usr_02.txt Per **Vim version 8.0.** Ultima modifica: 2016 Mar 14

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

I primi passi con Vim

Questo capitolo fornisce qualche sommaria informazione per scrivere un file con Vim. Non bene o alla svelta, ma potete scrivere. Dedicate un po' di tempo ad impratichirvi con questi comandi, essi sono la base per quanto segue.

Capitolo seguente: |**usr_03.txt**| Muoversi nel file
Capitolo precedente: |**usr_01.txt**| Sui manuali
Indice: |**usr_toc.txt**|

==========

02.1 Avviare Vim la prima volta

Per avviare Vim, usate questo comando: >

```
gvim file.txt
```

In UNIX potete scriverlo in ogni prompt di comando. Se usaste Microsoft Windows, aprite una finestra MS-DOS e digitate il comando.
In ogni caso, Vim viene avviato aprendo un file chiamato file.txt. Se questo fosse un file nuovo otterreste una finestra vuota. La vostra schermata apparirebbe così:

```
+----------------------------------------+
|#                                       |
|~                                       |
|~                                       |
|~                                       |
|~                                       |
|"file.txt" [File nuovo]                 |
+----------------------------------------+
        ('#" è la posizione del cursore.)
```

Le linee che iniziano con una tilde (~) indicano di non far parte del file.
In altre parole, quando Vim va oltre la fine del file mostra le linee con la tilde. In fondo allo schermo una linea di messaggio informa che il file si chiama file.txt ed indica che voi state creando un file nuovo. Il messaggio informativo è temporaneo ed informazioni successive lo sovrascrivono.

IL COMANDO VIM

Il comando gvim fa sì che l'editor apra una nuova finestra in cui scrivere.

Se invece usate questo comando: >

```
vim file.txt
```

lavorerete entro la vostra finestra di comando. In altre parole, se lavorate entro un xterm, l'editor impiegherà la vostra finestra di xterm. Se state usando una finestra di comando MS-DOS sotto Microsoft Windows, lavorerete entro questa finestra. Il testo entro la finestra sarà identico in entrambe le versioni, ma con gvim vi sono altre e maggiori possibilità, come una barra di menù. Maggiori informazioni più avanti.

==========

02.2 Inserire del testo

L'editor Vim è un editor modale. Ciò significa che si comporterà diversamente a seconda del modo in cui vi trovate. I due modi principali si chiamano Normal mode ed Insert mode. In Normal mode i caratteri che scrivete sono comandi. In Insert mode gli stessi caratteri vengono inseriti come testo.
Vim viene avviato in Normal mode. Per passare all'Insert mode inserite il comando "i" (i sta per Insert). Poi potrete scrivere del testo. Esso verrà inserito entro il file. Non preoccupatevi di aver commesso degli errori; potrete correggerli dopo. Per scrivere la seguente canzoncina del programmatore, dovete digitare quanto segue: >

```
iA very intelligent turtle
Found programming UNIX a hurdle
```

Dopo aver scritto "turtle" premete il tasto <Enter> per iniziare una nuova linea. In ultimo premete il tasto <Esc> per uscire dall'Insert mode e tornare al Normal mode. Ora ci saranno due linee di testo nella vostra finestra di Vim:

```
+----------------------------------------+
|A very intelligent turtle               |
|Found programming UNIX a hurdle         |
|~                                       |
|~                                       |
|                                        |
+----------------------------------------+
```

QUAL È IL MODO?

Per sapere in quale modo vi trovate, scrivete questo comando: >

```
:set showmode
```

Potete notare che scrivendo il carattere due punti Vim sposta il cursore nell'ultima linea della finestra. In questa linea potete digitare i comandi due punti (comandi che iniziano con il carattere due punti). Il comando viene concluso premendo il tasto <Enter> (tutti i comandi iniziati con i due punti vengono conclusi così).
Adesso, se scrivete il comando "i", Vim farà apparire la scritta --INSERT-- alla base della finestra. Ciò indicherà che vi trovate in Insert mode.

```
+----------------------------------------+
|A very intelligent turtle               |
|Found programming UNIX a hurdle         |
|~                                       |
|~                                       |
|-- INSERISCI --                         |
+----------------------------------------+
```

Premendo <Esc> per tornare al Normal mode l'ultima linea tornerà vuota.

