usr_12.txt   Per Vim version 7.0. Ultima modifica: 2006 Apr 24

VIM USER MANUAL - di Bram Moolenaar
Traduzione di questo capitolo: Rossella Diomede

Trucchi ingegnosi

Combinando diversi comandi potete far fare a Vim quasi ogni cosa. In questo capitolo verrà presentata una serie di combinazioni utili. Utilizzeremo i comandi introdotti nei capitoli precedenti e molti altri.

|12.1| Sostituzione di una parola
|12.2| Modifica di "Last, First" in "First Last"
|12.3| Ordinamento di un elenco
|12.4| Inversione dell'ordine delle righe
|12.5| Conteggio di parole
|12.6| Ricerca di una pagina man
|12.7| Eliminazione di spazi vuoti
|12.8| Ricerca di una parola all'interno di un file

  Capitolo seguente: |usr_20.txt| Immissione rapida dei comandi sulla linea di comando
Capitolo precedente: |usr_11.txt| Recupero dopo un blocco
             Indice: |usr_toc.txt|

==

12.1 Sostituzione di una parola

Il comando substitute può essere utilizzato per sostituire tutte le occorrenze di una parola con un'altra: >

```
:%s/four/4/g
```

L'intervallo "%" significa sostituirla in ogni riga. Il flag "g" alla fine fa sostituire tutte le parole nella stessa riga.
   Il comando non farà la cosa giusta se il vostro file contenesse anche "thirtyfour". Esso verrebbe sostituito da "thirty4". Per evitare ciò, utilizzate l'elemento "\<" per indicare l'inizio di una parola: >

```
:%s/\<four/4/g
```

Ovviamente, andrà ancora male con "fourty". Utilizzate "\>" per indicare il termine di una parola: >

```
:%s/\<four\>/4/g
```

Se state programmando, potreste voler sostituire "four" nei commenti, ma non nel codice. Poiché ciò è difficile da specificare, aggiungete il flag "c" per far sì che il comando substitute chieda quando effettuare la sostituzione: >

```
:%s/\<four\>/4/gc
```

SOSTITUZIONE IN PIU' FILE

Immaginate di voler sostituire una parola in più di un file. Potete aprire ciascun file e digitare manualmente il comando. E' molto più veloce utilizzare record e playback.
   Supponete di avere una directory con file C++, che terminano con ".cpp". C'è una funzione chiamata "GetResp" che volete rinominare "GetAnswer".

```
vim *.cpp               Avviate Vim, definendo la lista di argomenti
                        che contiene tutti i file C++ .  Siete ora
                        nel primo file.
qq                      Iniziate la registrazione nel registro q
:%s/\<GetResp\>/GetAnswer/g
                        Effettuate le sostituzioni nel primo file.
:wnext                  Scrivete il file e passate al successivo.
q                       Interrompete la registrazione.
@q                      Eseguite il registro q.  Ciò ripeterà la
                        sostituzione e il ":wnext".  Potete verificare
                        che non ci sia un messaggio di errore.
999@q                   Eseguite il registro q nei restanti file.
```

Nell'ultimo file avrete un messaggio di errore, in quanto ":wnext" non può passare al file successivo. Ciò fermerà l'esecuzione e tutte le sostituzioni saranno state effettuate.

Note:
Durante la lettura di una sequenza registrata, un errore arresta l'esecuzione.
Pertanto, accertatevi che non ci sia un messaggio di errore durante la registrazione.

C'è un problema: se uno dei file .cpp non contiene la parola "GetResp", avrete un messaggio di errore e la sostituzione verrà interrotta. Per evitare ciò, aggiungete il flag "e" al comando substitute: **>**

```
:%s/\<GetResp\>/GetAnswer/ge
```

Il flag "e" dice al comando :substitute che non è un errore non trovare corrispondenze.

