

Guida a VIM

Abstract

Il VIM (Vi Improved - Vi Migliorato) è uno degli editor dei sistemi unix-like per eccellenza.

Questa guida permette a un utente alle prime armi di poter iniziare ad utilizzare il VIM nella maniera più produttiva ed ad essere da subito operativo.

Data di stesura: 03/11/2003

Data di pubblicazione: 15/12/2003

Ultima modifica: 15/12/2003

di **Davide Coppola**

VIM?

Per chi non lo sapesse VIM è un potente editor di testo per linux (anche se esistono versioni per la maggior parte dei SO), ispirato a vi, uno degli editor di testo di base dei sistemi unix-like, sui quali è installato ormai di default, comunque, chi volesse scaricarlo una nuova versione o volesse installarlo in quanto sprovvisto, può benissimo rivolgersi al sito ufficiale^[1].

La vera differenza tra un normale editor di testo (tipo notepad per intenderci) e il VIM è che quest'ultimo permette l'esecuzione di svariati comandi, la possibilità di: creare comandi personalizzati, utilizzare script, gestire finestre di testo multiple. Inoltre aggiunge l'assistenza fornita ai programmatori grazie all'opzione di indentazione (le varie spaziature che si inseriscono all'interno di un sorgente per renderlo più leggibile) automatica e alla presenza di schemi di colori che rendono i codici più facilmente leggibili.

Oltre tutto quello già elencato si può dire che il VIM è stato pensato e realizzato per tutti quegli utenti "avanzati" che fanno ampio uso di editor di testo e che odiano dover spostare le mani dalla tastiera, quindi anche se ad un principiante potrebbe dare inizialmente un'impressione di elevata difficoltà consiglio a tutti di imparare ad usarlo alla perfezione in quanto è uno strumento indispensabile per la programmazione.

In questa guida saranno analizzati (quasi) tutti i comandi utilizzati durante l'editing, per quanto riguarda gli script e le configurazioni avanzate del programma magari scriverò un'altra guida più, se ci sarà voglia e tempo!

Iniziamo

In questo paragrafo descriverò come creare dei file di testo e come l'editing dei file già esistenti.

Per lanciare il VIM basta digitare dalla shell:

```
[m3xican@napolihak m3xican]$ vim
```

Così facendo verrà eseguito il programma.

Se però proverete a scrivere qualcosa vi accorgete che c'è qualcosa di strano ... ovvero che non viene scritto niente! (o peggio!)

Ma non preoccupatevi, è "normalissimo", questo perché il VIM possiede due modalità principali di funzionamento: una "comando" e una "inserimento".

Quando lanciate il programma partite sempre dalla modalità comando, ovvero quella modalità in cui i tasti corrispondono all'esecuzione dei vari comandi disponibili, se invece volete scrivere qualcosa dovete passare in modalità inserimento digitando, ad esempio, il comando:

```
i
```

Adesso potete scrivere quello che volete proprio come se stesste in un banalissimo editor di testo.

Dopo ore e ore di scrittura (spero per voi sensata...) arriverà il momento in cui vorrete salvare il vostro lavoro, niente di più semplice! Basta tornare in modalità comando premendo il tasto:

```
ESC
```

Adesso possiamo accedere ad una terza modalità, che è una particolare modalità comando detta "modalità ultima linea" a cui si accede digitando ":" quando ci trova in modalità comando.

Adesso vedrete apparire il simbolo ":" all'inizio dell'ultima riga dell'editor, e il cursore posizionarsi dopo di esso. Questa modalità può essere considerata come una sorta di shell del VIM, qui infatti, per eseguire un comando è necessario premere INVIO dopo averlo digitato.

Adesso finalmente potete salvare il vostro file digitando:

```
:w nome_file
```

oppure

```
:sav nome_file
```

In entrambi i casi viene creato un file con il nome da voi indicato che contiene tutto quello che avete scritto fino a quel momento.

Se oltre a salvare volete anche uscire dal VIM non dovete fare altro che digitare ":q".

Una volta premuto INVIO ci si ritrova nella shell di partenza.

Un ottimo sistema per ottimizzare il tutto è quello di abbinare i comandi (caratteristica utilissima del VIM), in pratica scrivendo:

```
:wq nome_file
```

Salviamo il nostro nuovo file e usciamo.

Quando poi abbiamo bisogno di tornare a lavorare sul nostro documento non dobbiamo fare altro che digitare dalla shell:

```
[m3xican@napolihak m3xican]$ vim nome_file
```

Verrà riavviato il VIM e caricato il file indicato come parametro sotto shell.