SUPERARE I PROBLEMI

Uno dei problemi per il principiante di Vim è la confusione dei modi, che può avvenire dimenticando in quale modo ci si trovi o scrivendo accidentalmente un comando che cambia modo. Per tornare nel Normal mode non c'è problema, qualunque sia il modo in cui vi troviate premete il tasto <Esc>. Se lo premete due volte Vim vi avvertirà con un suono che siete già nel Normal mode.

==
02.3 Spostarsi attraverso il file

Dopo il vostro ritorno nel Normal mode, vi potete spostare usando questi tasti:

```
h   sinistra                                                    *hjkl*
j   giù
k   su
l   destra
```

A prima vista potrebbe apparire che questi comandi siano stati scelti a

casaccio. Dopo tutto, chi mai ha usato l per dire destra? Ma in realtà c'è una ragione molto valida alla base di queste scelte: lo spostamento del cursore è una delle cose più frequenti in un editor, e questi tasti si trovano in basso a destra nella tastiera. In altre parole questi comandi si trovano dove potete scriverli più velocemente (specialmente se scrivete con dieci dita).

> Nota:
> Potete anche spostare il cursore usando i tasti freccia. Se lo fate, comunque, dovrete rallentare la vostra velocità di digitazione per premerli, dovendo muovere la mano dai tasti testuali a quelli freccia. Considerando che dovrete farlo centinaia di volte all'ora, ciò significa sprecare una considerevole quantità di tempo.
> Inoltre ci sono tastiere che non hanno i tasti freccia, o che li hanno in posizioni insolite; così, conoscere l'uso dei tasti hjkl, aiuta in queste situazioni.

Un modo per ricordarsi di questi comandi è che h si trova a sinistra, l è a destra e j punta all'ingiù. Visualmente: >

```
          k
      h       l
        j
```

Il modo migliore di imparare questi comandi è di usarli. Con il comando "i" inserite alcune linee di testo. Poi provate a spostarvi con i tasti hjkl ed ad inserire qualche parola qua e là. Non scordate di premere <Esc> per tornare al Normal mode. Il |**vimtutor**| è un altro modo piacevole per imparare con la pratica.

Per utenti giapponesi, Hiroshi Iwatani suggerisce di fare così:

```
                    Komsomolsk
                        ^
                        |
   Huan Ho      <--- --->  Los Angeles
(Fiume giallo)          |
                        v
                  Java (l'isola, non il linguaggio)
```

==========

02.4 La cancellazione di caratteri

Per cancellare un carattere, spostate il cursore su di esso e premete "x". (Questo è un retaggio dei vecchi giorni della macchina per scrivere, quando si cancellavano cose scrivendovi sopra xxxx.) Spostando il cursore all'inizio della prima riga della canzoncina e scrivendo xxxxxxx (sette x) si cancellerà "A very ". Il risultato dovrebbe apparire così:

```
+----------------------------------------+
|intelligent turtle                      |
|Found programming UNIX a hurdle         |
|~                                       |
|~                                       |
|                                        |
+----------------------------------------+
```

Adesso si può inserire del testo nuovo, ad esempio scrivendo: >

```
iA young <Esc>
```

Ciò inizia un'inserzione (la i), scrive le parole "A young", ed esce dall'Insert mode (l'<Esc> finale). Il risultato:

```
+----------------------------------------+
|A young intelligent turtle              |
|Found programming UNIX a hurdle         |
|~                                       |
|~                                       |
|                                        |
+----------------------------------------+
```

CANCELLAZIONE DI UN'INTERA LINEA

Per cancellare una linea usate il comando "dd". La linea che segue si sposterà verso l'alto a riempire il vuoto:

```
+----------------------------------------+
|Found programming UNIX a hurdle         |
|~                                       |
|~                                       |
|~                                       |
|                                        |
+----------------------------------------+
```

CANCELLAZIONE DI UN "A CAPO"

