Introduzione all'editor vi

Avviare vi

Per avviare vi, digitate semplicemente le lettere `vi` seguite dal nome del file che volete editare o creare.
`vi` parte in modalità comandi: qualsiasi cosa che inserirete verrà interpretata come un comando.
I due comandi di input base sono:

```
i     inserisce del testo a sinistra del cursore
a     aggiunge del testo a destra del cursore
```

Questi due comandi fanno passare vi dalla modalità comando alla modalità inserimento.
Per ritornare alla modalità comando bisogna premere il tasto escape <ESC>.

Tutti i comandi in ***vi*** sono preceduti dalla pressione del tasto **escape <ESC>**.
Ogni volta che si deve intraprendere un nuovo comando si deve utilizzare il tasto di **escape**.
L'editor ***vi*** è case sensitive (sensibile alla differenza minuscolo e maiuscolo).

Comandi di spostamento del cursore sono:

- h sposta il cursore uno spazio a sinistra
- j sposta il cursore uno spazio verso il basso
- k sposta il cursore uno spazio verso l'alto
- l sposta il cursore uno spazio a destra

Comandi di cancellazione testo:

- x cancella il carattere sul cursore
- dd cancella una linea

Salvataggio e uscita:

- **:w salva (scrive su disco)**
- **:q esce**
- :q! esce senza salvare

I principali comandi per vi

Tutti i comandi in ***vi*** sono preceduti dalla pressione del tasto **escape <ESC>**.
Ogni volta che si deve intraprendere un nuovo comando si deve utilizzare il tasto di **escape**.
L'editor ***vi*** è case sensitive (sensibile alla differenza minuscolo e maiuscolo).

Note: **Ctrl** indica il tasto **control,** (*n*) indica un numero ed è opzionale

Comandi movimento cursore:

- **(n) h (*n*) spazi a sinistra**

(n)j **(*n*) righe giù**

(n)k **(*n*) righe su**

(n)l **(*n*) spazi a destra**

(Generalmente funzionano anche i tasti freccia)

Ctrl F	avanti di una schermata
Ctrl B	indietro di una schermata
Ctrl D	giù di mezza schermata
Ctrl U	su di mezza schermata
H	all'inizio della linea superiore della schermata
M	all'inizio della linea mediana della schermata
L	all'inizio dell'ultima linea della schermata
G	all'inizio dell'ultima linea del file
(n)G	**all'inizio della linea (*n*)**
0	**(zero) all'inizio della linea**
$	**alla fine della linea**
(n)w	**avanti (*n*) parole**
(n)b	**indietro (*n*) parole**
e	**fine della parola**

Inserimento testo:

i	**inserimento testo prima del cursore**
a	**aggiunta testo dopo il cursore (non sovrascrive altro testo)**
I	inserimento testo all'inizio della linea
A	aggiunta testo alla fine della linea
r	sostituisce il carattere posto sotto il cursore con il prossimo carattere digitato
R	sovrascrive i caratteri fino alla fine della linea (o fino a quando il tasto **escape** viene digitato per cambiare comando)
o	**o (lettera o minuscola) inserisce una nuova linea dopo la linea corrente per inserire del testo**
O	**O (lettera o maiuscola) inserisce una nuova linea prima della linea corrente per inserire del testo**

Cancellazione testo:

dd	**cancella la linea corrente**

(n)dd	**cancella (*n*) linee**
(n)dw	**cancella (*n*) parole**
D	**cancella dal cursore fino alla fine della linea**
x	**cancella il carattere corrente**
(n)x	**cancella (*n*) caratteri**
X	**cancella il carattere precedente**

Comandi di modifica:

(n)cc	modifica (*n*) caratteri sulla linea fino alla fine della linea (o fino a quando viene digitato il tasto **escape**)
cw	modifica i caratteri di una parola fino alla fine della parola (o fino a quando viene digitato il tasto **escape**)
(n)cw	modifica i caratteri delle prossime (*n*) parole
c$	modifica il testo alla fine della linea
ct(x)	modifica il testo alla lettera (*x*)
C	modifica il testo rimanente sulla linea corrente (fino a quando viene digitato il tasto **escape**)
~	modifica il minuscolo/maiuscolo del carattere corrente
J	**unisce la linea corrente a quella successiva**
u	**annulla l'ultimo comando realizzato sulla linea corrente**
.	ripete l'ultima modifica
s	sostituisce il carattere corrente con il testo digitato
S	sostituisce la linea corrente con il testo digitato
:s	**sostituisce vecchie parole con nuove parole :<linee considerate> s/vecchio/nuovo/g**
&	ripete l'ultimo comando di sostituzione (:s)
(n)yy	**«strappa» (*n*) linee dal buffer**
y(n)w	**«strappa» (*n*) parole dal buffer**
P	**(p minuscola) inserisce il testo eliminato o «strappato» dopo il cursore**
P	**(p maiuscola) inserisce il testo eliminato o «strappato» prima del cursore**

