Breve guida a `nvi`

Antonio Frecentese

Questa vuole essere una guida di base a uno dei *text editor* più utilizzati in ambiente Linux e Unix in genere, nonostante la sua ormai veneranda età: `vi`.

`vi` è un *text editor*, non un *word processor*. Serve quindi per scrivere dei *file* di testo puri e semplici: pagine *web*, *file* di configurazione, *scripts*, sorgenti di programmi e così via. Nei testi scritti con esso non ci sono cose come il grassetto, il corsivo, tipi di caratteri differenti, colori, immagini, tabelle, eccetera eccetera. Questi sono dettagli che riguardano la formattazione del testo, non la sua stesura; se volete tenerli sotto controllo nel momento stesso in cui scrivete, dovete rivolgervi invece a un *word processor* come Microsoft Word (o, meglio ancora, OpenOffice, LibreOffice o Abiword, che sono

gratuiti).

Un'altra cosa importante da sapere è che con vi tutto si può fare da tastiera, senza usare il *mouse*. Questo implica che bisogna trovare un sistema per indicare a vi come distinguere un comando dal testo che si sta scrivendo. Per questo si dice che vi è un *editor modale*, ovvero usa differenti modalità per inserire il testo e per eseguire i comandi. Non è una cosa immediata da imparare, sono il primo ad ammetterlo, ma permette agli esperti di compiere operazioni tipo "cancella le righe dalla 3 alla 5 e al loro posto inserisci la data corrente" in meno tempo che non usando un *word processor*.

Una doverosa precisazione: io uso una precisa variante di questo programma, ovvero nvi, che rispetto all'originale ha il vantaggio di poter leggere e modificare *file* di testo con codifica Unicode (importante per me che devo poter scrivere in giapponese). Questo significa che quanto si trova in questa guida vale anche per il ben più completo gvim (che offre altri comandi e cose in più tra cui un'interfaccia grafica con icone), ma molti comandi e funzioni presenti in quest'ultimo non valgono per nvi e quindi non li troverete.

Buona lettura!

Primi passi

Innanzitutto apriamo il programma. Possiamo farlo digitando `nvi` e premendo [INVIO] in una finestra di terminale, eventualmente fornendo come argomento il nome del *file* di testo da aprire, per esempio `nvi tesi.tex`[INVIO]. Se il *file* specificato non esiste, verrà creato.

A questo punto apro una piccola parentesi per dirvi che `nvi` mantiene il testo digitato in memoria e lo salva solo se gli viene esplicitamente detto di farlo (tra poco vedremo come). Quindi un *file* nuovo non verrà effettivamente creato o modificato finché non si avrà dato a `nvi` il comando "salva".

Non appena aperto `nvi` (con un documento o no), ci si trova nel *command mode*, cioè in "modalità comando", quella che ci consente di eseguire comandi sul documento eventualmente aperto, di caricare documenti in memoria o di salvarli, di uscire dal programma stesso. È possibile tornare a questa modalità in ogni momento premendo il tasto [ESC]. Se ci si trova già in *command mode*, si udirà un *beep* e/o si vedrà un *flash* dello schermo. Questa è una cosa utile quando, ad esempio, per qualche motivo non ci si ricorda in quale modalità ci si trovi: sarà sufficiente premere [ESC] fino al *beep* per essere sicuri di essere tornati al *command mode*.

Cominciamo dunque con lo scrivere qualcosa nel nostro documento. Prima di tutto bisogna entrare nell'*insert mode*, la "modalità di inserimento". In effetti vi sono ben 6 modi per farlo:

- premendo `i` si comincia a inserire del testo prima della posizione del cursore
- premendo `I` ci si posiziona all'inizio della riga su cui si trova il cursore
- premendo `a` si comincia a inserire del testo dopo la posizione del cursore (modalità "*append*")
- premendo `A` ci si posiziona alla fine della riga su cui si trova il cursore
- premendo `o` si aggiunge una linea vuota al di sotto del cursore
- premendo `O` (questa è una "O" maiuscola, non uno "zero") si aggiunge una linea vuota al di sopra del cursore

Ora si può digitare il proprio testo. A proposito: quando si raggiunge il bordo destro della finestra, `nvi` ci fa continuare "a capo", ma in realtà il programma continua a considerarla una riga unica finché non premiamo [INVIO]. Si tratta del cosiddetto *word wrap*. Tenetelo bene a mente!