In Vim potete unire assieme due linee per farle diventare una sola, ciò significa che l'interruzione di linea tra di esse è stata cancellata. Il comando "J" fa ciò.
   Prendiamo queste due linee:

```
A young intelligent ~
turtle ~
```

Portate il cursore sulla prima linea e premete "J":

```
A young intelligent turtle ~
```

==========

02.5 Undo e Redo

Supponiamo che abbiate cancellato più del dovuto. Bene, potete riscriverlo da capo, ma c'è un modo più semplice. Il comando "u" elimina l'ultima modifica. Osservate questa azione: dopo aver usato "dd" per cancellare la prima linea , "u" la riporta a come era originariamente.
   Ancora una: Portate il cursore sulla A nella prima linea:

```
A young intelligent turtle ~
```

Ora scrivete xxxxxxx per cancellare "A young". Rimarrà quanto segue:

```
 intelligent turtle ~
```

Scrivete "u" per eliminare l'ultima cancellazione. Poiché delete aveva rimosso la g, undo ripristina il carattere.

```
g intelligent turtle ~
```

Un ulteriore comando u ripristina il precedente carattere cancellato:

```
ng intelligent turtle ~
```

Il prossimo comando u darà la u, e così via:

```
ung intelligent turtle ~
oung intelligent turtle ~
young intelligent turtle ~
 young intelligent turtle ~
A young intelligent turtle ~
```

Nota:
Se premete "u" due volte, ed il risultato è che ottenete lo stesso testo, avete Vim configurato per lavorare in modo compatibile Vi. Andate a vedere qui per correggerlo: |**not-compatible**|.
   Questo manuale presume che stiate lavorando in "Modo Vim". Potreste preferire l'utilizzo del vecchio modo Vi, ma dovrete aspettarvi alcune piccole differenze nel testo in quel caso.

REDO

Se avete impiegato il comando u troppe volte, potete premere CTRL-R (redo) per invertire il comando precedente. In altre parole, ciò cancella la

cancellazione. Per vederlo in azione premete CTRL-R due volte. Il carattere A e lo spazio dopo di esso spariranno:

```
        young intelligent turtle ~
```

C'è una versione speciale del comando undo, il comando "U" (undo linea). Il comando undo linea ripristina tutte le modifiche effettuate sull'ultima linea su cui avevate lavorato. Scrivendo questo comando due volte si cancellerà il precedente "U".

```
        A very intelligent turtle ~
          xxxx                          Cancella very

        A intelligent turtle ~
                      xxxxxx            Cancella turtle

        A intelligent ~
                                        Ripristina la linea con "U"
        A very intelligent turtle ~
                                        Cancella "U" usando "u"
        A intelligent ~
```

Il comando "U" modifica da solo, quello che il comando "u" cancella e CTRL-R rifà. Ciò potrebbe essere un tantino confuso. Non preoccupatevi, con un "u" e CTRL-R potrete ripristinare qualunque situazione aveste. Più informazioni nella sezione |**32.2**|.

==========

02.6 Altri comandi

Vim possiede un gran numero di comandi per modificare il testo. Guardate |**Q_in**| ed oltre.
Ve ne sono alcuni usati di rado.

AGGIUNGERE IN FONDO

Il comando "i" inserisce un carattere prima del carattere sotto il cursore.
Lavora bene; ma cosa succede se volete inserirlo alla fine della linea? Per fare ciò dovete inserire il testo dopo il cursore. Ciò si ottiene con il comando "a" (append).
Ad esempio, per cambiare la linea

```
        and that's not saying much for the turtle. ~
```

in

```
        and that's not saying much for the turtle!!! ~
```

spostate il cursore sul punto alla fine della linea. Poi scrivete "x" per cancellare il punto. Il cursore è ora posizionato alla fine della linea sulla e di turtle. Adesso scrivete

```
        a!!!<Esc>
```

per aggiungere i tre punti esclamativi dopo la e in turtle:

```
        and that's not saying much for the turtle!!! ~
```

INSERIRE UNA NUOVA LINEA

Il comando "o" crea una nuova linea vuota sotto il cursore e pone Vim nell'Insert mode. Ora potete scrivere il testo per la nuova linea.
Supponiamo che il cursore sia da qualche parte nella prima di queste due linee:

```
        A very intelligent turtle ~
        Found programming UNIX a hurdle ~
```

Se adesso usate il comando "o" e scrivete il testo:

```
        oThat liked using Vim<Esc>
```

Il risultato sarà:

```
        A very intelligent turtle ~
        That liked using Vim ~
        Found programming UNIX a hurdle ~
```

Il comando "O" (maiuscolo) apre una linea sopra il cursore.