Manipolazione file:

:w (file)	**scrive i cambiamenti nel file specificato (file corrente di default)**
:wq	**scrive i cambiamenti nel file corrente e conclude la sessione**

	di editing (esce)
:w! (file)	**sovrascrive il file (file corrente di default)**
:q	**esce dalla sessione di editing se non sono stati creati cambiamenti**
:q!	**esce dalla sessione di editing e scarta eventuali cambiamenti non salvati**
:n	edita il prossimo file nella lista dell'argomento
:f (nome)	modifica il nome del file corrente in quello specificato
:r (file)	legge il contenuto del file specificato all'interno del corrente editing e alla corrente posizione del cursore (inserisce un file)
:!(comando)	escape di shell
:r!(comando)	inserisce il risultato del comando di shell specificato nella posizione corrente

Una chicca: usare vi come editor binario (esadecimale)

Dopo avere lanciato vim, si puo' utilizzarlo come hexeditor lanciando il comando

:%!xxd to turn vim into a hexeditor

e si puo' tornare alla modalita' testuale normale, usando il comando

:%!xxd -r to go back to normal mode

Un'altra chicca: passare uno script di comandi vi a vi

for name in `find ./ -type f -name "*.[hc]" -exec grep -l 'PrintERROR_andExit(rc' '{}' \; ` ; do **vim -s** ../../VI_COMMAND_SUBSTITITION ${name} ; done

dove il file VI_COMMAND_SUBSTITITION contiene

:%s/PrintERROR_andExit(rc/PrintERROR_andExit(errno/g
:wq

Utilizzare la colorazione di parole chiave: color schemes

Comandi da utilizzare all'interno di vi per abilitare la colorazione di parole

:syntax on	**abilita la colorazione di parole chiave secondo uno schema di colori**
:syntax off	**disabilita la colorazione**
:colorscheme nomeschema	**Stabilisce quale schema utilizzare, il nome**

dello schema e' il nome del file che contiene lo schema
gli schemi di sistema sono contenuti in /usr/share/vim/vim74/syntax/
gli schemi dei singoli utenti sono contenuti nella directory /home/nomeutente/.vim/colors/

Ad esempio, se il file /home/vic/.vim/colors/Insulti.vim contiene la sintassi per riconoscere e colorare gli insulti,
l'utente vic puo' abilitare l'uso di quello schema aprendo vim e lanciando i seguenti comandi di vi

:syntax on
:colorscheme Insulti

`:syntax off` **disabilita la colorazione**

Se il file /home/vic/.vim/colors/Insulti.vic contiene la sintassi per riconoscere e colorare gli insulti,
l'utente vic puo' abilitare l'uso di quello schema aprendo vim e lanciando i seguenti due comandi di vi

:syntax on
:colorscheme Insulti

~~In alternativa, l'utente puo' configurare vi affinche' vi utilizzi automaticamente quello schema. Si potrebbe cioe' inserire nel file .vimrc i due comandi (indicati qui sopra) da lanciare all'apertura di vim. In alcune versioni NON FUNZIONA !!!!!~~

Il file dello schema Insulti.vim potrebbe essere fatto cosi':

```
" Vim syntax file
" Language:     Insulti
" Maintainer:   Vittorio Ghini
" Last Change:  2016 Apr 12

"Quit when a (custom) syntax file was already loaded
" if exists("b:current_syntax")
"   finish
" endif

let s:cpo_save = &cpo           " salva l'attuale impostazione di compatbilita' di vim
set cpo&vim                     " resetta l'impostazione di vim portandola al default per vim

" A bunch of useful keywords
syn keyword     Insulti1        Cretino Fesso Idiota
```

```
syn keyword     Insulti2        Rincoglionito Coglione Testadicazzo

syn match       Insulti3        '[pP][oO][rR][cC][oO]'
syn match       Insulti3        '[mM][eE][rR][dD][aA]'

hi def link Insulti1    Error
hi def link Insulti2    Statement
hi def link Insulti3    String

let b:current_syntax = "Insulti"

let &cpo = s:cpo_save
unlet s:cpo_save
" vim: ts=4
```