Ora possiamo riprendere la stesura del nostro testo, però questa volta, quando avremo bisogno di salvare basterà passare in modalità comando e digitare `":w"`, mentre quando avremo intenzione di salvare il nostro file con un nome diverso basterà digitare `":sav nuovo_nome"` (sempre dalla modalità comando), tenendo ben presente che tutte le modifiche che verranno effettuate da quel momento in poi sul testo, una volta salvate, andranno applicate al nuovo file appena creato. In pratica `"w"` è il corrispettivo di `"salva"` e `"sav"` è il corrispettivo di `"salva con nome"`.

Poiché a tutti capita di sbagliare... nel caso abbiamo inavvertitamente modificato un file che non dovevamo toccare possiamo sempre uscire senza salvare.

Per fare ciò è necessario utilizzare l'operatore `":q!"` che forza i comandi ed ignora i messaggi d'errore, in pratica bisogna digitare quanto segue:

```
:q!
```

Una volta tornati alla shell potrete verificare che il file non ha subito nessuna modifica visualizzandolo con un `cat`, riaprendolo con VIM o verificando la data dell'ultima modifica con `ls`.

Come già vi ho detto il VIM è pensato per persone che non amano perdere molto tempo, soprattutto nella gestione dei comandi, una scorciatoia alla procedura di salvataggio e uscita spiegata poco prima è il comando:

```
ZZ
```

Tale comando salva il file (che deve essere un file già esistente) ed esce.

In pratica si passa dalla sequenza:

```
ESC  
:  
wq  
INVIO
```

alla sequenza:

```
ESC
```

```
zz
```

non male vero?

Muoviamoci

Fino ad ora avete imparato ad usare il VIM come un semplice editor di testo, è giunto il momento di iniziare ad usarlo in maniera più efficace.

Ricordatevi di essere sempre in modalità comando quando eseguite i comandi che sto per spiegare.

I primi comandi da conoscere per potersi spostare all'interno di un file in editing con VIM sono i seguenti:

```
      k
h      l
      j
```

L'ordine in cui li ho rappresentati non è affatto casuale, infatti essi sono usati al posto delle frecce per muoversi sul testo:

"h" sposta il cursore a sinistra, "j" in basso, "k" in alto e "l", indovinate un po' ... a destra.

Perchè usare questi tasti al posto delle frecce?

Semplice, per non spostare le mani dalla zona centrale della tastiera! (Inoltre, storicamente parlando, non tutti i terminali possedevano lo stesso layout e le frecce cursore erano in certi casi un "lusso", ndLM)

I comandi appena descritti permettono di muoverci di un carattere alla volta, se invece abbiamo intenzione di spostarci di 'n' caratteri non dobbiamo fare altro che sfruttare un'altra importantissima caratteristica del VIM, ovvero la possibilità di far eseguire un comando quante volte vogliamo.

Per poter ottenere ciò basta anteporre al comando il numero 'n' di volte che vogliamo che esso venga ripetuto, quindi se un semplice "l" ci sposta di un carattere a destra

```
nl
```

dove 'n' è un numero, ci sposta di 'n' caratteri a destra.

Questa utile caratteristica può essere utilizzata con quasi tutti i comandi che verranno illustrati in questo paragrafo, pertanto segnalerò solo i comandi che non possono utilizzarla, lasciandola sottintesa per tutti gli altri.

Alle volte può essere scomodo spostarsi di un carattere alla volta, questo soprattutto quando ci si deve posizionare alla fine della riga che si sta modificando.

Un comando che ci permette di effettuare uno spostamento più ampio è:

```
w
```

Che non farà nient'altro che spostare il cursore sul primo carattere della successiva parola. Ricordate che tutti i segni di punteggiatura e i caratteri speciali sono considerati parole.

Se invece vogliamo spostarci sulla parola precedente è necessario digitare il comando:

b

Ricordate che anche questo comando sposta il cursore sempre sul primo carattere della parola.

Se invece vogliamo spostarci di una parola in avanti, però sull'ultimo carattere bisogna digitare il comando:

e

Mentre per spostarsi sull'ultimo carattere della parola precedente:

ge

Da notare che quando ci si trova nel mezzo di una parola i comandi "b" ed "e" spostano il cursore rispettivamente sul primo e sull'ultimo carattere della parola stessa e che si spostano su di una parola successiva o precedente solo se il cursore si trova sul primo o sull'ultimo carattere della parola corrente.

Questi comandi permettono di spostarci lungo le parole, ma esistono anche dei comandi per spostarsi lungo le righe.

Per spostarci alla fine della riga in cui è presente il cursore è necessario utilizzare il comando:

\$

Mentre per spostarsi sul primo carattere non di spaziatura di una riga si utilizza il comando:

^

Questo comando ignora qualsiasi numero gli sia anteposto.