USARE IL NUMERO DI RIPETIZIONI

Immaginiamo di voler salire di nove linee. Potete scrivere "kkkkkkkkk" od usare il comando "9k". Difatti, si possono far precedere molti comandi con un numero.
Prima in questo capitolo, ad esempio avete aggiunto tre punti esclamativi alla fine di una linea scrivendo "a!!!<Esc>". Un altro modo per farlo è di usare il comando "3a!<Esc>". Il numero 3 dice al comando che segue di venire eseguito tre volte. Analogamente per cancellare tre caratteri potete usare il comando "3x". Il numero deve venire prima del comando che deve essere eseguito.

==
02.7 Come uscire

Per uscire usate il comando "ZZ". Questo comando scrive il file ed esce.

> Nota:
> Diversamente da molti altri editor, Vim non farà automaticamente un file di backup. Se scrivete "ZZ", le vostre modifiche verranno sovrascritte e non potrete più tornare indietro. Potete configurare Vim per generare un file di backup, vedete |**07.4**|.

ABBANDONARE LE MODIFICHE

Sovente farete un sacco di modifiche per poi capire improvvisamente di aver fatto qualcosa di diverso da quanto volevate. Nulla di preoccupante; Vim ha un comando esci-e-getta-via-tutto. Si tratta di:

```
        :q!
```

Non scordatevi di premere <Enter> per ultimare il comando.

Per coloro che fossero interessati ai particolari, le tre parti di questo comando sono i due punti (:), che fa entrare in modo Command-line; il comando q che effettua la chiusura dell'editor; ed il modificatore di comando ignora (!).
Il comando ignora è necessario poiché Vim non vorrebbe ignorare le modifiche. Se scriveste soltanto ":q", Vim mostrerebbe un messaggio di errore e rifiuterebbe di uscire:

```
        E37: Non salvato dopo modifica (aggiungi ! per eseguire comunque)"
```

Specificando di ignorare, state effettivamente dicendo a Vim, "Lo so che ciò che sto facendo sembra stupido, ma io sono adulto e voglio davvero fare questo."

Se voleste continuare a lavorare sul file con Vim: il comando ":e!" ricaricherebbe la versione originale del file.

==
02.8 Trovare un aiuto

Tutto ciò che vorreste sapere può essere trovato nei file di help di Vim. Non esitate a chiedere!

Se sapete cosa cercare, è solitamente più semplice cercare usando la documentazione di aiuto (:help), invece di usare Google. Infatti gli argomenti sono esposti seguendo delle appropriate linee guida stilistiche.

Inoltre l'help ha il vantaggio di documentare la particolare versione di Vim che si sta adoperando. Non sono descritti comandi aggiunti in seguito. Questi ultimi, naturalmente, non funzionerebbero nelle versioni precedenti.

Per avere un aiuto generico si usi questo comando:

```
:help
```

Si può usare il primo tasto di funzione <F1>. Se la vostra tastiera avesse un tasto <Help> questo potrebbe funzionare altrettanto bene.
Se non si fornisce un argomento, ":help" mostra la finestra generica di aiuto. I creatori di Vim hanno fatto qualcosa di molto astuto (o pigro) con il sistema di help: hanno fatto la finestra di help come una normale finestra di editing. Potete usare tutti i comandi normali di Vim per navigare attraverso le informazioni di help. Comunque h, j, k, e l operano uno spostamento del cursore verso sinistra, giù, su e destra.
Per uscire dalla finestra di help, usate lo stesso comando che usereste per uscire dall'editor: "ZZ". Si chiuderà solo la finestra di help, non Vim.

