usr_22.txt   Per Vim version 7.0. Ultima modifica: 2006 Apr 24

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

Trovare il file da aprire

I file possono venir trovati ovunque. Così, come fare a trovarli? Vim offre vari modi per esplorare l'albero delle directory. Ci sono comandi per saltare ad un file che è menzionato in un altro. E Vim ricorda quali file siano stati modificati in precedenza.

  Capitolo seguente: |usr_23.txt| Modifica di altri file
Capitolo precedente: |usr_21.txt| Andarsene e ritornare
             Indice: |usr_toc.txt|

==

22.1 Il file explorer

Vim ha un plugin che rende possibile visualizzare una directory. Provate questo: >

```
:edit .
```

Tramite la magia dei comandi automatici e degli script di Vim, la finestra verrà riempita con i contenuti della directory. Apparirà come questa:

```
" Press ? for keyboard shortcuts ~
" Sorted by name (.bak,~,.o,.h,.info,.swp,.obj,.orig,.rej at end of list) ~
"= /home/mool/vim/vim6/runtime/doc/ ~
../ ~
check/ ~
Makefile ~
autocmd.txt ~
change.txt ~
eval.txt~ ~
filetype.txt~ ~
help.txt.info ~
```

Potrete vedere queste voci:
1. Un commento su come usare ? per ricevere aiuto per le funzionalità
    del file explorer.
2. La seconda linea spiega come sono elencati i contenuti della directory.
    Essi possono essere ordinati in diversi modi.
3. La terza linea è il nome della directory corrente.
4. La voce "../" directory. Questa è la directory genitore.
5. I nomi delle directory.
6. I nomi dei file ordinari. Come rammentato nella seconda linea,
    alcuni non sono qui ma "alla fine dell'elenco".
7. I nomi dei file meno ordinari. Si suppone che non li usiate spesso,
    quindi sono stati spostati alla fine.

Se l'evidenziazione della sintassi è abilitata, le diverse parti sono messe in evidenza per poterle individuare più facilmente.

Potete usare i comandi di Vim in Normal mode per muovervi nel testo. Ad esempio, spostatevi su un file e premete <Invio>. Adesso state scrivendo su quel file. Per tornare indietro all'explorer usate di nuovo ":edit .".
  Funziona anche CTRL-O.
   Provate a usare <Invio> mentre il cursore è sul nome di una directory.
  Il risultato è che l'explorer si sposta in quella directory e mostra i suoi contenuti. Premere <Invio> sulla prima directory "../" vi sposta al livello superiore. Premere "-" fa la stessa cosa, senza il bisogno di muovere prima il cursore su "../".

Potete premere ? per avere un piccolo aiuto sulle cose che potete fare nell'explorer.
Questo è ciò che otterrete:

```
" <enter> : open file or directory ~
" o : open new window for file/directory ~
" O : open file in previously visited window ~
```

```
" p : preview the file ~
" i : toggle size/date listing ~
" s : select sort field    r : reverse sort ~
" - : go up one level      c : cd to this dir ~
" R : rename file          D : delete file ~
" :help file-explorer for detailed help ~
```

I primi pochi comandi sono per selezionare il file da vedere. A seconda del comando che userete, il file comparirà da qualche parte:

<Enter>	Usa la finestra corrente.
o	Apre una nuova finestra.
O	Usa la finestra visitata in precedenza.
p	Usa la finestra di anteprima (preview window) e sposta il cursore nella finestra dell'explorer. |preview-window|

I seguenti comandi sono usati per mostrare altre informazioni:

i	Mostra le dimensioni e la data dei file. Usando i di nuovo le informazioni verranno nascoste.
s	Usa il campo in cui si trova il cursore per ordinare. Prima visualizzate le dimensioni e la data con i. Poi spostate il cursore sulla dimensione di un qualsiasi file e premete s. I file adesso saranno ordinati per dimensione. Premete s mentre il cursore si trova su una data e le voci saranno ordinate per data.
r	Inverte l'ordine (sia la dimensione che la data)

Ci sono altri pochi comandi extra:

c	Imposta la directory corrente sulla directory mostrata. Potete poi battere un comando ":edit" per uno dei file senza anteporre il percorso.
R	Rinomina il file sotto il cursore. Vi sarà chiesto il nuovo nome.
D	Elimina il file sotto il cursore. Vi sarà chiesto di confermare questa azione.