Se invece ci si vuole spostare all'inizio di una riga, indipendentemente dal fatto che il primo carattere sia un carattere di spaziatura, si deve utilizzare il comando:

0

Ovvero il numero zero.

Anteporre un numero a questo comando annulla il comando stesso, in pratica non si ottiene alcun effetto.

Quando si edita un testo si ha spesso la necessita di portarsi su di una riga in particolare, soprattutto quando si programma (dopo un errore segnalato dal compilatore ad

esempio...).

Per venire incontro a quest'esigenza il VIM mette a disposizione alcuni comandi molto utili. Spesso è utile visualizzare la fine di un documento, per spostarsi sull'ultima riga di un file si usa il comando:

G

Anteponendo un numero 'n' a questo comando ci si posizionerà sulla riga n-esima. Un altro modo per spostarsi all'n-esima riga è quello di utilizzare il comando in modalità ultima riga:

:n

dove 'n' è il numero della riga in cui ci si vuole posizionare.

Digitando ":" si ottiene lo stesso effetto del comando "G".

Mentre per spostarsi sulla prima riga c'è il comando:

gg

Se invece ci si vuole spostare ad una certa percentuale del testo bisogna utilizzare il comando:

%n

Dove la 'n' è la percentuale del testo che vogliamo raggiungere.

Alle volte è anche utile spostarsi all'interno della finestra momentaneamente aperta, per permetterci movimenti relativi al suo interno il VIM ci permette tre comandi:

H

posiziona il cursore sulla prima riga della finestra aperta

M

posiziona il cursore sulla riga mediana della finestra aperta

L

posiziona il cursore sull'ultima riga della finestra aperta.

Pertanto se abbiamo aperto una finestra che contiene 9 righe l'esecuzione del comando "H" ci posizionerà sulla prima, il comando "M" sulla quinta e il comando "L" sulla nona.

Spesso si ha necessità di focalizzare quello che si legge su una riga ben precisa, ad esempio per avere una visione migliore di una parte di codice di un programma.

Anche in questo caso sono presenti tre comandi dedicati a spostamenti della schermata relativi ad una riga:

```
zz
```

centra la schermata sulla riga in cui è presente il cursore

```
zt
```

fa diventare la riga in cui è presente il cursore la prima riga della schermata

```
zb
```

fa diventare la riga in cui è presente il cursore l'ultima riga della schermata

Un'altra potente caratteristica del VIM è la possibilità di marcare le righe in modo da poterle raggiungere facilmente in seguito.

Per marcare una riga è necessario eseguire il comando:

```
mx
```

Dove 'x' è la lettera che marcherà la riga.

Per poter tornare su quella riga basterà utilizzare il comando:

```
`x
```

dove 'x' è la lettera che marca la riga che vogliamo raggiungere.

Un utile comando da modalità ultima riga è:

```
:marks
```

che mostra tutte le linee marcate e i corrispettivi caratteri associati.

Molto importanti sono anche i comandi dedicati allo scrolling, che permettono di spostarsi rapidamente all'interno del documento ed effettuare rapidi controlli.

Le operazioni di scrolling nel VIM non si limitano solo a quelle dei normali editor di testo (page down, page up), ma permettono un controllo maggiore degli scroll, andiamo a vedere come:

```
CTRL-b
```

in luogo di page up e

```
CTRL-f
```

in luogo di page down

sono i classici comandi di scroll, che permettono di spostarsi di una schermata.

CTRL -u

e

CTRL -d

si spostano di mezza schermata, rispettivamente verso sopra il primo e verso sotto il secondo.

In conclusione (di questo paragrafo...) voglio segnalare un comando utile durante la programmazione:

%

questo comando se eseguito su di una parentesi si posiziona sulla sua corrispettiva (di apertura o chiusura, a seconda di dove ci si era posizionati prima del comando).

Scriviamo

Editing

Abbiamo imparato (almeno spero) come muoverci sul nostro documento, penso che ora sia arrivato il momento di sapere come editarlo nel migliore dei modi.

Il comando principale per iniziare a scrivere è quello che abbiamo già visto nel 2° paragrafo di questa guida, ovvero "i". Tale comando ci "sposta" in modalità inserimento; dopo la pressione del tasto "i" possiamo iniziare a scrivere a partire dalla posizione che precede il cursore.

Se invece vogliamo iniziare a scrivere dalla posizione che segue quella corrente dobbiamo digitare il comando:

a

Tale comando sposterà il cursore di una casella a destra e ci porterà in modalità inserimento.

Alle volte si ha bisogno di aggiungere delle cose alla fine di una riga, e nel VIM non poteva mancare un comando per effettuare questa azione:

A

Tale comando porterà il cursore a destra dell'ultimo carattere non nullo della riga su cui ci si trova al momento della sua esecuzione e ci porterà in modalità inserimento.