Leggendo il testo di aiuto, noterete del testo racchiuso entro barre verticali (ad esempio, |**help**|). Ciò indica un iperlink. Se posizionate il cursore tra le barre e premete CTRL-] (salta al tag), il sistema di help vi darà l'oggetto indicato. (Per ragioni non discusse qui, la terminologia di Vim per un iperlink è tag. Così CTRL-] salta alla posizione del tag indicato dalla parola sotto il cursore.)
Dopo pochi passi, potreste voler tornare indietro. CTRL-T (pop tag) ritorna alla posizione precedente. Lavora bene anche CTRL-O (salta alla posizione più vecchia).
Alla sommità dello schermo di help, c'è la nota *help.txt*. Questo nome tra caratteri "*" viene usata dal sistema di help per definire un tag (destinazione dell'iperlink).
Vedere |**29.1**| circa i particolari per l'uso dei tag.

Per avere aiuto su un oggetto dato, usate il comando seguente:

```
:help {subject}
```

Per ottenere aiuto sul comando "x", ad esempio, scrivete quanto segue:

```
:help x
```

Per scoprire come cancellare del testo, usate questo comando:

```
:help deleting
```

Per avere un completo indice dei comandi di Vim, usate il comando seguente:

```
:help index
```

Se vi servisse aiuto per un comando a carattere di controllo (ad esempio, CTRL-A), dovete specificarlo con il prefisso "CTRL-".

```
:help CTRL-A
```

L'editor Vim ha molte modalità diverse. Di default il sistema di help mostra i comandi in Normal-mode. Ad esempio, il comando che segue mostra un aiuto per il comando CTRL-H in Normal mode: >

```
:help CTRL-H
```

Per identificare gli altri modi, utilizzate un prefisso di modo. Se volete un aiuto per la versione Insert-mode di un comando, usate "i_". Per CTRL-H ciò vi fornisce il comando che segue:

```
:help i_CTRL-H
```

Avviando l'editor Vim, potete usare molti argomenti a linea di comando.
Tutti questi iniziano con un trattino (-). Per trovare cosa faccia l'argomento -t, per esempio, usate il comando:

```
:help -t
```

L'editor Vim possiede un certo numero di opzioni che vi consentono di configurare e personalizzare l'editor. Se voleste un aiuto per qualche opzione, dovete racchiuderla tra due virgolette singole. Per ottenere informazioni su cosa faccia l'opzione 'number', ad esempio, usate il seguente comando:

```
        :help 'number'
```

La tabella con i prefissi per tutti i modi si può trovare più sotto:
|**help-summary**|.

I tasti speciali sono racchiusi tra parentesi angolate. Per ottenere aiuto
sul tasto freccia in su nell'Insert mode, per esempio, usate questo comando: >

```
        :help i_<Up>
```

Se vedeste un messaggio di errore che non capite, ad esempio:

```
        E37: Non salvato dopo modifica (aggiungi ! per eseguire comunque)"
```

Potete usare identificativo che inizia il messaggio per ricevere aiuto a
proposito del messaggio stesso: >

```
        :help E37
```

Sommario: *help-summary* >

0) :help può essere abbreviato in :h
1) Usando Ctrl-D dopo aver digitato un argomento, Vim elenca tutti gli help
   disponibili in proposito.
   Oppure si può immettere Tab per chiedere un completamento. Ad es.: >
        :help color<Tab>
< Per ulteriori informazioni sull'uso dell'help: >
        :help helphelp

2) Si seguano i puntatori racchiusi fra barre verticali per arrivare all'help
   su quel particolare argomento. Si può passare da un help dettagliato alla
   documentazione per l'utente, che descrive alcuni comando da un punto di
   vista dell'utente, con minori dettagli. P.es. dopo: >
        :help pattern.txt
< Si possono vedere gli argomenti |**03.9**| nella Guida utente e |**usr_27.txt**|
   nell'Introduzione.

