usr_27.txt Per Vim version 7.4. Ultima modifica: 2010 Mar 28

VIM USER MANUAL - di Bram Moolenaar
Traduzione di questo capitolo: Stefano Palmeri

Comandi di ricerca ed espressioni

Nel capitolo 3 sono stati menzionati pochi semplici espressioni di ricerca |03.9|. Vim può eseguire delle ricerche molto più complesse. Questo capitolo spiega quelle usate più spesso. Una dettagliata spiegazione può essere trovata qui: |pattern|

Capitolo seguente: |usr_28.txt| La piegatura
Capitolo precedente: |usr_26.txt| Ripetizione
Indice: |usr_toc.txt|

==========

27.1 Ignorare le differenze tra i caratteri maiuscoli e minuscoli

Di default, le ricerche di Vim rispettano le differenze tra caratteri maiuscoli e minuscoli. Quindi, "include", "INCLUDE" e "Include" sono tre parole differenti ed una ricerca ne troverà solo una.
Adesso abilitate l'opzione 'ignorecase': >

```
:set ignorecase
```

Cercate di nuovo "include" e ora corrisponderà a "Include", "INCLUDE" e "InClUDe". (Impostate l'opzione 'hlsearch' per vedere velocemente dove una espressione corrisponda.)
Potete disabilitarla di nuovo con : >

```
:set noignorecase
```

Lasciamola però impostata e cerchiamo "INCLUDE". Troverà esattamente lo stesso testo che "include" aveva trovato. Adesso impostiamo l'opzione 'smartcase' : >

```
:set ignorecase smartcase
```

Se avete un'espressione con almeno un carattere maiuscolo, la ricerca rispetterà le differenze tra i caratteri maiuscoli e minuscoli. L'idea è che voi non eravate obbligati a scrivere quel carattere maiuscolo, quindi dovete averlo fatto perché volevate cercare una maiuscola. Questo è intelligente!
Con queste due opzioni impostate troverete le seguenti corrispondenze:

espressione	corrispondenze ~
word	word, Word, WORD, WoRd, etc.
Word	Word
WORD	WORD
WoRd	WoRd

CARATTERI MAIUSCOLI E MINUSCOLI IN UNA SOLA ESPRESSIONE

Se volete ignorare le differenze tra i caratteri maiuscoli e minuscoli per uno specifica espressione, potete farlo anteponendo la stringa "\c". Usare "\C" fa sì che l'espressione rispetti le differenze. Tutto ciò sovrascrive le opzioni 'ignorecase' e 'smartcase'; quando "\c" o "\C" vengono usate i loro valori non contano.

espressione	corrispondenze ~
\Cword	word

```
        \CWord                  Word
        \cword                  word, Word, WORD, WoRd, etc.
        \cWord                  word, Word, WORD, WoRd, etc.
```

Un grande vantaggio nell'usare "\c" e "\C" è che esse rimangono con l'espressione. Così se voi ripetete un'espressione dalla cronologia delle ricerche, succederà la stessa cosa, senza che abbia importanza se **'ignorecase'** o **'smartcase'** siano state cambiate.

> Nota:
> L'uso di argomenti "\" nelle espressioni di ricerca dipende dall'opzione **'magic'**. In questo capitolo noi assumiamo che **'magic'** sia attiva, poiché questa è l'impostazione standard e raccomandata. Se voleste cambiare **'magic'**, molte espressioni di ricerca all'improvviso diventerebbero non valide.

> Nota:
> Se la vostra ricerca richiede più tempo di quanto vi aspettavate, potete interromperla con **CTRL-C** in Unix e **CTRL-Break** in MS-DOS e MS-Windows.

==========

27.2 Aggirare (nella ricerca) la fine del file

Di default, una ricerca in avanti inizia a cercare la stringa data dalla posizione corrente del cursore. Essa poi continua fino alla fine del file. Se fino ad allora non ha trovato la stringa, comincia da principio e cerca dall'inizio del file fino alla posizione del cursore.
Tenete a mente che ripetendo il comando "n" per cercare la corrispondenza successiva, potreste eventualmente tornare alla prima corrispondenza. Se non vi accorgete di questo, cercherete all'infinito! Per darvi un consiglio, Vim mostra questo messaggio:

> **raggiunto il FONDO nella ricerca, continuo dalla CIMA** ~

Se usate il comando "?", per cercare nell'altra direzione, ricevete questo messaggio:

> **raggiunta la CIMA nella ricerca, continuo dal FONDO** ~

Tuttavia, voi non sapete quando siete tornati alla prima corrispondenza. Un modo per saperlo è quello di attivare l'opzione **'ruler'**: >

```
:set ruler
```

Vim mostrerà la posizione del cursore nell'angolo in basso a destra della finestra (nella linea di stato, se ce n'è una). Appare così:

> **101,29 84%** ~

Il primo numero è il numero di linea del cursore. Ricordate il numero di linea dal quale siete partiti, affinché possiate controllare se avete passato di nuovo questa posizione.