==========

12.2 Modifica di "Last, First" in "First Last"

Avete una lista di nomi in questo formato:

```
Doe, John ~
Smith, Peter ~
```

La volete modificare in:

```
John Doe ~
Peter Smith ~
```

Ciò può essere fatto con un unico comando: **>**

```
:%s/\([^,]*\), \(.*\)/\2 \1/
```

Suddividiamolo in più parti. Ovviamente esso inizia con un comando substitute. Il "%" è la lunghezza della riga, che rimane per l'intero file. In tal modo la sostituzione viene effettuata in ogni riga del file.
Gli argomenti del comando substitute sono "/from/to/". Gli slash separano il modello "from" e la stringa "to". Cioè che il modello "from" contiene:

```
                                                \([^,]*\), \(.*\) ~

La prima parte tra \( \) corrisponde a "Last"   \(     \)
    corrisponde a nessuna virgola                        [^,]
    ogni volta                                       *
corrisponde a ", " letteralmente                        ,
La seconda parte tra \( \) corrisponde a "First"          \(  \)
    ogni carattere                                          .
    ogni volta                                                *
```

Nella parte "to" avete "\2" e "\1". Essi si chiamano backreferences. Fanno riferimento al testo cui corrispondono le parti "\(\)" nel modello. "\2" fa riferimento al testo cui corrisponde il secondo "\(\)", che è il nome "First". "\1" fa riferimento al primo "\(\)", che è il nome "Last".
Potete utilizzare più di nove backreferences nella parte "to" del comando substitute. "\0" sta per l'intero modello trovato. Ci sono diversi altri elementi speciali nel comando substitute. Vedere |sub-replace-special|.

==========

12.3 Ordinamento di un elenco

All'interno di un Makefile spesso è presente un elenco di file. Ad esempio:

```
OBJS = \ ~
        version.o \ ~
        pch.o \ ~
        getopt.o \ ~
        util.o \ ~
        getopt1.o \ ~
        inp.o \ ~
        patch.o \ ~
        backup.o ~
```

Per ordinare l'elenco, filtrate il testo attraverso il comando esterno di ordinamento: **>**

```
/^OBJS
j
:.,/^$/-1!sort
```

Esso va alla prima riga, dove "OBJS" è il primo elemento nella riga. Poi va alla riga successiva e filtra le righe fino ad arrivare ad una riga vuota. Potete anche selezionare le righe con la modalità Visual e poi utilizzare "!sort". Questo è più semplice da digitare, ma presenta maggior lavoro quando ci sono molte righe.
   Il risultato è il seguente:

```
OBJS = \ ~
        backup.o ~
        getopt.o \ ~
        getopt1.o \ ~
        inp.o \ ~
        patch.o \ ~
        pch.o \ ~
        util.o \ ~
        version.o \ ~
```

Attenzione alla presenza di un backslash al termine di ciascuna riga, usato per indicare che la riga continua. Dopo l'operazione si evidenzia un errore. La riga "backup.o" che era alla fine dell'elenco non ha il backslash. Ora, poiché viene inserita nell'elenco in una posizione diversa, è necessario che abbia il backslash.
   La soluzione più semplice è quella di aggiungerla digitando "A \<Esc>". Potete mantenere il backslash nell'ultima riga se siete certi che poi ci sia una riga vuota. In tal modo non avrete più questo problema.

==
12.4 Inversione dell'ordine delle righe

Il comando |:global| può essere combinato con il comando |:move| per spostare tutte le righe prima della riga iniziale, ottenendo come risultato un file invertito. Il comando è: >

```
:global/^/m 0
```

Abbreviato: >

```
:g/^/m 0
```

L'espressione "^" segna l'inizio della riga (anche se la riga è vuota). Il comando |:move| sposta la riga trovata dopo la mitica riga zero, in modo che diventi la prima riga del file. Poiché il comando |:global| non viene confuso dalla numerazione delle righe modificata, |:global| continua a cercare tutte le restanti righe del file ed a porle ciascuna per prima.

