usr_23.txt   Per Vim version 6.1. Ultima modifica: 2001 Set 03

VIM USER MANUAL - di Bram Moolenaar
Traduzione di questo capitolo: Cristian Rigamonti

Modifica di altri file

Questo capitolo tratta l'elaborazione di file diversi da quelli ordinari.
Con Vim potete elaborare file compressi o cifrati, file a cui si accede solo
via internet e, con alcune restrizioni, file binari.

  Capitolo seguente: |usr_24.txt| Inserzione rapida
Capitolo precedente: |usr_22.txt| Trovare il file da aprire
            Indice: |usr_toc.txt|

==
23.1 File DOS, Mac e Unix

Ai vecchi tempi, le macchine telescriventi usavano due caratteri per iniziare
una nuova linea: uno per far ritornare il carrello alla prima posizione
(carriage return, <CR>), un altro per far scorrere la carta (line feed, <LF>).
   Con l'arrivo dei computer, lo spazio di immagazzinamento divenne costoso e
qualcuno decise che non c'era bisogno di due caratteri per l'interruzione di
linea. I programmatori UNIX decisero di usare solo <Line Feed> per
l'interruzione di linea, i programmatori Apple si accordarono per <CR>, quelli
MS-DOS (e Microsoft Windows) decisero di tenere il vecchio <CR><LF>.
   Questo significa che se provate a trasferire un file da un sistema
all'altro, incontrate dei problemi nelle interruzioni di linea. L'editor Vim
riconosce automaticamente i diversi formati di file e si occupa di gestirli in
modo trasparente.
   L'opzione 'fileformats' contiene i vari formati che verranno provati quando
viene elaborato un nuovo file. Il comando seguente, ad esempio, dice a Vim di
provare per primo il formato UNIX e poi quello MS-DOS: >

```
:set fileformats=unix,dos
```

Riconoscerete il formato dal messaggio che otterrete quando aprite un file. Se
state usando il formato nativo non vedrete alcun messaggio, questo succede ad
esempio elaborando un file Unix su Unix; se invece aprite un file dos, Vim vi
avvertirà:

```
"/tmp/test" [dos] 3L, 71C ~
```

Per un file Mac vedreste "[mac]".
   Il formato del file riconosciuto sarà immagazzinato nell'opzione
'fileformat'. Per vedere il formato corrente, eseguite il seguente comando: >

```
:set fileformat?
```

I tre nomi usati da Vim sono:

```
unix            <LF>
dos             <CR><LF>
mac             <CR>
```

USARE IL FORMATO MAC

Su Unix, <LF> è usato per interrompere una linea. Non è insolito avere un
carattere <CR> nel mezzo di una linea. Incidentalmente, ciò capita abbastanza
spesso negli script di Vi (e Vim).
   Sul Macintosh, dove <CR> è il carattere di interruzione di linea, è
possibile incontrare un carattere <LF> all'interno di una linea.
   Il risultato è che non è possibile essere sicuri al 100% del fatto che un
file che contiene sia caratteri <CR> che <LF> sia un file Mac o Unix. Vim
assume quindi che su Unix probabilmente non si useranno file Mac e non
controllerà per questo tipo di file. Per controllare comunque anche per
questo tipo di file, aggiungete "mac" a 'fileformats': >

```
:set fileformats+=mac
```

Allora Vim darà uno sguardo al formato del file. Attenzione ai casi in cui Vim veda sbagliato.

FORZARE IL FORMATO

Se usate il buon vecchio Vi e provate a elaborare un file in formato MS-DOS, vi accorgerete che ogni linea finisce con un carattere ^M (^M corrisponde a <CR>).
Il riconoscimento automatico di Vim evita questa situazione. Ma supponiamo che invece vogliate elaborare il file nel suo formato originario: dovete forzare il formato: **>**

```
:edit ++ff=unix file.txt
```

La stringa "++" avvisa Vim che il valore dell'opzione specificata prevarrà sul valore predefinito limitatamente al comando che segue. "++ff" è usato al posto di 'fileformat'; potete anche usare "++ff=mac" o "++ff=dos".
   Questo meccanismo non funziona per tutte le opzioni: al momento sono implementate solo "++ff" e "++enc". Si possono usare anche i nomi completi "++fileformat" e "++encoding".

CONVERSIONE

Potete usare l'opzione 'fileformat' per convertire un file da un formato all'altro. Supponete, ad esempio, di avere un file MS-DOS chiamato README.TXT, che volete convertire in formato UNIX. Iniziate a elaborare il file in formato MS-DOS: **>**

```
vim README.TXT
```

Vim riconoscerà che questo è un file in formato dos. Ora cambiate il formato in UNIX: **>**

```
:set fileformat=unix
:write
```

Il file viene scritto in formato Unix.