Quando avete finito di scrivere il vostro testo, immagino abbiate voglia di salvarlo. Bene, allora premete [ESC] per tornare al *command mode*, digitate `:w nomefile` e premete [INVIO]. I due punti sono il carattere che indicano a `nvi` che si sta per digitare un comando, mentre `w` sta per "*write*". Ora il *file* è salvato e potete uscire: digitate `:q` e premete [INVIO].

Se invece dovete salvare il *file* su cui state lavorando sovrascrivendolo (quindi "salva" invece di "salva con nome"), usate semplicemente `:w`.

Sappiate poi che `:wq` salva il file su disco e poi esce (per farlo si può usare anche `:x`), mentre `:q!` esce senza salvare (il punto esclamativo indica che si ha proprio l'intenzione di eseguire il comando).

Ora muoviamoci un pochino...

Scrivendo un testo è possibile spostarsi da una riga all'altra e all'interno della riga corrente tramite i tasti cursore, ma molti consigliano di utilizzare alcuni determinati tasti mentre ci si trova nel *command mode*:

- `h` per andare a sinistra
- `l` per andare a destra
- `j` per andare giù
- `k` per andare su

Questi tasti permetterebbero di spostarsi più rapidamente perché sono posizionati sulla tastiera e non a lato di essa, quindi le dita si trovano già molto vicino ad essi. È utile soprattutto quando si usa magari un piccolo *laptop* che non ha i tasti cursore. Detto tra noi, io non li uso mai, preferisco i tasti cursore.

Vi sono poi altri tasti utili per spostarsi all'interno del testo:

- `0` (questo è uno "zero", non una "O" maiuscola) ci porta all'inizio della riga
- `$` ci porta invece alla fine della riga
- `b` va all'inizio della parola precedente o al precedente segno di punteggiatura (`B` invece salta la punteggiatura)
- `w` va all'inizio della parola seguente o al precedente segno di punteggiatura (`W` invece salta la punteggiatura)
- `e` va alla fine della parola, e se premuto ancora va alla fine della parola seguente

Notate come per utilizzare questi comandi non sia necessario premere [INVIO].

È anche possibile far precedere questi comandi da un numero: per esempio, `5w` ci fa avanzare di 5 parole. Comodo, no?

1. Muoversi più in fretta

Per andare a una riga specifica, basta digitare il numero della riga e premere poi `G`. Quindi `1G` ci porterà all'inizio della prima riga, `24G` all'inizio della ventiquattresima, e così via. Se poi, per esempio, si digita `14G` e il file ha meno di 14 righe, si avrà un *beep*. Per andare all'ultima riga, basta la `G`.

È poi possibile premere

- [CTRL+]`f` (oppure [CTRL+]`F`) per avanzare di un'intera schermata (se siamo già alla fine del file si udirà un *beep*)
- [CTRL+]`b` (oppure [CTRL+]`B`) per indietreggiare di una schermata (se siamo già all'inizio del file si udirà un *beep*)
- [CTRL+]`d` (oppure [CTRL+]`D`) per avanzare di mezza schermata (se siamo già alla fine del file si udirà un *beep*)
- [CTRL+]`u` (oppure [CTRL+]`U`) per indietreggiare di mezza schermata (se siamo già all'inizio del file si udirà un *beep*)

Un'altra cosa utile da sapere è che premendo CTRL+`g` si potrà vedere in basso il nome del file, se è stato modificato dopo l'ultimo salvataggio, su quale riga ci si trova e su un totale di quante righe e infine a che punto del *file* ci si trova (in percentuale).

Al trotto!

