffer, Vim mostrerà '~' nella prima colonna della linea successiva nella finestra, così:

```
+-----------------------+
|qualche linea          |
|ultima linea           |
|~                      |
|~                      |
+-----------------------+
```

Così le linee con '~' indicano che è stata raggiunta la fine del buffer.

Se l'ultima linea in una finestra non può essere visualizzata completamente, Vim lo indicherà con una '@' nella prima colonna dell'ultima linea nella finestra, così: >

```
+-----------------------+
|prima linea            |
|seconda linea          |
|@                      |
|@                      |
```

```
        +----------------------+
```

Così le linee '@' indicano che c'è una linea che non è possibile visualizzare completamente nella finestra.

Quando è presente la flag "lastline" entro l'opzione 'display', non vedrete i caratteri '@' sul lato sinistro della finestra. Se l'ultima linea non può essere visualizzata completamente, solo la parte che si riesce a visualizzare verrà mostrata, e gli ultimi tre caratteri dell'ultima linea verranno sostituiti con "@@@", così:

```
        +----------------------+
        |prima linea           |
        |seconda linea         |
        |linea molto lunga che n|
        |on ci sta nella fine@@@|
        +----------------------+
```

Se ci fosse una sola linea troppo lunga per andare bene nella finestra, questa è una situazione speciale. Vim mostrerà solo una parte della linea, attorno alla quale c'è il cursore. Non vengono mostrati caratteri speciali, così potete modificare tutte le parti di questa linea.
{Vi: dà un "internal error" per linee non visualizzabili in una finestra}

L'inserimento di '@' nell'opzione 'highlight' può venire usata per impostare una evidenziazione speciale per i caratteri '@' e '~'. Ciò rende possibile distinguerli dai caratteri reali contenuti nel buffer.

L'opzione 'showbreak' contiene la stringa da mettere davanti alle linee solo parzialmente visualizzate nello schermo.

wrap-off
Se l'opzione 'wrap' è disabilitata, le linee lunghe non andranno a capo. Soltanto la parte che riempie lo schermo verrà mostrata. Se il cursore viene mosso sino ad una parte della linea che non viene mostrata, lo schermo scorrerà in orizzontale. Il vantaggio di questo metodo è che le colonne vengono mostrate così come sono e le linee che non possono essere interamente contenute sullo schermo possono venire modificate. Lo svantaggio è che non potete vedere tutti i caratteri di una linea in una volta sola. L'opzione 'sidescroll' può venire impostata al minimo numero di colonne da far scorrere.
{Vi: non prevede l'opzione 'wrap'}

Tutti i caratteri normali ASCII vengono mostrati direttamente sullo schermo. Il <Tab> viene sostituito dal numero di spazi che rappresenta. Altri caratteri non stampabili vengono sostituiti con "^{char}", dove {char} è il carattere non stampabile con aggiunto 64. Così il carattere 7 (campanello) verrà mostrato come "^G".
Caratteri tra 127 e 160 vengono sostituiti con "~{char}", dove {char} è il carattere con sottratto 64. Questi caratteri occupano più di una posizione sullo schermo. Il cursore può venire posizionato solo sulla prima delle due.

Se impostate l'opzione 'number', tutte le linee verranno precedute dal loro numero. Suggerimento: Se non vi piace che le linee che vanno a capo si mescolino con i numeri di linea, impostate l'opzione 'showbreak' su otto spazi:

```
        ":set showbreak=\ \ \ \ \ \ \ "
```

Se avete impostato l'opzione 'list', il carattere <Tab> non verrà mostrato come molti spazi, ma come "^I". Un '$' verrà posto alla fine della linea, così da visualizzare gli spazi posti alla fine della riga.

In Command-line mode solo la linea comandi viene mostrata correttamente. La vista dei contenuti del buffer viene aggiornata quando tornate in Command mode.

L'ultima linea della finestra viene usata per visualizzare lo status ed altri messaggi. I messaggi di status verranno usati solo se un'opzione è impostata:

messaggio status	opzione	default	default Unix~
modo corrente	'showmode'	on	on
caratteri di comando	'showcmd'	on	off
posizione del cursore	'ruler'	off	off

Il modo corrente è "-- INSERISCI --" o "-- SOSTITUISCI --", vedere

|'showmode'|. I caratteri di comando sono quelli che avete digitato ma non sono ancora stati usati. {Vi: non mostra i caratteri che avete digitato o la posizione del cursore}

Se avete un terminale lento potete eliminare i messaggi di status per velocizzare l'editing:

```
:set nosc noru nosm
```

Se ci fosse un errore, verrà mostrato un messaggio di errore per almeno un secondo (in video invertito). {Vi: i messaggi di errore possono venire sovrascritti con altri messaggi prima che riusciate a leggerli}

Alcuni comandi mostrano quante linee sono state interessate. Sopra quale soglia ciò deve avvenire può venire controllato con l'opzione 'report' (default 2).