3) Le opzioni sono racchiuse tra apici singoli. Per vedere l'help riguardo
   all'opzione list: >
        :help 'list'
< Se si sa solo di essere alla ricerca di una certa opzione si può
   immettere: >
        :help options.txt
< per aprire la pagina di help che descrive la gestione di tutte le opzioni
   e da qui cercare il file usando espressioni regolari, p.es. textwidth.
   Alcune opzioni hanno ulteriori suddivisioni. P.es. si usi: >
        :help cpo-<carattere>
< per il flag corrispondente nelle impostazioni 'cpoptions', sostituendo
   <carattere> con un flag specifico, p.es.: >
        :help cpo-;
< E per i flag di guioption: >
        :help go-<lettera>

4) I comandi in modo Normal non hanno un prefisso. Per andare alla pagina di
   help per il comando "gt": >
        :help gt

5) I comandi in modo Insert iniziano con i_. Help per cancellare una parola: >
        :help i_CTRL-W

6) I comandi in modo Visual iniziano con v_. Help per saltare al lato opposto
   dell'area Visual: >
        :help v_o

7) I comandi e gli argomenti per editare una riga iniziano con c_. Help per
   usare l'argomento di comando %: >
        :help c_%

8) I comandi di Ex iniziano sempre con ":", quindi per vedere l'help relativo
   al comando :s si digita: >
        :help :s
```
```

9) Comandi rivolti verso il debugging iniziano con ">". Per vedere l'help relativo al comando debug "cont", immettere: >
```
:help >cont
```

10) Combinazioni di tasti. Iniziano con una singola lettera che indica il modo di Vim in cui si possono usare. P.es.: >
```
:help i_CTRL-X
```
< conduce alla famiglia di comandi Ctrl-X per il modo Insert, che possono essere usati per gli autocompletamenti in differenti situazioni. Nota: certi tasti verranno sempre indicati allo stesso modo, p.es. il tasto Control sarà sempre scritto come CTRL.
Per i comandi in modo Normal, non c'è prefisso, e l'argomento è disponibile scrivendo
:h CTRL-<carattere>. P.es. >
```
:help CTRL-W
```
< A differenza di ciò,
```
:help c_CTRL-R
```
< descriverà quel che Ctrl-R fa quando si immettono comandi sulla riga di comando e >
```
:help v_Ctrl-A
```
< riguarda il modo per incrementare numeri in modo Visual e >
```
:help g_CTRL-A
```
< riguarda il comando g<C-A> (ovvero, si deve immettere "g" e poi <Ctrl-A>). Qui la "g" identifica il comando in modo Normal "g" che si aspetta sempre che venga premuto un secondo tasto, prima di fare qualcosa di simile ai comandi che iniziano per "z".

11) Gli operandi di espressioni regolari iniziano sempre con con /. Quindi, per vedere l'help relativo al quantificatore "\+" nelle espressioni regolari di Vim, si scrive: >
```
:help /\+
```
< Per cominciare dall'inizio a imparare le espressioni regolari, si può leggere: >
```
:help pattern.txt
```

12) I registri iniziano sempre con la parola "quote". Per infomazioni sul registro speciale ":", si digiti: >
```
:help quote:
```

13) Il linguaggio Script di Vim è descritto nel file richiamato immettendo >
```
:help eval.txt
```
< Alcuni aspetti del linguaggio sono disponibili con :h expr-X dove "X" è un singolo carattere. P.es. >
```
:help expr-!
```
< visualizza la descrizione di come funziona l'operatore "!" (Not) negli Script Vim.
È anche importante >
```
:help function-list
```
< per trovare una breve descrizione di tutte le funzioni disponibili. Gli help per le funzioni in script Vim includono sempre "()", ossia: >
```
:help append()
```
< descrive la funzione Vim append, e non invece come aggiungere (appendere) del testo in fondo al buffer corrente.

14) Le mappature sono spiegate nella pagina di help :h |**map.txt**|. Si usi >
```
:help mapmode-i
```
< per informazioni riguardo al comando |**:imap**| command. Si usi anche :map-<argomento>
per trovare aiuto riguardo ad alcuni sottoargomenti delle mappature. P.es.: >
```
:help :map-local
```
< per le mappature locali a un buffer (valide solo in quel buffer) o >
```
:help map-bar
```
< per sapere come è gestito il segno '|' nelle mappature.

15) La definizione di comandi è trattata in :h command-<argomento>, si usi >
```
:help command-bang
```
< per informazioni riguardo all'argomento '!' nei comandi utente.