1. Scrivi e riscrivi...

Una delle operazioni che si fanno più di frequente con un testo è quella di cancellare del testo immesso per errore. Per farlo si entra in *command mode* e si digita:

- `x` per rimuovere il carattere su cui si trova il cursore; `4x` rimuove 4 caratteri a partire da quello su cui si trova il cursore, e così via
- `X` rimuove il carattere prima di quello su cui si trova il cursore
- `dd` per rimuovere l'intera riga su cui si trova il cursore; `3dd` rimuove 3 righe partendo da quella su cui si trova il cursore, e così via
- `dw` per rimuovere la parola su cui si trova il cursore (per la precisione, la cancella dal punto in cui si trova il cursore fino allo spazio prima della parola seguente, quindi se abbiamo `parola1 parola2 parola3` e il cursore si trova sull'`1` di `parola1`, premendo `dw` avremo `parolaparola2 parola3`); `6dw` rimuove 6 parole, e così via
- `D` per cancellare tutto partendo dal cursore fino alla fine della riga

Ci sono anche alcuni comandi per rimpiazzare il testo errato con quello corretto:

- `r` rimpiazza il carattere su cui si trova il cursore (dopo la `r` si inserisce il carattere giusto); questo comando resta in *command mode*
- anche `R` permette di rimpiazzare un singolo carattere, ma poi si entra in *insert mode* e quindi si continua a scrivere il testo
- `s` rimpiazza il carattere su cui si trova il cursore (dopo la `s` si inserisce il carattere giusto); dopo questo comando si va in *insert mode*
- `S` sostituisce l'intera riga, quindi cancella la riga in cui ci si trova ed entra nell'*insert mode*
- `cw` si usa per cambiare la parola su cui si trova il cursore (si va poi in *insert mode*, quindi è necessario premere [ESC] una volta terminata la modifica affinché essa venga accettata)
- `cc` viene usato per cambiare l'intera riga su cui si trova il cursore (si va poi in *insert mode*)
- `C` viene usato per cambiare il testo da dove si trova il cursore alla fine della linea (si va poi in *insert mode*)

Un'altra funzione utile è il classico "*undo*", che permette di tornare alla situazione precedente l'ultima modifica. Se per esempio avete cancellato una riga che avrebbe dovuto restare dov'era e desiderate reinserirla, è sufficiente premere `u`.

Se a questo punto si preme ancora `u`, verrà annullata l'azione di *undo*.

Il comando `.` ("punto"), invece, ripete l'ultima modifica fatta (quindi se si usa `dd` per cancellare una riga, il "punto" ne cancellerà un'altra).

2. Taglia e cuci

Credo che praticamente tutti quelli che usano un *word processor* (me compreso) amino molto le funzioni di

taglia/copia/incolla, in particolare chi scrive spesso dei testi piuttosto lunghi. Anche `nvi` consente di compiere queste operazioni.

Il comando da usare per copiare parte del testo in memoria è `y` ("*yank*"), da usare in combinazione con i comandi di movimento per indicare cosa copiare in memoria (una parola, fino alla fine della riga, tre righe e così via).

Per inserire, invece, il testo copiato in memoria, ci si sposta sul punto desiderato e si preme `P` per incollare il testo in memoria prima del cursore oppure `p` per inserirlo dopo il cursore. Notate che il testo resta in memoria, quindi lo si può incollare ancora.

Da segnalare poi il comando `J` per unire la riga corrente con quella successiva. Per separare invece due righe, bisogna entrare in *insert mode* e premere [INVIO] nel punto desiderato.

Al galoppo!!

1. Ricerca e sostituzione

Ovviamente una delle cose più utili in un qualunque programma per scrivere testi è la possibilità di cercare una data parola.

- si usa `/` per effettuare la ricerca dal punto in cui si trova il cursore in poi
- si usa `?` per cercare a partire dal cursore verso l'inizio del file

Bisogna prima digitare il comando, poi digitare di seguito la parola o l'espressione da trovare e per finire premere [INVIO] per dare inizio alla ricerca della parola o espressione appena scritta. Per esempio, per cercare la parola "mamma" partendo dal punto in cui si trova il cursore bisognerà digitare `/mamma` e poi premere [INVIO].

Per ripetere la ricerca basta premere ancora `/` (oppure `?`) e subito dopo [INVIO].