Un altro utile comando è quello che ci permette di andare a scrivere dall'inizio della riga ovvero:

|

Un'altra coppia di comandi molto importanti è quella che ci permette di aggiungere una riga al testo che stiamo editando, con:

o

Si aggiunge una riga sotto quella su cui è presente il cursore al momento del comando, mentre:

O

Aggiunge una riga sopra quella corrente.

Sebbene l'aggiunta di una riga sotto quella corrente può essere effettuata anche premendo INVIO il comando "O" risulta di indubbia utilità in quanto ci permette di evitare un procedimento lungo per aggiungere una riga sopra quella che stiamo editando (ovvero sali su, posizionati alla fine della riga, premi INVIO).

Dopo l'esecuzione di entrambi i comandi ci si ritrova in modalità inserimento con il cursore posto all'inizio della riga appena creata.

Cancellazione

Durante la stesura dei nostri capolavori ... ci capiterà spesso di sbagliare, il VIM non può non venirci incontro anche questa volta. Il comando base per la cancellazione è:

x

Questo comando cancella il carattere sottostante il cursore, eseguendolo più volte si cancellano più caratteri. Digitando un numero 'n' prima di esso si cancellano 'n' caratteri a partire dalla posizione del cursore.

Invece il comando:

X

cancella il carattere posto a sinistra del cursore.

Per effettuare cancellazioni più "complesse" abbiamo a disposizione il comando "d" che di solito va abbinato ai comandi di posizionamento per ottenere l'effetto desiderato.

Se ad esempio vogliamo cancellare una parola abbiamo a disposizione il comando:

dw

Tale comando accetta un numero posto tra le due lettere che lo compongono, in quanto si va ad indicare di quante parole ci si vuole spostare; pertanto per cancellare 'n' parole dobbiamo digitare il comando:

```
dnw
```

Dove 'n' è il numero di parole che vogliamo cancellare.

Lascio al lettore la sperimentazione delle varie combinazioni "d + comando_di_spostamento".

Per poter cancellare un'intera riga si ha a disposizione il comando:

```
dd
```

che elimina la riga su cui è posizionato il cursore, anche in questo caso antepo-
nendo un numero 'n' al comando si cancellano 'n' righe.

Se invece vogliamo cancellare tutto quello che segue il cursore sulla riga in cui esso è
presente dobbiamo eseguire il comando:

```
D
```

Andando ad anteporre un numero 'n' al comando si cancellano 'n-1' righe a partire da
quella su cui ci si trova, ovvero si cancella tutto quello che segue il cursore sulla riga in cui
ci si trova e poi 'n-1' righe seguenti.

I comandi per la cancellazione appena citati lasciano il VIM in modalità comando.

Se invece abbiamo bisogno di cancellare qualcosa e poi trovarci in modalità inserimento
abbiamo bisogno di eseguire dei corrispettivi, ovvero:

```
c
```

(c seguito da uno spazio), al posto di `d`

```
cc
```

(due 'c' seguite da uno spazio) al posto di `dd`

```
C
```

(una C seguita da uno spazio) al posto di `D`.

Questi comandi hanno lo stesso effetto dei loro corrispettivi, ma in più restituiscono la
modalità inserimento.

Alle volte ci si può sbagliare anche quando si effettua una cancellazione, per recuperare il
testo eliminato abbiamo a disposizione il comando:

```
u
```

che in pratica è il comando di undo, ovvero quel comando che annulla l'ultimo comando eseguito.

Quando si è proprio recidivi nel commettere errori, si può dover anche annullare l'annullamento di un comando (uno scioglilingua!) in tal caso abbiamo a disposizione il comando per il redo:

CTRL -r

In pratica questi due comandi sono quelli rappresentati dalle frecce "avanti" e "indietro" nei vari programmi di grafica o di editing testi.

Ricordate che questi due comandi non agiscono soltanto con i comandi di cancellazione, ma bensì con ogni comando eseguito.

Un altro utile comando che può essere associato alla coppia undo-redo è il comando:

.

che ripete l'ultimo comando effettuato.

Modalità visuale

Un'altra modalità molto utile è la modalità visuale.

Tale modalità ci permette di selezionare un'area di testo su cui eseguire un comando.

Dopotutto alle volte è molto più comodo selezionare quello che si vuole cancellare mentre si legge un rigo che leggere il rigo, contare le parole da cancellare, posizionarsi sulla prima ed eseguire "dnw"! Comunque *de gustibus* ... ognuno fa come meglio crede!

Allora, per passare in modalità visuale bisogna utilizzare il comando:

v

Una volta che ci si trova in modalità visuale ci si può muovere sul testo come meglio si crede, spostandosi con i tasti direzione o con i comandi, gli spostamenti saranno sempre abbinati all'evidenziazione del testo.