==
23.2 File su internet

Qualcuno vi spedisce un messaggio e-mail che fa riferimento ad un file tramite la sua URL. Ad esempio:

```
Trovi le informazioni qui: ~
        ftp://ftp.vim.org/pub/vim/README ~
```

Potreste avviare un programma per scaricare il file, salvarlo sul vostro disco locale ed allora avviare Vim per elaborarlo.
   C'è un modo più semplice. Spostate il cursore su un qualsiasi carattere dell'URL. Poi usate questo comando: **>**

```
gf
```

Con un pò di fortuna, Vim troverà quale programma usare per scaricare il file, scaricarlo ed aprirne la copia. Per aprire il file in una nuova finestra usate CTRL-W f.
   Se qualcosa andasse storto, riceverete un messaggio di errore. E' possibile che l'URL sia sbagliata, che non abbiate il permesso di leggerlo, che la connessione di rete non sia attiva, ecc. Purtroppo è difficile stabilire la causa dell'errore. Dovrete tentare il modo manuale di scaricare il file.

L'accesso dei file via internet funziona col plugin netrw. Al momento sono riconosciuti gli URL con questi formati:

```
ftp://          usa ftp
rcp://          usa rcp
scp://          usa scp
http://         usa wget (sola lettura)
```

Vim non fa lui stesso la comunicazione, ricorre ai programmi menzionati che sono installati sul vostro computer. Sulla maggior parte dei sistemi Unix "ftp" e "rcp" saranno presenti; "scp" e "wget" probabilmente dovranno venire installati.

Vim riconosce queste URL per ciascun comando che inizi ad aprire un nuovo file, anche, ad esempio, con ":edit" e ":split". Funziona anche nei comandi di scrittura, eccetto per http://.

Per maggiori informazioni, anche a proposito delle password, si veda |netrw|.

==
23.3 Cifratura

Alcune informazioni preferite tenerle per voi stessi. Ad esempio, quando state scrivendo una prova d'esame su un computer usato anche dagli studenti. Non volete che i più furbi trovino un modo per leggere le domande prima dell'inizio dell'esame. Vim può cifrare il file, dandovi qualche protezione.
Per iniziare a scrivere un nuovo file con la cifratura, usate l'argomento "-x" per avviare Vim. Esempio: >

```
vim -x exam.txt
```

Vim vi chiede una chiave usata per cifrare e decifrare il file:

```
Enter encryption key: ~
```

Accuratamente scrivete la chiave segreta: non vedrete i caratteri che digitate, saranno sostituiti da asterischi. Per evitare che un errore di battitura causi problemi, Vim vi chiede di immettere ancora la chiave:

```
Enter same key again: ~
```

Ora potete elaborare il file normalmente e scriverci tutti i vostri segreti. Quando avete finito e dite a Vim di uscire, il file viene cifrato e scritto.
Quando aprite il file con Vim, vi chiederà di immettere nuovamente la stessa chiave. Non avrete bisogno di usare l'argomento "-x". Potete anche usare il normale comando ":edit". Vim aggiunge una stringa magica al file, dalla quale riconosce che il file è stato cifrato.
Se tentate di vedere il file con un altro programma, tutto ciò che ottenete è spazzatura. Così, se aprite il file con Vim e immettete la chiave sbagliata, ottenete spazzatura. Vim non ha un meccanismo per controllare se la chiave sia quella giusta (ciò rende molto più difficile tentare di indovinare la chiave).

ATTIVARE E DISATTIVARE LA CIFRATURA

Per disabilitare la cifratura di un file, impostate l'opzione 'key' a una stringa vuota: >

```
:set key=
```

La prossima volta che scriverete il file, verrà fatto senza cifratura.
Impostare l'opzione 'key' per abilitare la cifratura non è una buona idea, visto che la chiave apparirebbe in chiaro. Chiunque fosse dietro di voi potrebbe leggere la chiave.
Per evitare questo problema è stato introdotto il comando ":X". Vi chiede una chiave di cifratura, proprio come fa l'argomento "-x": >

```
:X
Enter encryption key: ******
Enter same key again: ******
```

LIMITI DELLA CIFRATURA

L'algoritmo di cifratura usato da Vim è debole. È abbastanza valido per tenere alla larga il ficcanaso occasionale, ma non lo è abbastanza per un esperto crittologo con molto tempo a disposizione. Dovete anche tenere presente che il file di swap non è cifrato, quindi mentre elaborate un file, gli utenti con privilegi di amministratore possono leggere il testo in chiaro da questo file.
Un modo per evitare che la gente legga il vostro file di swap è quello di non usarne uno. Aggiungendo l'argomento -n alla linea di comando, non viene usato alcun file di swap: Vim terrà tutto in memoria. Ad esempio per elaborare il file cifrato "file.txt" senza usare il file di swap, usate il comando seguente: >

```
vim -x -n file.txt
```

Quando state elaborando un file, il file di swap può esere disabilitato con: >

```
:setlocal noswapfile
```

Poichè manca il file di swap, il recupero diverrà impossibile. Salvate il file un pò più spesso del solito per evitare il rischio di perdere le modifiche effettuate.