Si possono usare anche altri due comandi per ripetere una ricerca:

- `n` ripete la ricerca appena compiuta nella stessa direzione (quindi se la si era fatta con `/` va avanti mentre se la si era fatta con `?` va indietro)
- `N` ripete la ricerca appena compiuta ma nell'altra direzione (quindi se la si era fatta con `/` va indietro mentre se la si era fatta con `?` va avanti)

In effetti quello che si scrive dopo `/` oppure `?` per effettuare una ricerca può essere molto più di una parola. Si entra nel campo delle cosiddette espressioni regolari, ovvero sequenze di caratteri che rispondono a determinati requisiti.

Per esempio, `/cas.` cercherà, partendo dalla posizione attuale del cursore, le occorrenze dei tre caratteri `cas` (in questo ordine e tutti attaccati) seguiti da 1 solo carattere, qualsiasi esso sia; troverà quindi sia “casa” che “caso”.

Per quanto riguarda la sostituzione, si può usare ad esempio il comando `:%s/sbagliato/corretto/` per sostituire (la `s`) in tutto il *file* (il carattere `%`) la parola `sbagliato` con la parola `corretto`.

Al posto di `%` si possono usare altre combinazioni: il carattere “punto” indica la riga corrente, mentre `$` indica la fine del *file*; si può quindi scrivere per esempio:

- `:5,$s/sbagliato/corretto/` per fare la ricerca e sostituzione di prima limitandosi alle righe dalla 5 alla fine del *file*
- `:.,.+20s/sbagliato/corretto/` per cercare e sostituire dalla riga corrente per le successive 20 righe
- possiamo poi indicare a `nvi` che vogliamo confermare ogni singola sostituzione aggiungendo `gc` alla fine, come in `:5,20s/sbagliato/corretto/gc`. A ogni parola `sbagliato` il programma ci chiederà conferma e basterà premere [INVIO] per saltare alla prossima occorrenza del termine, `y` per confermare la sostituzione e `q` per interrompere la ricerca. In questo comando, la `g` di `gc` indica “ogni occorrenza in una data riga”.

Un altro modo per una ricerca e sostituzione globale utilizza il comando `g`: invece di `:%s/sbagliato/corretto/g` si può scrivere anche `:g/sbagliato/s//corretto/g`. Questo ci consente di

usare anche altri parametri in modo da rispettare un dato schema, un *pattern*:

- il carattere “punto” indica un singolo carattere (escluso “a capo”)
- `*` indica che il carattere immediatamente precedente deve essere presente un numero qualunque di volte (anche nessuna); quindi dato che `.` indica “un qualunque carattere”, `.*` indica “ogni carattere, zero o più volte”
- `[...]` indica ognuno dei caratteri tra parentesi: `[AB]` significa “A oppure B”, `[A-Z]` significa “un singolo carattere maiuscolo tra la A e la Z”
- `^` richiede che l’espressione seguente compaia a inizio riga
- `$` richiede che l’espressione seguente compaia a fine riga
- `\<` richiede che l’espressione seguente compaia a inizio parola (quindi `\<ami` trova parole che cominciano con *ami* come *amicizia* ma non trova *lasciami*)
- `\>` richiede che l’espressione seguente compaia a fine parola
- la barra inversa `\` tratta il carattere seguente come carattere ordinario, quindi `\*` per indicare l’asterisco e così via

Tutto questo ci consente di creare comandi anche molto particolari come:

- `:g/^$/d` per cancellare tutte le righe vuote
- `:1,10g/\./s//;/g` consente di sostituire tutti i “punti” delle righe dalla 1 alla 10 con dei “punti e virgola” (notate come il “punto” sia preceduto dalla barra inversa perché qui intendiamo il carattere “punto” vero e proprio)
- `:g/□□*/s//□/g` per sostituire ogni sequenza di uno o più spazi con uno spazio singolo (qui lo spazio è stato mostrato con □)

Per ulteriori cose che potreste fare con questi comandi, è bene imparare l’uso di queste sequenze di caratteri, le stesse delle espressioni regolari che si usano con programmi come

`grep`.