Quando abbiamo terminato di selezionare l'area che ci interessa non dobbiamo fare altro che smettere di spostarci ed eseguire il comando che desideriamo.

Durante la selezione possiamo avere il bisogno di spostarci all'inizio dell'area evidenziata, per invertire la posizione del cursore abbiamo a disposizione il comando:

o

che sposta il cursore nel punto da cui abbiamo iniziato a selezionare, permettendoci poi di muoverci per selezionare all'indietro o per deselezionare il selezionato.

Infatti per deselezionare un'area selezionata è sufficiente ripassarci sopra con il cursore.

Quando abbiamo bisogno di selezionare righe intere possiamo usare il comando:

```
|v
```

spostandoci verso l'alto o il basso possiamo selezionare/deselezionare tutte le righe che vogliamo e poi eseguire su di esse un comando, ad esempio una cancellazione o una copia.

Come abbiamo già visto fino ad ora il VIM è pieno di risorse, e così nella modalità visuale è presente anche un supporto per le tabelle (selezioni rettangolari).

Eseguendo il comando:

```
CTRL -v
```

si passa in un particolare tipo di modalità visuale detta "visuale blocco" che permette di muoversi lungo le tabelle per selezionare gli elementi.

Con la modalità visuale blocco si può continuare ad utilizzare il comando "o" per spostarsi da un estremo all'altro della riga selezionata e in più si può utilizzare il comando:

```
O
```

per spostarsi diagonalmente all'interno dell'area selezionata.

È importante notare che i comandi "o" ed "O" eseguiti in modalità visuale non sono gli stessi comandi eseguiti in modalità comando, questo proprio grazie alle due differenti modalità.

Taglia, copia, incolla

Per completare le nostre operazioni di editing del testo non possiamo non conoscere i classici comandi di taglia, copia, incolla.

Il comando per copiare il testo è:

```
y
```

Tale comando eseguito nella sua forma base (ovvero digitando solo 'y') copia un solo carattere, e più precisamente il carattere su cui è presente il cursore.

Però il VIM permette di abbinarlo ai comandi di spostamento già visti (come nel caso della cancellazione) e alla modalità visuale.

Quindi ad esempio possiamo eseguire i seguenti comandi:

```
yw
```

che copia fino al carattere iniziale della parola successiva escluso, quindi se si hanno due parole e si è posizionati sulla prima, eseguendo questo comando si copia la prima parola e in più lo spazio che le separa

```
ye
```

che copia una parola, ma non lo spazio che segue

y\$

che copia dalla posizione del cursore fino alla fine della riga, e così via.

Un altro utile comando è

Y

oppure l'equivalente

yy

che copia la riga intera su cui è posizionato il cursore.

Il VIM apparentemente non possiede un comando per il taglio del testo, ma solo apparentemente...

Infatti per eseguire un taglio di testo, ovvero un'eliminazione di una determinata parte di testo e il suo inserimento da qualche altra parte, è sufficiente utilizzare i comandi di cancellazione.

Infatti tutte le volte che cancelliamo una parte del testo, tale parte viene memorizzata in un buffer (in parole povere un contenitore temporaneo) e può essere incollata sul testo in un qualsiasi altro punto.

Per poter incollare il testo che abbiamo memorizzato nel nostro buffer temporaneo (dopo un'operazione di copia o di taglio/cancellazione) abbiamo a disposizione due comandi:

p

incolla il testo prima del cursore, e

P

che incolla il testo dopo il cursore.

Da notare che quando si esegue una copia o una cancellazione di un'intera riga viene memorizzato anche il carattere di a capo, quindi andando ad eseguire il comando 'p' si incolla la riga sotto la riga corrente, ed eseguendo il comando 'P' sopra.

Per evitare di dover cancellare lo spazio che si è creato si può utilizzare la modalità visuale andando a selezionare soltanto il testo.

Ricerche

Un'azione molto ricorrente durante l'editing di un testo è la ricerca di una stringa, ovvero di un insieme di caratteri. Quando si ha a che fare con documenti composti da centinaia di righe è praticamente impossibile sperare di effettuare questa ricerca "visivamente" e bisogna affidarsi a programmi (ad esempio "grep" sotto linux) e comandi di ricerca contenuti negli editor.

Il VIM presenta due tipi di ricerca, la ricerca per carattere e la ricerca per stringa.