==========

22.2 La directory corrente

Proprio come la shell, Vim ha il concetto di directory corrente. Supponete che voi siate nella vostra home directory e vogliate aprire alcuni file nella directory "VeryLongFileName". Potete fare: **>**

```
:edit VeryLongFileName/file1.txt
:edit VeryLongFileName/file2.txt
:edit VeryLongFileName/file3.txt
```

Per evitare troppe battiture, fate questo: **>**

```
:cd VeryLongFileName
:edit file1.txt
:edit file2.txt
:edit file3.txt
```

Il comando ":cd" cambia la directory corrente. Potete vedere quale sia la directory corrente con il comando ":pwd" : **>**

```
:pwd
/home/Bram/VeryLongFileName
```

Vim ricorda l'ultima directory che avete usato. Usate "cd -" per ritornarvi. Esempio: **>**

```
:pwd
/home/Bram/VeryLongFileName
:cd /etc
:pwd
/etc
:cd -
:pwd
/home/Bram/VeryLongFileName
:cd -
:pwd
/etc
```

LA FINESTRA DELLA DIRECTORY LOCALE

Quando dividete una finestra, entrambe le finestre useranno la stessa directory corrente. Quando volete modificare un certo numero di file da qualche altra parte nella nuova finestra, potete far sì che essa usi un'altra directory, senza cambiare la directory corrente nell'altra finestra. Questa si chiama directory locale. **>**

```
:pwd
/home/Bram/VeryLongFileName
:split
:lcd /etc
:pwd
/etc
CTRL-W w
:pwd
/home/Bram/VeryLongFileName
```

Fintanto che il comando ":lcd" non sia stato usato, tutte le finestre condivideranno la stessa directory corrente. Eseguire un comando ":cd" in una finestra cambierà la directory corrente anche nell'altra finestra.
Per una finestra dove sia stato usato il comando ":lcd" verrà ricordata una directory corrente differente. Usare ":cd" o ":lcd" in altre finestre non la cambierà.
Quando si usa il comando ":cd" in una finestra posta in una diversa directory corrente, farà tornare ad usare la directory condivisa.

==========

22.3 Trovare un file

State scrivendo un programma in linguaggio C che contiene questa linea:

```
#include "inits.h" ~
```

Volete vedere cosa c'è in quel file "inits.h". Muovete il cursore sul nome del file e battete: **>**

```
gf
```

Vim troverà il file e ne mostrerà il contenuto.
Cosa succede se il file non è nella directory corrente? Vim userà l'opzione 'path' per trovare il file. Questa opzione è un elenco di nomi di directory nelle quali cercare il vostro file.
Supponete che i vostri file include siano in "c:/prog/include". Questo comando la aggiungerà alla opzione 'path': **>**

```
:set path+=c:/prog/include
```

Questa directory si trova in un percorso assoluto. Non importa dove voi siate, sarà lo stesso posto. Cosa fare se avete collocato dei file in una subdirectory, al di sotto di dov'è il file? Potete specificare un percorso relativo. Questo inizia con un punto: **>**

```
:set path+=./proto
```

Questo dice a Vim di cercare nella directory "proto", sotto la directory dove si trova il file nel quale avete usato "gf". Così, usare "gf" su "inits.h" farà sì che Vim cerchi "proto/inits.h", iniziando nella directory del file.
Senza "./", quindi "proto", Vim dovrebbe cercare nella directory "proto" sotto la directory corrente. La directory corrente, però, potrebbe non essere quella dove il file che state modificando è collocato.

L'opzione 'path' permette di specificare le directory dove cercare i file in molti più modi. Leggete l'aiuto per l'opzione 'path'.
L'opzione 'isfnamè è usata per decidere quali caratteri sono inclusi nel nome del file e quali non lo sono (es., il carattere " nell'esempio in alto).

Quando conoscete il nome del file, ma non deve essere trovato nel file, potete battere questo: **>**

```
:find inits.h
```

Vim userà quindi l'opzione 'path' per trovare il file. Questa è la stessa cosa del comando ":edit", eccetto che per l'uso di 'path'.

Per aprire il file trovato in una nuova finestra usate CTRL-W f anziché "gf",

oppure usate ":sfind" al posto di ":find".

Un bel modo per avviare direttamente Vim per aprire un file che sia ovunque nel 'path' è: >

```
vim "+find stdio.h"
```

Ciò trova "stdio.h" nel vostro valore di 'path'. I doppi apici sono necessari per avere un solo argomento |-+c|.

==========

22.4 La lista dei buffer

L'editor Vim usa il buffer del terminale per descrivere un file che si sta aprendo. In realtà, il buffer è una copia del file sul quale voi state lavorando. Quando avrete finito di modificare il buffer, voi scriverete i contenuti del buffer nel file. I buffer non contengono solo i contenuti del file, ma anche tutti i segnalibri, le impostazioni e le altre cose che lo accompagnano.