NON AGGIRARE LA FINE DEL FILE

Per disabilitare l'aggiramento della fine del file in una ricerca , usate il seguente comando: >

```
:set nowrapscan
```

Adesso quando la ricerca raggiunge la fine del file, un messaggio d'errore dice:

> **E385: la ricerca ha raggiunto il FONDO senza successo per: %s** ~

Così potete trovare tutte le corrispondenze andando all'inizio del file con "gg" e continuando a cercare fino a che non vedete questo messaggio.
Se cercate nell'altra direzione, usando "?", voi ricevete:

> **E384: la ricerca ha raggiunto la CIMA senza successo per: %s** ~

==
27.3 Scostamento

Di default, il comando di ricerca lascia il cursore posizionato all'inizio dell'espressione. Potete dire a Vim di lasciarlo in qualche altro posto specificando uno scostamento. Per il comando di ricerca in avanti "/", lo scostamento è specificato aggiungendo una barra (/) e lo scostamento: >

```
/default/2
```

Questo comando cerca l'espressione "default" e poi sposta il cursore all'inizio della seconda linea dopo la stessa. Usando questo comando per il paragrafo sopra, Vim trova la parola "default" nella prima linea. Poi il cursore è spostato due linee più sotto e si posa su "specificando".

Se lo scostamento è un semplice numero, il cursore sarà posizionato all'inizio della linea che dista quel numero di linee dalla corrispondenza. Lo scostamento numerico può essere positivo o negativo. Se è positivo, il cursore si sposta in basso di quel numero di linee; se è negativo si sposta verso l'alto.

SCOSTAMENTO DEI CARATTERI

Lo scostamento "e" indica uno scostamento dalla fine della corrispondenza. Esso sposta il cursore sull'ultimo carattere della corrispondenza. Il comando: >

```
/const/e
```

mette il cursore sulla "t" di "const".
Da quella posizione, aggiungendo un numero si sposta in avanti di quel numero di caratteri. Questo comando muove il cursore sul carattere proprio dopo la corrispondenza: >

```
/const/e+1
```

Un numero positivo sposta il cursore verso destra, uno negativo lo sposta verso sinistra. Ad esempio: >

```
/const/e-1
```

posiziona il cursore sulla "s" di "const".

Se lo scostamento comincia con "b", il cursore si sposta all'inizio dell'espressione. Questo non è molto utile, dal momento che lasciar fuori la "b" fa la stessa cosa. Diventa utile quando si aggiunge o si sottrae un numero. Il cursore dopo va avanti o indietro di quel numero di caratteri. Per esempio: >

```
/const/b+2
```

Posiziona il cursore all'inizio della corrispondenza e poi due caratteri verso destra. Così si posa sulla "n".

RIPETIZIONE

Per ripetere una ricerca della precedente espressione, ma con un diverso scostamento, omettete l'espressione: >

```
/that
//e
```

È uguale a: >

```
/that/e
```

Per ripetere con lo stesso scostamento: >

```
/
```

"n" fa la stessa cosa. Per ripetere ed al tempo stesso rimuovere un offset usato in precedenza: >

```
//
```

CERCARE ALL'INDIETRO

Il comando "?" usa gli offset nello stesso modo, ma dovete usare "?" per separare l'offset dall'espressione, anziché "/": >

```
?const?e-2
```

La "b" e la "e" mantengono il loro significato; essi non cambiano direzione se si usa "?".

POSIZIONE DI PARTENZA

Quando si comincia una ricerca, essa normalmente inizia dalla posizione del cursore. Quando voi specificate uno scostamento di linea, ciò può causare dei problemi. Ad esempio: >

```
/const/-2
```

Questo trova la successiva parola "const" e quindi sposta il cursore due linee verso l'alto. Se usate "n" per cercare di nuovo, Vim potrebbe iniziare dalla posizione corrente e trovare la stessa corrispondenza "const". Allora, usando di nuovo lo scostamento, potreste tornare da dove eravate partiti. Dovreste esservi bloccati!
   Potrebbe essere peggio: supponete che ci sia un'altra corrispondenza con "const" nella linea successiva. In questo caso ripetendo la ricerca in avanti trovereste questa corrispondenza e spostereste il cursore due linee insù.
   Così in realtà spostereste il cursore indietro!