Ricerca per carattere

il comando da utilizzare quando si vuole trovare la prima occorrenza di un carattere 'x' a partire dalla posizione del cursore è:

```
fx
```

per proseguire nella ricerca, ovvero per cercare una successiva occorrenza del carattere appena trovato, si usa il comando:

```
;
```

Se invece si vuole cercare la prima occorrenza del carattere 'x' che precede la posizione del cursore (ovvero se si vuole cercare all'indietro), bisogna usare il comando:

```
Fx
```

in questo caso per proseguire con la ricerca a ritroso bisogna utilizzare il comando:

```
,
```

Una coppia di comandi simile a quella appena descritta è quella costituita dal comando

```
tx
```

e dal comando

```
Tx
```

che in pratica svolgono la stessa funzione dei comandi "f" e "F", ma spostano il cursore non sul carattere trovato, ma bensì sul carattere che lo precede.

Ricerca per stringa

Quando abbiamo bisogno di trovare una determinata stringa di testo all'interno di un file dobbiamo effettuare una ricerca con il comando:

```
/stringa
```

dove "stringa" è la parola (o le parole) che vogliamo cercare, la ricerca avverrà dal cursore fino alla fine del testo.

Se invece vogliamo effettuare una ricerca a ritroso, ovvero dal cursore fino all'inizio del testo, dobbiamo utilizzare il comando:

```
?stringa
```

Un importante aiuto alle nostre ricerche è dato dal carattere '.' che ha la stessa funzione del '?' sotto shell (ad esempio il comando 'ls cas?' elenca tutti i file contenuti nella directory corrente che iniziano per 'cas' e che sono costituiti da quattro caratteri). Il carattere '.' infatti sostituisce a se stesso tutti i possibili caratteri, facendoci così trovare tutte le possibili varianti di una parola in base ai nostri bisogni. Ad esempio il comando

```
/cas.
```

eseguito prima della frase "Sono andato a casa per caso" segnerà come trovate (evidenzierà) le parole "casa" e "caso".

Il carattere '.' non è l'unico mezzo che abbiamo per specializzare le nostre ricerche, sono presenti anche altri operatori che aiutano il nostro compito.

Infatti la semplice operazione di ricerca trova le occorrenze di una stringa all'interno del testo, il che significa che alle volte è possibile ottenere come risultato della ricerca di una determinata parola una parte di un'altra parola (ad es.: ricercando "ama" troviamo "ama"to).

Il VIM ci viene incontro grazie alla sintassi:

```
/\
```

che effettua una ricerca proprio della stringa "parola" (quindi ricercando "ama" non ci viene evidenziato "amato").

Ma questa non è l'unica sintassi permessaci, infatti abbiamo anche la possibilità di cercare tutte le parole che terminano con la stringa "stringa" eseguendo il comando:

```
/parola\>
```

se invece abbiamo bisogno di cercare una stringa posta soltanto all'inizio di una riga possiamo utilizzare la sintassi:

```
/^parola
```

mentre per cercare una stringa posta alla fine di una riga possiamo utilizzare:

```
/parola$
```

Da notare che i caratteri *[]^%/\~\$ sono considerati caratteri speciali e pertanto necessitano del carattere '\' prima di essi all'interno della stringa. Ovvero se vogliamo cercare la stringa:

```
[come stai?]
```

bisogna usare la sintassi seguente:

```
/\[come stai?\]
```

Due altri importanti comandi sono:

*

che effettua una ricerca nel testo della parola su cui è presente il cursore al momento dell'esecuzione del comando e:

#

simile al precedente, ma effettua una ricerca a ritroso.

Non appena eseguiamo un comando di ricerca (ovvero non appena premiamo INVIO) il cursore viene posizionato sulla prima occorrenza della stringa trovata, se vogliamo spostarci su una successiva occorrenza dobbiamo eseguire il comando:

n

per spostarci invece su di un'occorrenza precedente dobbiamo utilizzare il comando:

N

Sostituzione

Sostituzione manuale

Un gruppo di comandi molto utili in fase di correzione o cambiamento di un testo è quello relativo alle sostituzioni.

In questo paragrafo sono presentati i vari comandi utilizzati per sostituzioni manuali, ovvero per quelle sostituzioni eseguite manualmente da chi edita il testo.

Il più semplice comando di sostituzione è quello per sostituire un solo carattere:

r

tale comando deve essere eseguito posizionando il cursore sul carattere che vogliamo sostituire e poi digitando il nuovo carattere.

Se ad esempio abbiamo scritto la parola "casa" e la vogliamo modificare nella parola "case" dobbiamo posizionarci sull'ultima lettera, digitare 'r' e poi digitare 'è'.

Se invece vogliamo sostituire un numero arbitrario di caratteri dobbiamo utilizzare il comando:

R

Tale comando ci sposterà in una nuova modalità, che possiamo chiamare di sostituzione. Una volta terminate le nostre sostituzioni dobbiamo digitare ESC per tornare in modalità comando.