Sull'Amiga Vim girerà entro una finestra CLI. Il nome Vim e l'intero nome del file corrente verranno mostrate nella barra del titolo. Quando la finestra viene ridimensionata, Vim ridisegnerà automaticamente la finestra. Potete fare la finestra piccola come vi piace, ma se diventa troppo piccola neppure una sola linea potrebbe essere contenuta interamente. Fatele almeno larghe quaranta caratteri per poter leggere la maggior parte dei messaggi sull'ultima linea.

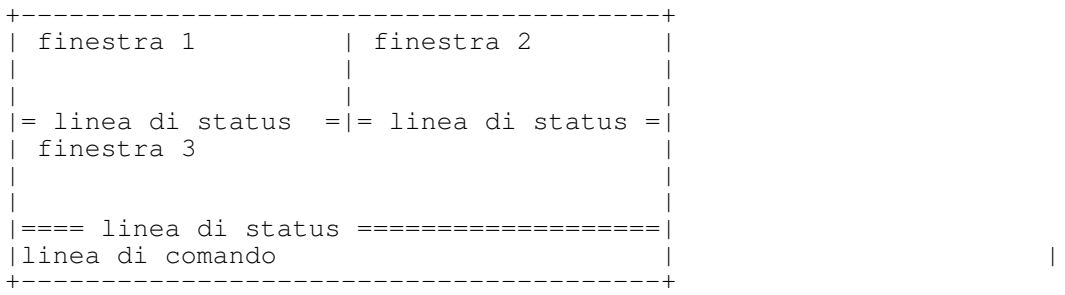
Sulla maggior parte dei sistemi Unix, il ridimensionamento della finestra viene riconosciuto e gestito correttamente da Vim. {Vi: non ok}

==

8. Definizioni *definitions*

buffer	Contiene righe di testo, di solito lette da un file.
schermo	Tutta l'area che Vim usa per lavorarci. Può essere una finestra di emulatore di terminale. Anche chiamata "la finestra Vim".
finestra	Una vista su di un buffer. Ci possono essere più finestre per un unico buffer.

Uno schermo contiene una o più finestre, separate da linee di status e con la linea dei comandi in basso.



La linea di comando viene anche usata per i messaggi. Scorre verso l'alto dello schermo quando non c'è abbastanza spazio nella linea comandi stessa.

Ci sono quattro differenti tipi di linee:

linee del buffer	Le linee nel buffer. Sono le stesse linee che vengono lette o scritte entro un file. Possono essere lunghe migliaia di caratteri.
linee logiche	Le linee del buffer con la piegatura applicata. Le linee del buffer in una piegatura chiusa divengono una sola linea logica: "+-- 99 linee piegate". Possono essercene lunghe migliaia di caratteri.
linee della finestra	Le linee mostrate entro una finestra: Un insieme di linee logiche con l'andata a capo, interruzioni di linea, etc. applicate. Possono essere lunghe soltanto quanto la larghezza della finestra consente, linee più lunghe verranno visualizzate su più linee della finestra o troncate.
linee dello schermo	Le linee dello schermo che Vim usa. Consistono delle linee della finestra di tutte le finestre, con aggiunte le linee di status e quella dei comandi.

Possono essere lunghe soltanto quanto la larghezza
dello schermo consente. Se la linea di comando
diviene più lunga continua sulla linea seguente e le
linee più sopra vengono fatte scorrere per fare
spazio.

```
linee del buffer linee logiche  linee d. fines. linee dello schermo  ~

1. uno           1. uno         1. +-- piega  1.  +-- piegate
2. due           2. +-- piegate 2. cinque     2.  cinque
3. tre           3. cinque      3. sei        3.  sei
4. quattro       4. sei         4. sette      4.  sette
5. cinque        5. sette                     5.  === linea status ===
6. sei                                        6.  aaa
7. sette                                      7.  bbb
                                              8.  ccc ccc c
1. aaa           1. aaa         1. aaa        9.  cc
2. bbb           2. bbb         2. bbb        10. ddd
3. ccc ccc ccc   3. ccc ccc ccc 3. ccc ccc c  11. ~
4. ddd           4. ddd         4. cc         12. === linea status ===
                                5. ddd        13. (linea comandi)
                                6. ~
```

==========

vim:tw=78:ts=8:ft=help:norl:

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