Questo comando opera anche su di un intervallo di righe. Per prima cosa spostatevi sulla prima riga e contrassegnatela con "mt". Quindi spostate il cursore sull'ultima riga dell'intervallo e digitate: >

```
:'t+1,.g/^/m 't
```

==
12.5 Conteggio di parole

Capita a volte di dover scrivere un testo con un numero massimo di parole. Vim può contare le parole per voi.
   Se volete contare le parole dell'intero file, utilizzate questo comando: >

```
g CTRL-G
```

Non digitare uno spazio dopo la g, qui è stato usato per rendere il comando facile da leggere.
   Il risultato appare così:

```
Col 1 of 0; Line 141 of 157; Word 748 of 774; Byte 4489 of 4976 ~
```

Potete vedere su quale parola siete (748), ed il numero complessivo di parole nel file (774).

Quando il testo è solo una parte del file, potete spostarvi all'inizio del testo, battere "g CTRL-G", spostarvi alla fine del testo, battere ancora "g CTRL-G", e calcolare così la differenza nella posizione della parola. E' un buon esercizio ma esiste un sistema più semplice. Con il Visual mode, selezionate il testo del quale volete contare le parole. Digitate quindi g CTRL-G. Risultato:

```
Selezionate 5 di 293 righe; 70 di 1884 parole; 359 di 10928 byte ~
```

Per altri metodi relativi al conteggio di parole, righe ed altri elementi, vedere |count-items|.

==
12.6 Ricerca di una pagina man   *find-manpage*

Nel modificare uno script shell o un programma C, state usando un comando o una funzione di cui volete cercare la pagina man (su Unix). Utilizziamo prima un semplice sistema: spostate il cursore sulla parola per la quale si chiede aiuto e premete **>**

```
K
```

Vim eseguirà il programma esterno "man" sulla parola. Se la pagina man è stata trovata, viene visualizzata. Tale sistema utilizza il paginatore normale per scorrere il testo (di solito il programma "more"). Premendo il tasto <Invio> quando terminato, si tornerà a Vim.

Uno svantaggio è quello di non poter vedere contemporaneamente la pagina man ed il testo sul quale state lavorando. Esiste un artificio per far sì che la pagina man compaia in una finestra di Vim. Per prima cosa, caricate il plugin del filetype di man: **>**

```
:runtime! ftplugin/man.vim
```

Inserite questo comando nel file vimrc se prevedete di effettuare questa operazione spesso. Potete ora utilizzare il comando ":Man" per aprire una finestra su una pagina man: **>**

```
:Man csh
```

Potete scorrere il testo e noterete che questo è evidenziato. Ciò vi consente di trovare l'aiuto che stavate cercando. Utilizzate CTRL-W per spostarvi nella finestra contenente il testo sul quale stavate lavorando.
   Per cercare una pagina man in un sezione specifica, inserite come prima cosa il numero della sezione.
Ad esempio, per cercare "echo" nella sezione 3: **>**

```
:Man 3 echo
```

Per spostarvi ad un'altra pagina man, che è nel testo nel formato tipico "word(1)", premete CTRL-] su di essa. Ulteriori comandi ":Man" utilizzeranno la stessa finestra.

Per visualizzare una pagina man per la parola al di sotto del cursore, utilizzate: **>**

```
\K
```

(Se avete ridefinito il <Leader>, utilizzatelo al posto del backslash). Ad esempio, volete conoscere il valore di ritorno di "strstr()" mentre digitate la riga:

```
if ( strstr (input, "aap") == ) ~
```

Spostate il cursore su "strstr" e digitate "\K". Si aprirà una finestra per visualizzare la pagina man per strstr().

==
12.7 Eliminazione di spazi vuoti

Alcuni trovano inutili e brutti gli spazi e le tabulazioni alla fine di una riga. Per eliminare gli spazi alla fine di ciascuna riga eseguite il seguente comando: **>**

```
:%s/\s\+$//
```

Viene usato l'intervallo di riga "%", in tal modo questo funzionerà per l'intero file. Il modello a cui il comando ":substitute" corrisponde è "\s\+$". Esso troverà spazi vuoti (\s), 1 o più (\+), prima della fine della riga ($). Più avanti sarà illustrato come scrivere modelli come questo |usr_27.txt|.
   La parte "to" del comando di sostituzione è vuota: "//". In tal modo essa non viene sostituita con nessun carattere, eliminando effettivamente gli spazi vuoti trovati.