Un ulteriore comando che può essere associato alle operazioni di modifica è:

```
~
```

ovvero una "tilde" (su tastiera italiana ottenibile con la combinazione di tasti AltGr+ì) seguita da uno spazio

eseguendo questo comando quando ci si trova con il cursore su di un carattere lo si cambia da maiuscolo a minuscolo e viceversa. Per eseguire questo comando su parole intere o gruppi di parole è sufficiente selezionare il testo da convertire in modalità visuale e poi eseguire il comando.

Sostituzione automatica

Alle volte eseguire sostituzioni manuali può diventare un'impresa impossibile, soprattutto se la sostituzione da effettuare è frequente e il testo molto vasto.

Per questo motivo il VIM ci viene incontro con un utilissimo comando per la sostituzione automatica.

In pratica noi non abbiamo altro da fare che specificare al comando la stringa che deve essere sostituita (s1) , la stringa che deve prendere il suo posto (s2) e un area (A) in cui effettuare la sostituzione, per ottenere la sostituzione di tutte le occorrenze di s1 con s2 all'interno di A.

Il comando di sostituzione di base è un comando da modalità ultima linea così definito:

```
:s/str1/str2
```

dove "s" è il comando vero e proprio, "str1" è la stringa da sostituire e "str2" è la stringa che deve prendere il posto di "str1".

Questo comando andrà a sostituire alla prima occorrenza di "str1", nella riga in cui è presente il cursore al momento di eseguire il comando, la stringa "str2".

Se invece vogliamo sostituire tutte le occorrenze di "str1" su di una riga dobbiamo aggiungere un'opzione al comando, ovvero:

```
:s/str1/str2/g
```

dove "g" è l'opzione appena citata ('g' sta per globale).

Con il comando di sostituzione "s" è possibile effettuare alcune azioni abbastanza complesse non presenti nei vari comandi di sostituzione di casa MS (che pagate cari... il VIM è free! ;).

Ad esempio possiamo andare a cancellare una stringa eseguendo il comando:

```
:s/str1//
```

che in pratica è una sostituzione di "str1" con una stringa vuota.

Oppure il comando:

```
:s/^/str2
```

che permette di andare ad aggiungere all'inizio della riga la stringa "str2" e il comando:

```
:s/$/str2
```

per andare ad aggiungere alla fine della riga la stringa "str2".

se invece vogliamo andare ad concatenare "str2" a "str1" dobbiamo usare l'operatore & nel modo seguente:

```
:s/str1/&str2
```

il risultato di questa operazione sarà un'unica stringa formata dall'unione di "str1" e "str2", ovvero "str1str2".

Una maggiore potenzialità del comando "s" è fornita dalla possibilità di specificare l'area di testo su cui devono essere eseguite le sostituzioni. Per fare ciò è necessario utilizzare il comando "s" con una sintassi leggermente diversa ovvero:

```
:a,bs/str1/str2
```

dove 'a' e 'b' sono i due numeri di riga che delimitano l'area di testo in cui vogliamo effettuare le sostituzioni. Ricordate che il comando appena descritto non fa altro che cercare la prima occorrenza di "str1" nelle righe comprese tra 'a' e 'b' e sostituire ad essa "str2".

Se invece abbiamo bisogno di sostituire tutte le occorrenze di "str1" con "str2" all'interno dell'intervallo specificato dobbiamo aggiungere alla fine del comando "/g", come ho mostrato in precedenza.

Similmente, per effettuare una sostituzione di tutte le occorrenze di "str1" all'interno del nostro testo dobbiamo eseguire il comando:

```
:1,$s/str1/str2/g
```

come abbiamo già visto, infatti, il simbolo "\$" viene interpretato dal VIM come l'ultima riga.

Un altro simbolo utilizzato è "." che sta ad indicare la riga corrente.

Questi due simboli possono essere usati per scrivere piccole espressioni che ci consentono di selezionare aree di testo relative ad alcune posizioni prefissate.

Se ad esempio vogliamo eseguire una sostituzione in un'area di testo che parte dalla terza riga che precede la corrente e che finisce due righe prima dell'ultima riga del testo dobbiamo scrivere il comando con la seguente sintassi:

```
:. -3,$-2s/str1/str2
```

Impostazioni

Il VIM ci permette di impostare molte opzioni per rendere più agevole le nostre sessioni di editing.

In genere è possibile attivare un'opzione con il comando eseguito in modalità ultima linea:

```
:set opzione
```

quando invece la si vuole disattivare bisogna eseguire lo stesso comando facendo però precedere al nome dell'opzione la stringa "no".