Quando voi specificate uno scostamento di carattere, Vim terrà conto di ciò. In questo modo la ricerca inizia pochi caratteri avanti o indietro, cosicché la stessa corrispondenza non verrà trovata di nuovo.

==========

27.4 Effettuare più volte la ricerca

L'argomento "*" specifica che l'argomento che lo precede può corrispondere un qualsiasi numero di volte.
Così: >

```
/a*
```

corrisponde ad "a", "aa", "aaa", etc. Ma anche a "" (la stringa vuota), poiché le zero volte sono incluse.
   Il segno "*" influisce solo sull'argomento direttamente prima di esso. Quindi "ab*" corrisponde ad "a", "ab", "abb", "abbb", etc. Per far sì che di un'intera stringa si trovino corrispondenze multiple, essa deve essere raggruppata in un unico argomento. Questo si fa mettendo "\(" prima della stringa e "\)" dopo di essa. Così questo comando: >

```
/\(ab\)*
```

corrisponde a: "ab", "abab", "ababab", etc. Corrisponde anche a "".

Per evitare di trovare le corrispondenze con le stringhe vuote, usate "\+". Questo fa sì che l'argomento precedente abbia una o più corrispondenze: >

```
/ab\+
```

Corrisponde ad "ab", "abb", "abbb", etc. Non corrisponde ad "a" quando questa non è seguita da una "b".

Per trovare le corrispondenze con un argomento con diverse opzioni, usate "\=". Esempio: >

```
/folders\=
```

corrisponde a "folder" e "folders".

CONTEGGI SPECIFICI

Per trovare uno specifico numero di corrispondenze degli argomenti usate il formato "\{n,m}". "n" e "m" sono numeri. L'argomento che lo precede corrisponderà da "n" a "m" volte (incluse |inclusive|). Esempio: >

```
/ab\{3,5}
```

corrisponde ad "abbb", "abbbb" e "abbbbb".
Quando "n" è omesso, esso assume il valore di zero. Quando "m" è omesso, esso assume il valore di infinito. Quando ",m" è omesso, esso troverà corrispondenze esattamente "n" volte.
Esempi:

espressione	conteggio corrispondenze ~
\{,4}	0, 1, 2, 3 o 4
\{3,}	3, 4, 5, etc.
\{0,1}	0 o 1, lo stesso che \=
\{0,}	0 o più, lo stesso che *
\{1,}	1 o più, lo stesso che \+
\{3}	3

CERCARE IL MENO POSSIBILE

Fin qui gli argomenti cercano corrispondenze col maggior numero possibile di caratteri che possono trovare. Per cercare il numero minimo di corrispondenze, usate "\{-n,m}". Funziona allo stesso modo di "\{n,m}", eccetto per il fatto che ne viene usata la minore quantità possibile.
Per esempio, usate: >

```
/ab\{-1,3}
```

Troverà "ab" in "abbb". In realtà, esso non cercherà mai più di una "b", perché non c'è ragione di cercare oltre. Qualcos'altro è richiesto per forzare la ricerca di corrispondenze oltre il limite più basso.
Le stesse regole si applicano rimuovendo "n" e "m". È anche possibile rimuoverle entrambi i numeri, risultando in "\{-}". Questo trova l'argomento prima di esso zero o più volte, il meno possibile. L'argomento in sé stesso corrisponde sempre zero volte. È utile quando è combinato con qualcos'altro.
Esempio: >

```
/a.\{-}b
```

Questo trova le corrispondenze "axb" in "axbxb". Se si usasse questa espressione: >

```
/a.*b
```

Esso cercherebbe di trovare corrispondenza con il maggior numero possibile di caratteri con ".*", ossia troverebbe "axbxb" interamente.

==========

27.5 Alternative

L'operatore "or" in una espressione è "\|". Esempio: >

```
/foo\|bar
```

Questo trova "foo" o "bar". Più alternative possono venire concatenate: >

```
/one\|two\|three
```

Cerca "one", "two" e "three".
Per ricercare più volte, tutto deve essere posto tra "\(" e "\)": >

```
/\(foo\|bar\)\+
```

Questo trova "foo", "foobar", "foofoo", "barfoobar", etc.
Un altro esempio: >

```
/end\(if\|while\|for\)
```

Questo trova "endif", "endwhile" e "endfor".