Sino a quando è in memoria, il file è in chiaro. Qualunque utente privilegiato può guardare nella memoria dell'editor e scoprire i contenuti del file.
   Se usate un file viminfo, fate attenzione che i contenuti dei registri di testo sono scritti in chiaro.
   Se davvero volete proteggere i contenuti di un file, elaboratelo solo su un computer portatile non connesso in rete, usate buoni strumenti crittografici e tenete il computer chiuso in un posto sicuro quando non lo usate.

==
23.4 File binari

Potete aprire file binari con Vim. Poiché Vim non è davvero fatto per questo, ci sono alcune restrizioni. Ma potete leggere un file, cambiare un carattere e riscriverlo, col risultato che solo quel carattere sarà stato modificato, mentre il file è identico altrove.
   Per essere sicuri che Vim non usi qualcuno dei suoi trucchi nel modo sbagliato, aggiungete l'argomento "-b" all'avvio: **>**

```
vim -b datafile
```

Ciò imposta l'opzione 'binary', che ha l'effetto di disattivare effetti collaterali inaspettati. Ad esempio, 'textwidth' è impostato a zero per evitare la formattazione automatica delle linee. Ed i file vengono comunque letti in formato Unix.

Il Binary mode può essere usato per modificare un messaggio in un programma. Fate attenzione a non inserire o cancellare qualche carattere, il programma non funzionerebbe più. Usate "R" per entrare in modalità sostituzione.

Molti caratteri del file non saranno stampabili. Per vederli in formato Hex: **>**

```
:set display=uhex
```

Altrimenti potete usare il comando "ga" per vedere il valore del carattere sotto il cursore. Il risultato, quando il cursore è su un <Esc>, è questo:

```
<^[>  27,  Hex 1b,  Octal 033 ~
```

Potrebbero non esserci molte interruzioni di linea nel file. Per avere una panoramica del file, disabilitate l'opzione 'wrap': **>**

```
:set nowrap
```

POSIZIONE IN BYTE

Per vedere in quale byte del file vi trovate, usate questo comando: **>**

```
g CTRL-G
```

Il risultato è lungo:

```
    Col 9-16 of 9-16; Line 277 of 330; Word 1806 of 2058; Byte 10580 of 12206 ~
```

Gli ultimi due numeri sono la posizione in byte nel file e il numero totale di byte. Viene tenuto conto di come 'fileformat' modifica il numero di byte usati dalle interruzioni di linea.
    Per spostarvi ad un byte specifico del file, usate il comando "go". Ad esempio, per spostarvi al byte 2345: **>**

```
2345go
```

USARE XXD

Un vero editor binario mostra il testo in due modi: così come è ed in formato hex. Potete farlo in Vim convertendo prima il file con il programma "xxd", fornito con Vim.
   Per prima cosa, aprite il file in modalità binaria: **>**

```
vim -b datafile
```

Ora convertite il file in un hex dump usando xxd: **>**

```
:%!xxd
```

Il testo apparirà così:

```
0000000: 1f8b 0808 39d7 173b 0203 7474 002b 4e49  ....9..;..tt.+NI ~
0000010: 4b2c 8660 eb9c ecac c462 eb94 345e 2e30  K,.`.....b..4^.0 ~
0000020: 373b 2731 0b22 0ca6 c1a2 d669 1035 39d9  7;'1."......i.59. ~
```

Ora potete vedere ed modificare il testo a vostro piacimento. Vim tratta l'informazione come testo ordinario. Modificare il codice hex non modifica automaticamente il carattere stampabile, o quant'altro.
Alla fine convertitelo di nuovo con: **>**

```
:%!xxd -r
```

Solo le modifiche alla parte hex vengono usate. Le modifiche alla parte del testo stampabile sulla destra vengono ignorate.

Per maggiori informazioni, consultare la pagina di manuale di xxd.

==========

23.5 File compressi

Questo è facile: potete elaborare un file compresso come qualsiasi altro file. Il plugin "gzip" si occupa di decomprimere il file quando lo aprite. E di comprimerlo quando lo scrivete.
Attualmente sono supportati questi metodi di compressione:

```
.Z      compress
.gz     gzip
.bz2    bzip2
```

Per comprimere e decomprimere, Vim usa i programmi menzionati sopra. Potrebbe essere necessario installarli prima.

==========

Capitolo seguente: |usr_24.txt| Inserzione rapida

Segnalare refusi a Bartolomeo Ravera - E-mail: barrav at libero.it