Eccovi una lista di opzioni utili:

```
:set number
```

visualizza all'inizio di ogni riga il suo numero corrispondente.

```
:set ruler
```

mostra sull'ultima riga informazioni sulla posizione del cursore.

```
:set ignorecase
```

ignora la differenza tra minuscole e maiuscole nelle ricerche.

```
:set hlsearch
```

attiva l'evidenziazione delle stringhe trovate da un'operazione di ricerca.

```
:set incsearch
```

inizia ad effettuare le ricerche durante la digitazione della stringa che si vuole trovare (quindi non aspetta la pressione di INVIO).

```
:set autoindent
```

Attiva l'autoindentazione, utile alla stesura di programmi facilmente leggibili.

```
:set showmode
```

mostra la scritta "-- INSERISCI --" sull'ultima riga quando ci si trova in modalità inserimento, ovvero ci segnala quando ci troviamo in modalità inserimento (opzione molto utile per non eseguire comandi involontariamente).

Siccome non è molto conveniente impostare le opzioni desiderate ogni volta che si usa il VIM esiste un file di configurazione che possiamo editare a nostro piacimento per configurare il programma come meglio ci aggrada.

Il file in questione è ".vimrc" che dovrebbe essere contenuto nella home directory, possiamo verificare la sua posizione grazie al comando:

```
:scriptnames
```

che visualizza il percorso di tutti i file di script del VIM.

Per far eseguire un'opzione all'avvio del programma è sufficiente inserire all'interno di questo file una riga simile alla seguente:

```
set opzione
```

Quando editerete questo file la prima volta, se già esistente, potreste notare la presenza di tutte le opzioni impostate di base, le quali dovrebbero essere anche commentate, per poter annullare i loro effetti è sufficiente eliminare la riga desiderata oppure porre un "no" prima dell'opzione.

Help

La mia guida benché ampia non copre tutti gli aspetti del VIM (che sono tantissimi), e magari quelli che copre non li spiega in maniera appropriata. Chi volesse imparare e/o capirci di più (spero tutti) su questo fantastico programma può consultare la guida on-line eseguendo il comando in modalità ultima riga:

```
:help
```

La guida on-line contiene diversi link colorati in azzurro, tali link possono essere visualizzati posizionandosi con il cursore sopra e digitando "CTRL-]".

Una guida più sintetica può essere visualizzata con il comando:

```
:help index
```

che visualizza la lista di tutti i comandi del VIM (e sono molti...) con una breve descrizione associata.

A tale guida si può accedere anche se si desidera visualizzare l'help per un comando, opzione o codice d'errore ben definito, per far ciò è necessario eseguire il comando:

```
:help stringa
```

dove stringa rappresenta il comando, l'opzione o il codice d'errore sul quale si vogliono delle spiegazioni.

Tabella comandi

Eccovi una breve tabella che riassume i comandi di uso frequente.

Mi raccomando non barate saltando tutta la guida e andando a leggere solo la tabella!

```
Per entrare in modalità inserimento
```

comando	descrizione
---------	-------------

i	prima del cursore
a	dopo il cursore
I	a inizio riga
A	a fine riga
o	riga successiva
O	riga precedente

Spostamento del cursore	
comando	descrizione

k	sopra
l	destra
j	sotto
h	sinistra
\$	fine riga
^	inizio riga
w	parola successiva
b	parola precedente
nG	riga n-esima

Cancellazione	
comando	descrizione

x	carattere selezionato
dw	parola
dd	intera riga
D	fino a fine linea

Vari	
comando	descrizione

u	undo
CTRL -r	redo
.	ripeti comando
y	copia carattere
Y	copia riga
p	incolla sotto
P	incolla sopra
ZZ	scrive e esce
:q	esce
:w	salva

:sav	salva con nome
!	ignora errori

Controllo video

comando	descrizione
---------	-------------

CTRL-d	page down
CTRL-u	page up
CTRL-f	1/2 schermata in avanti
CTRL-b	1/2 schermata indietro
CTRL-l	refresh video

Informazioni sull'autore


Davide Coppola, studente di informatica alla Federico II di Napoli, appassionato di programmazione (C - C++ - Assembly - PHP - Javascript - HTML - SQL - bash), sicurezza e linux. Fondatore e sviluppatore di [NapoliHak](#).

È possibile consultare l'elenco degli [articoli scritti da Davide Coppola](#).

Risorse

1. Il sito ufficiale di VIM; link per il download, news ed altro.
<http://www.vim.org/>
2. Una vasta raccolta di scripts, tips, documentazione.
<http://vim.sf.net/>

- Complete or partial reproduction of this document without authors' permission is strictly forbidden. -
- La riproduzione completa o parziale di questo documento senza il permesso della redazione è proibita. -

SiForge.org © 2002, Stefano Fago, Giovanni Giorgi, Marco Lamberto. Layout & Graphics are © . Made with GIMP & WPP.
http://www.SiForge.org/articles/2003/12/15-guida_vim.html - Revised: 15/12/2003