Un argomento correlato è "\&". Questo richiede che entrambe le alternative corrispondano nello stesso posto. La ricerca risultante usa l'ultima alternativa. Esempio: >

```
/forever\&...
```

Questo trova "for" in "forever". Non troverà la corrispondenza in "fortuin", per esempio.

==========

27.6 Intervalli di caratteri

Per cercare "a", "b" o "c" potete usare "/a\|b\|c". Quando voleste cercare tutte le corrispondenze di tutte le lettere dalla "a" alla "z", ciò richiederebbe molto tempo. C'è un metodo più breve: >

```
/[a-z]
```

Il costrutto [] trova un singolo carattere. All'interno specificate quali caratteri cercare. Potete includere una lista di caratteri, come questa: >

```
/[0123456789abcdef]
```

Questo troverà le corrispondenze di ogni carattere incluso. Per i caratteri consecutivi potete specificare l'intervallo. "0-3" è uguale a "0123". "w-z" sta per "wxyz". Così lo stesso comando appena visto sopra può essere abbreviato in: >

```
/[0-9a-f]
```

Per trovare proprio il carattere "-" fate in modo che sia il primo o l'ultimo dell'intervallo.
Questi caratteri speciali sono accettati per rendere più facile usarli dentro un intervallo [] (essi in realtà possono essere usati dovunque nella espressione di ricerca):

```
\e      <Esc>
\t      <Tab>
\r      <CR>
\b      <BS>
```

Ci sono alcune situazioni speciali riguardo agli intervalli []; vedete |/[]| per l'intera storia.

INTERVALLI COMPLEMENTARI

Per escludere la ricerca di uno specifico carattere, usate "^" all'inizio di un intervallo. L'argomento di [] quindi cerca tutte le corrispondenze tranne i caratteri inclusi. Esempio: >

```
/"[^"]*"
<
 "        le virgolette
  [^"]    qualsiasi carattere che non siano virgolette
      *   il maggior numero possibile
       "  di nuovo le virgolette
```

Questo corrisponde a "foo" e "3!x", virgolette incluse.

INTERVALLI PREDEFINITI

Un certo numero di intervalli sono usati molto spesso. Vim offre una scorciatoia per questi. Per esempio: >

```
/\a
```

Cerca caratteri alfabetici. Questo equivale a usare "/[a-zA-Z]". Qui di seguito ci sono alcune di queste scorciatoie:

argomento	corrispondenze	equivale a ~	
\d	numeri	[0-9]	
\D	non-numeri	[^0-9]	
\x	numeri esadecimali	[0-9a-fA-F]	
\X	numeri non-esadecimali	[^0-9a-fA-F]	
\s	spazio bianco	[ ]	(<Tab> e <Space>)
\S	caratteri non-bianchi	[^]	(non <Tab> e <Space>)
\l	caratteri alfabetici	[a-z]	
\L	non-lettere minuscole	[^a-z]	
\u	lettere maiuscole	[A-Z]	
\U	non-lettere maiuscole	[^A-Z]	

Nota:
Usare questi intervalli predefiniti funziona molto più velocemente che non gli intervalli di caratteri equivalenti.
Questi argomenti non possono essere usati all'interno di []. Quindi "[\d\l]" NON trova le corrispondenze di un numero o di lettera minuscola. Usate invece "\(\d\|\l\)".

Vedete |/\s| per avere l'intera lista di questi intervalli.

==========

27.7 Classi di caratteri

L'intervallo di caratteri trova le corrispondenze con una serie prefissata di caratteri. Una classe di caratteri è simile, ma con una differenza essenziale: la serie di caratteri può essere ridefinita senza cambiare l'espressione di ricerca.
Ad esempio, cercate questa espressione: >

```
/\f\+
```

L'argomento "\f" sta per caratteri di nomi di file. Quindi corrisponde a sequenze di caratteri che possono essere il nome di un file.
Quali caratteri possono far parte del nome di un file dipende dal sistema operativo che voi state usando. In MS-Windows, la backslash è inclusa, in Unix non lo è. Questo è specificato con l'opzione 'isfname'. Il valore di default per Unix è: >

```
:set isfname
isfname=@,48-57,/,.,-,_,+,,,#,$,%,~,=
```

Per altri sistemi il valore di default è diverso. Quindi potete creare una espressione di ricerca con "\f" per trovare il nome di un file ed esso automaticamente si adatterà al sistema nel quale lo state usando.