Un altro uso inutile degli spazi è quando vengono posizionati prima di una

tabulazione. Spesso possono essere eliminati senza alterare il numero degli spazi necessari. Ma non sempre è possibile. Pertanto, è preferibile eseguire l'operazione manualmente. Utilizzate questo comando di ricerca: **>**

```
/
```

Non potete vederlo, ma in questo comando c'è uno spazio prima della tabulazione. Quindi è "/<Space><Tab>". Utilizzate ora "x" per eliminare lo spazio e verificate che il numero degli spazi non è stato modificato. Se fosse stato modificato sarebbe necessario inserire una tabulazione. Digitate "n" per trovare la prossima corrispondenza. Ripetete questa operazione finché non ci saranno più altre corrispondenze.

==

12.8 Ricerca di una parola all'interno di un file

Se siete utenti di UNIX, potete utilizzare una combinazione di comandi di Vim e grep per modificare tutti i file che contengano una determinata parola. Ciò risulta estremamente utile quando state lavorando su di un programma e volete visualizzare o modificare i file che contengono una variabile specifica.
   Ad esempio, immaginate di voler modificare tutti i file del programma C che contengono la parola "frame_counter". Per fare ciò utilizzate il comando: **>**

```
vim `grep -l frame_counter *.c`
```

Osserviamo il comando nei dettagli. Il comando grep cerca in un gruppo di file una determinata parola. Poiché l'argomento -l è indicato, il comando elencherà soltanto i file che contengono la parola e non le righe corrispondenti. La parola da trovare è "frame_counter". In realtà, essa può essere qualsiasi espressione regolare. (Note: Quella che grep considera un'espressione regolare non è considerata allo stesso modo da Vim.)
   L'intero comando è racchiuso tra apici inversi (`). Questo comunica alla shell di UNIX di eseguire il comando e pretende che i risultati vengano scritti sulla linea di comando. Pertanto ciò che avviene è che il comando grep viene eseguito e produce una lista di file, i quali vengono inseriti nella riga di comando di Vim. Vim modificherà la lista di file data come output da grep. Potete quindi utilizzare comandi come ":next" e ":first" per dare un'occhiata tra i file.

RICERCA DI UNA PAROLA ALL'INTERNO DI UNA RIGA

Il comando sopra descritto cerca esclusivamente i file che contengono la parola. Dovete ora cercare la parola all'interno dei file.
   Vim ha un comando incorporato che potete utilizzare per cercare una determinata stringa all'interno di un gruppo di file. Se volete trovare tutte le ripetizioni di "error_string" in tutti i file del programma C, ad esempio, digitate il seguente comando: **>**

```
:grep error_string *.c
```

Esso fa sì che Vim cerchi la stringa "error_string" in tutti i file (*.c). L'editor aprirà il primo file dove è stata trovata la corrispondenza e posizionerà il cursore sulla prima riga. Per spostarvi alla riga successiva (non importa in quale file), utilizzate il comando ":cnext". Per andare alla corrispondenza precedente, utilizzate il comando ":cprev". Utilizzate ":clist" per visualizzare tutte le corrispondenze ed il punto in cui si trovano.
   Il comando ":grep" utilizza i comandi esterni di grep (su Unix) o di findstr (su Windows). Potete modificare ciò impostando l'opzione 'grepprg'.

==

Capitolo seguente: |usr_20.txt| Immissione rapida dei comandi sulla linea di comando

Segnalare refusi a Bartolomeo Ravera - E-mail: barrav at libero.it
          oppure ad Antonio Colombo - E-mail: azc100 at gmail.com