Per i più esperti

1. Altri modi di spostarsi e di modificare il testo

- Un comando simile a `0` (lo “zero”) è `^`, che porta però al primo *non-white-space* della riga
- `H` ci porta alla prima riga presente sullo schermo (non necessariamente quindi la prima del file); *n*`H` ci porta quindi alla *n*-esima riga presente sullo schermo
- `M` ci porta alla riga centrale presente sullo schermo (non quindi quella in mezzo al file)
- `L` ci porta all’ultima riga presente sullo schermo (non necessariamente l’ultima del file); *n*`L` ci porta quindi alla *n*-esima riga a partire dall’ultima presente sullo schermo

È poi interessante notare che se si posiziona il cursore su una parentesi aperta e si preme `%`, si verrà portati alla parentesi chiusa corrispondente (e viceversa). Questo vale sia per le parentesi tonde sia per quelle quadre sia per quelle graffe.

In *command mode* ci può spostare in un dato punto del testo e poi premere `m` seguito da una lettera per creare un *mark*, al quale si può tornare in seguito col comando `‘` seguito dalla lettera che identifica quel dato *mark*.

2. Editare altri file

Può capitare di dover apportare modifiche a un altro *file* mentre si sta già lavorando a un primo testo, per poi dover tornare al primo *file*. Possiamo allora procedere col comando `:e nomefile`[INVIO] (se di dà questo comando senza prima salvare il lavoro, `nvi` ci avvertirà che in questo modo perderemo le modifiche fatte, pertanto non eseguirà il comando). `nvi` ricorderà a che punto ci trovavamo nel primo *file* e aprirà il secondo. Una volta fatte (e salvate!) le modifiche che occorrono, si dà il comando `:e#` per tornare al primo documento.

È possibile poi copiare e incollare del testo da un *file* all'altro in questo modo. Innanzitutto aprire il primo *file* e copiare in memoria (o tagliare) ciò che ci serve; poi bisogna aprire il secondo *file* in una nuova "finestra" col comando `:Edit nome_secondo_file`; recarsi nel punto desiderato e incollare il testo. Notare l'iniziale maiuscola di `:Edit`.

3. Altre funzioni utili

È anche possibile inserire in un dato punto del testo un altro file posizionando il cursore sul punto in cui bisogna inserire il secondo file e dando il comando `:r nomefile`.

Questo comando è utile anche per inserire, nel punto in cui si trova il cursore, l'*output* di un comando da *shell*. Supponiamo che si voglia inserire, per esempio, l'*output* del comando `hostname`: è sufficiente scrivere `:r!hostname` e, quando si preme [INVIO], nel *file* verrà inserito il risultato.

4. Schermo affiancato

Il comando `:vsplit` consente di dividere momentaneamente lo schermo in due, in senso verticale. In questo modo è possibile tenere visualizzata a sinistra una parte del *file* (partendo dal punto in cui si trovava il cursore quando si è dato il comando) e, nella parte destra, continuare a modificarlo. Quando si darà poi il comando `:q`, si ritornerà allo schermo intero.

Si tratta di una funzione utile quando, ad esempio, si ha nelle primissime righe del *file* un breve elenco dei possibili comandi da usare e si vuole tenere sott'occhio tali comandi mentre si modifica il resto del testo.

5. Opzioni di configurazione

Col comando `:set all` si possono visualizzare tutte le opzioni che è possibile impostare per modificare in qualche modo il comportamento di `nvi`.

Per esempio, una funzione molto utile per chi scrive programmi è poter avere i numeri di riga sul margine sinistro dello schermo, in modo da sapere sempre a che punto ci si trova. È possibile impostare la visualizzazione dei numeri di riga con `:set nu` oppure `:set number` (`:set nonu` oppure `:set nonumber` per non visualizzarli).

`:set ic` fa sì che la ricerca con `/` oppure `?` non faccia distinzione tra maiuscole e minuscole.

Io ho impostato anche `:set smd` per far sì che il programma visualizzi in basso a destra la modalità in cui si trova al momento e un asterisco per segnalare che il *file* ha subìto delle modifiche, così come `:set ruler` per fare in modo che in basso al centro venga visualizzato il numero della

riga e del carattere della riga in cui mi trovo in un dato momento.

È possibile poi salvare tali impostazioni nel *file* `~/.exrc`, in modo che venga caricato a ogni avvio del programma.

Ho introdotto qui le funzioni più utili per poter cominciare a usare `nvi`, ma come ho già anticipato c'è ancora molto. Potete scoprire tante altre funzioni con l'*help* in linea di `nvi`, che potete consultare col comando `:viusage`.