Nota:
In realtà, Unix permette di usare qualsiasi carattere nel nome di un file, spazi bianchi inclusi. Includere questi caratteri in 'isfname' sarebbe teoricamente corretto, ma renderebbe impossibile trovare la fine del nome del file nel testo. Quindi il valore predefinito di 'isfname' è un compromesso.

Le classi di caratteri sono:

argomento	corrispondenze	opzioni ~
\i	caratteri di identificativi	'isident'
\I	come \i, escludendo i numeri	
\k	caratteri di parole chiave	'iskeyword'
\K	come \k, escludendo i numeri	
\p	caratteri stampabili	'isprint'
\P	come \p, escludendo i numeri	
\f	caratteri di nomi di file	'isfname'
\F	come \f, escludendo i numeri	

==========

27.8 Ricerca di interruzioni di linea

Vim può trovare una espressione che includa una interruzione di linea. Dovete specificare dov'è l'interruzione, poiché tutti gli argomenti menzionati fin

qui non cercano le interruzioni di linea.
   Per verificare una interruzione di linea in una posizione specifica, usate l'argomento "\n" : >

```
/the\nword
```

Questo corrisponderà a una linea che finisca con "the" seguita da una linea che inizi con "word". Per trovare "the word" così com'è, dovete trovare una interruzione o uno spazio bianco.
L'argomento da usare per far questo è "_s": >

```
/the\_sword
```

Per permettere qualsiasi numero di spazi bianchi: >

```
/the\_s\+word
```

Questo trova le corrispondenze anche quando "the " è alla fine di una linea e "   word" si trova all'inizio della successiva.

"\s" trova lo spazio bianco, "_s" trova lo spazio bianco o una interruzione di linea. Similarmente, "\a" corrisponde a un carattere alfabetico e "_a" corrisponde ad un carattere alfabetico o a una interruzione di linea. Le altre classi o intervalli di caratteri possono essere modificati nello stesso modo inserendo un "_".

Molti altri argomenti possono essere creati per trovare una interruzione di linea anteponendo "_". Ad esempio: "_." corrisponde ad ogni carattere od interruzione di linea.

Nota:
"_.*" trova qualsiasi cosa fino alla fine del file. State attenti con questo, poiché può rendere un comando di ricerca molto lento.

Un altro esempio è "_[]", un intervallo di caratteri che includa una interruzione di linea: >

```
/"\_[^"]*"
```

Questo trova un testo tra virgolette che può essere distribuito in diverse linee.

==========

27.9 Esempi

Ecco alcune espressioni di ricerca che voi potreste trovare utili. Mostrano come possono essere combinati gli argomenti menzionati sopra.

TROVARE UNA TARGA AUTOMOBILISTICA DELLA CALIFORNIA

Un semplice numero di targa è "1MGU103". Esso ha un numero, tre lettere maiuscole e tre numeri. Mettiamo questo direttamente in una espressione di ricerca: >

```
/\d\u\u\u\d\d\d
```

Un altro metodo è quello di specificare che ci sono tre numeri e lettere con un conteggio: >

```
/\d\u\{3}\d\{3}
```

Usando gli intervalli [] invece: >

```
/[0-9][A-Z]\{3}[0-9]\{3}
```

Quali tra questi potreste usare? Quello che vi ricordate. Il metodo semplice che riuscite a ricordare è molto più veloce del metodo fantasioso che non riuscite a ricordare. Se riuscite a ricordarli tutti, evitate l'ultimo, poiché è sia più lungo da digitare, che più lento da eseguire.

TROVARE UN IDENTIFICATIVO

Nei programmi in C (e in molti altri linguaggi del computer), un identificativo inizia con una lettera e più oltre è composto da lettere e numeri. Anche gli underscore ("_") possono essere usati. Questi possono essere trovati con: >

```
/\<\h\w*\>
```

"\<" e "\>" sono usati per trovare solo parole intere. "\h" equivale a "[A-Za-z_]" e "\w" sta per "[0-9A-Za-z_]".

Nota:
"\<" e "\>" dipendono dall'opzione **'iskeyword'**. Se essa include "-", per esempio, poi "ident-" non è trovato. In questa situazione usate: >

```
/\w\@<!\h\w*\w\@!
<
```

Questo controlla se "\w" non trova corrispondenze prima o dopo l'identificativo.
Vedete |**/\@<!**| e |**/\@!**|.

==========

Capitolo seguente: |**usr_28.txt**| La piegatura

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