NASCONDERE I BUFFER

Supponete che voi stiate lavorando sul file one.txt e abbiate bisogno di passare al file two.txt.
Potreste usare semplicemente ":edit two.txt", ma poiché avete fatto delle modifiche in one.txt, quel comando non funzionerà. Inoltre non volete ancora salvare one.txt. Vim ha una soluzione per voi: >

```
:hide edit two.txt
```

Il buffer "one.txt" scompare dallo schermo, ma Vim sa ancora che voi state lavorando su questo buffer, così conserva il testo modificato. Questo viene chiamato buffer nascosto: il buffer contiene il testo, ma voi non potete vederlo.
L'argomento del comando ":hide" è un altro comando. Fa sì che quel comando appaia come se l'opzione 'hidden' fosse impostata. Potete anche voi stessi impostare questa opzione. L'effetto è che quando un buffer viene lasciato, esso diventa nascosto.
State attenti! Quando avete nascosto buffer con cambiamenti, non chiudete Vim senza essere sicuri di avere salvato tutti i buffer.

INATTIVARE I BUFFER

Quando un buffer è stato usato una volta, Vim ricorda alcune informazioni su di esso. Quando non appare in una finestra e non è nascosto, esso è ancora nella lista dei buffer. Questo viene chiamato buffer inattivo. Panoramica:

Attivo	Appare in una finestra, testo caricato.
Nascosto	Non è in una finestra, testo caricato.
Inattivo	Non è in una finestra, testo non caricato.

I buffer inattivi vengono memorizzati, poiché Vim conserva le informazioni che li riguardano, come i segnalibri. Ed anche il ricordare il nome del file è utile, affinché voi possiate vedere su quali file avete lavorato. Ed aprirli ancora.

ELENCARE I BUFFERS

Esaminate la lista dei buffer con questo comando: >

```
:buffers
```

Un comando che fa la stessa cosa, non così ovvio per elencare i buffer, ma molto più corto da battere, è: >

```
:ls
```

L'output potrebbe apparire come questo:

```
  1 #h  "help.txt"                    line 62 ~
  2 %a+ "usr_21.txt"                  line 1 ~
  3     "usr_toc.txt"                 line 1 ~
```

La prima colonna contiene il numero del buffer. Potete usare questo per

ritornare ad un buffer senza doverne battere il nome, guardate sotto.
   Dopo il numero del buffer vengono i flag. Quindi segue il nome del file ed il numero della linea dove era il cursore l'ultima volta.
   I flag che possono apparire sono questi (da sinistra a destra):

u	Il buffer è unlisted (non in lista) |unlisted-buffer|.
%	Buffer corrente.
#	Buffer alternativo.
a	Il buffer è caricato e mostrato.
h	Il buffer è caricato ma nascosto.
=	Il buffer è di sola lettura (read-only).
-	Il buffer non è modificabile, l'opzione 'modifiable' è inattiva.
+	Il buffer è stato modificato.

RITORNARE AD UN BUFFER

Potete ritornare ad un buffer tramite il suo numero. Questo evita di dover scrivere il nome del file : **>**

```
:buffer 2
```

Ma il solo modo di sapere il numero è guardare nell'elenco dei buffer. Potete, invece, usare il nome, o parte di esso: **>**

```
:buffer help
```

Vim troverà la migliore corrispondenza per il nome che avete battuto. Se c'è un solo buffer che corrisponde al nome, esso verrà usato. In questo caso "help.txt".
   Per aprire un buffer in una nuova finestra: **>**

```
:sbuffer 3
```

Funziona anche con il nome ugualmente bene.

USARE LA LISTA DEI BUFFER

Potete muovervi nella lista dei buffer con questi comandi:

```
:bnext          va al buffer successivo
:bprevious      va al buffer precedente
:bfirst         va al primo buffer
:blast          va all'ultimo buffer
```

Per rimuovere un buffer dalla lista, usate questo comando: **>**

```
:bdelete 3
```

Ancora, ciò funziona anche con un nome.
   Se cancellate un buffer che era attivo (visibile in una finestra), quella finestra verrà chiusa. Se cancellate il buffer corrente, la finestra corrente verrà chiusa. Se fosse l'ultima finestra, Vim cercherebbe un altro buffer da modificare. Voi non potete non avere nulla di aperto!

Note:
Anche dopo aver rimosso il buffer con ":bdelete", Vim lo ricorderà. In realtà esso è reso "unlisted" e non compare più nella lista di ":buffers". Il comando ":buffers!" elencherà i buffer unlisted (sì, Vim può fare l'impossibile). Perché Vim dimentichi veramente un buffer, usate ":bwipe". Vedete anche l'opzione 'buflisted'.

==

Capitolo seguente: |usr_23.txt| Modifica di altri file

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