16) I comandi di gestione finestre iniziano sempre con CTRL-W, e l'help corrispondente si può trovare digitando :h CTRL-W_<carattere>. P.es. >

```
        :help CTRL-W_p
<    per come passare alla finestra precedentemente acceduta.  È sempre
    possibile partire da >
        :help windows.txt
<    e leggere da lì in poi, se si vogliono informazioni sui comandi di
    gestione delle finestre.

17) Si usi |:helpgrep| per ricerche in tutte le pagine di help (comprese
    eventualmente quelle dei plugin installati).  Vedere |:helpgrep| per
    le modalità di uso.
    Per ricercare un argomento: >
        :helpgrep argomento
<    In questo modo si arriva alla prima pagina che ne tratta (potrebbero
   essercene altre).  Per passare alla pagina successiva: >
        :cnext
<    Tutte le possibili corrispondense sono disponibili nella finestre
    quickfix che si può aprire con: >
        :copen
<    Spostandosi sulla corrispondenza desiderata e schiacciando il tasto
    Invio si saltare a quella particolare pagina di help.

18) Il manuale utente.  Qui sono descritti gli argomenti per iniziare a
    editare, in maniera da facilitare utenti non esperti.  Si parta da
    |usr_toc.txt| per esaminare l'indice (Table Of Content): >
        :help usr_toc.txt
<    Si dia un'occhiata ai contenuti, alla ricerca di argomenti interessanti
    La descrizione che riguarda i caratteri speciali e quelli non presenti
    sulla tastiera è nel capitolo 24, e per andarci occorre immettere : >
        :help usr_24.txt
<    Inoltre, se si vuol accedere a un particolare capitolo del manuale
    utente, ci si può arrivare direcctamente così: >
        :help 10.1
<    va alla sezione 1 del capitolo 10, |usr_10.txt| che parla di come si
    possono preparare delle macro.

19) I gruppi di evidenziazione.  Iniziano sempre con hl-<nome-gruppo>.  P.es. >
        :help hl-WarningMsg
<    riguarda l'evidenziazione del gruppo dei messaggi di avvertimento
    WarningMsg.

20) L'evidenziazione sintattica è suddivisa per argomenti :syn-<argomento>.
    P.es. >
        :help :syn-conceal
<    riguarda l'argomento conceal (nascondi) del comando :syn.

21) I comandi quickfix iniziano solitamente con :c mentre di liste di
    posizioni (locazioni) solitamente iniziano con :l

22) Eventi che fanno eseguire autocomandi si possono trovare per nome: >
        :help BufWinLeave
<    Per vedere tutti gli eventi previsti: >
        :help autocommand-events

23) Argomenti binari sulla riga di comando iniziano sempre con "-".  Quindi
    per il file di help dell'argomento binario -f di Vim si userà: >
        :help -f

24) Le funzionalità opzionali iniziano sempre con "+".  Per informazioni
    sulla funzionalità conceal (nascondi) si usi: >
        :help +conceal

25) La documentazione per funzionalità relative a un dato tipo di file sono
    solitamente disponibili nella forma ft-<tipo-file>-<funzionalita>.
    Quindi >
        :help ft-c-syntax
<    parla del file di evidenziazione sintattica del linguaggio C e delle
    opzioni disponibili.  Talvolta ulteriori sezioni per il completamento
    "omni" >
        :help ft-php-omni
<    o per i plugin dipendenti dal tipo di file >
        :help ft-tex-plugin
<    sono disponibili.
```

26) I messaggi di errore e di avvertimento possono essere controllati
direttamente nell'help. Quindi >
:help E297
< porta alla descrizione di un errore nel file di swap e >
:help W10
< riguarda l'avvertimento che si sta cambiando un file che è stato
aperto in sola lettura.
Talora, tuttavia, non si trova una descrizione relativa al messaggio di
errore, ma si arriva a quella del comando Vim che normalmente genera
tali messaggi. Quindi: >
:help E128
< porta alla descrizione del comando |**:function**| command.

==

Capitolo seguente: |**usr_03.txt**| Muoversi nel file

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