starting.txt Per **Vim version 8.2.** Ultima modifica: 2020 Feb 04

VIM Manuale di Riferimento di Bram Moolenaar
Traduzione di questo testo: Marco Curreli e Antonio Colombo

Richiamare Vim *starting*

==========

1. Argomenti di Vim *vim-arguments*

Quasi sempre Vim è richiamato per modificare un solo file con il comando

vim nome-file *-vim*

Più in generale, Vim è richiamato con:

vim [opzioni | nome-file] ..

Argomenti che sono opzioni possono essere frammisti ad argomenti che sono nomi di file, e se ne possono dare quanti se ne vogliono. Tuttavia occorre prestare attenzione alle opzioni che richiedono un argomento.

Per la compatibilità con varie versioni di Vi, vedere |**cmdline-arguments**|.

Uno e solo uno dei seguenti cinque modi può essere usato per scegliere come iniziare a editare un file:

-file *---*

nome-file Uno o più nomi di file. Il primo file specificato sarà quello da cui iniziare e verrà letto in un buffer. Il cursore sarà posizionato sulla prima riga del buffer. Per evitare che un file il cui nome inizia con '-' sia interpretato come un'opzione, premettete alla lista "--", p.es.: >

```
vim -- -nome-file
```

< Tutti gli argomenti dopo "--" saranno considerati come nomi di file, nessun'altra opzione o argomento "+comando" può essere inserito.
Per l'utilizzo di apici in MS-Windows, vedere |**win32-quotes**|.

--

\- Questo argomento può indicare due cose, a seconda se si usa oppure no il modo Ex.

Iniziando in modo Normal: >

```
vim -
ex -v -
```

< Si inizia a editare un nuovo buffer, che è riempito con testo letto dallo stdin. Gli eventuali comandi che sarebbero normalmente letti dallo stdin saranno invece letti dallo stderr. Esempio: >

```
find . -name "*.c" -print | vim -
```

< Il buffer sarà marcato come modificato, in modo da far presente che il testo va salvato, al momento dell'uscita dalla sessione di edit. Per evitare ciò, basta mettere le righe seguenti nel file vimrc: >

```
" Non impostare 'modified' leggendo da stdin
au StdinReadPost * set nomodified
```

<

Iniziando in modo Ex: >

```
                        ex -
                        vim -e -
                        exim -
                        vim -E
<               Si inizia a editare in modo Silent.  Vedere |-s-ex|.

                                                        *-t* *-tag*
-t {tag}        Un tag.  Nel file tags si ricerca "tag", e il file associato
                al tag diviene il file corrente, e il comando associato viene
                eseguito.  Questo si usa principalmente per programmi C, nel
                qual caso, "tag" è spesso un nome di funzione.  L'effetto è
                che il file che contiene quella data funzione diventa il file
                corrente e il cursore è posizionato all'inizio della funzione
                (vedere |tags|).

                                                        *-q* *-qf*
-q [file_errori] Modo QuickFix.  Il file di nome [file_errori] è letto, e il
                primo errore è visualizzato.  Vedere |quickfix|.
                Se non si specifica [file_errori], il valore dell'opzione
                'errorfile' è usato come nome di file.  Vedere 'errorfile'
                per il valore di default.

(niente)        Se nessuno dei quattro modi elencati è usato, Vim inizierà a
                editare un nuovo buffer.  Il buffer è inizialmente vuoto e
                non ha un nome di file.


Il modo con cui iniziare può essere specificato richiamando un altro nome,
invece che "vim", il che equivale a specificare le opzioni sotto elencate:
ex      vim -e      Inizia in modo Ex (vedere |Ex-mode|).               *ex*
exim    vim -E      Inizia in modo Ex migliorato (vedere |Ex-mode|).    *exim*
                        (normalmente non installato)
view    vim -R      Inizia in modo sola-lettura (vedere |-R|).          *view*
gvim    vim -g      Inizia la GUI (vedere |gui|).                       *gvim*
gex     vim -eg     Inizia la GUI in modo Ex.                           *gex*
gview   vim -Rg     Inizia la GUI in modo sola-lettura.                 *gview*
rvim    vim -Z      Come "vim", ma in modo Limitato (vedere |-Z|)       *rvim*
rview   vim -RZ     Come "view", ma in modo Limitato.                   *rview*
rgvim   vim -gZ     Come "gvim", ma in modo Limitato.                   *rgvim*
rgview  vim -RgZ    Come "gview", ma in modo Limitato.                  *rgview*
evim    vim -y      Easy Vim: impostato 'insertmode' (vedere |-y|)      *evim*
eview   vim -yR     Come "evim" in modo sola-lettura                    *eview*
vimdiff vim -d      Inizia in modo Diff |diff-mode|
gvimdiff vim -gd    Inizia in modo Diff |diff-mode|

Ulteriori caratteri possono essere aggiunti al nome, e sono ignorati.  Per
esempio, si può immettere "gvim-8" per richiedere la GUI.  Naturalmente, ci
deve essere un programma eseguibile con quel nome.

In Unix, normalmente esiste un solo file eseguibile di nome Vim, con dei
link (legami simbolici) assegnati ai differenti nomi utilizzabili, e che
puntano tutti allo stesso file eseguibile.  Se il vostro sistema non
supporta i link, e non desiderate tenere molte copie dello stesso
programma eseguibile, si possono assegnare degli alias.  Per esempio: >
        alias view   vim -R
        alias gvim   vim -g
<
                                                        *startup-options*
Gli argomenti che sono opzioni possono essere dati in qualsiasi ordine.
Opzioni che consistono in una sola lettera possono essere combinate dopo
un solo trattino ("-").  Non possono essere specificate opzioni dopo aver
specificato l'argomento "--".

In VMS tutti gli argomenti che sono opzioni sono supposti essere scritti
a lettere minuscole, a meno che siano preceduti da una barra.  Quindi
"-R" significa recovery e "-/R" sola-lettura.

--help                                                  *-h* *--help* *-?*
-?
-h              Stampa un messaggio di aiuto ed esce.
                Vedere |info-message| se si vuol salvare il messaggio.

                                                        *--version*
```

--version Stampa l'informazione sulla versione ed esce. L'output è lo stesso del comando |:version|.
Vedere |info-message| se si vuol salvare il messaggio.

--noplugin
--noplugin Omette il caricamento dei plugin. Mette a off l'opzione 'loadplugins'.

Nota L'argomento |-u| è pure utilizzabile per omettere il caricamento dei plugin.

argomento carica:	file vimrc	plugin	defaults.vim ~
(nothing)	sì	sì	sì
-u NONE	no	no	no
-u DEFAULTS	no	no	sì
-u NORC	no	sì	no
--noplugin	sì	no	sì

--startuptime {nome-file} *--startuptime*
Durante l'inizializzazione scrive messaggi con data e ora al file {nome-file}.
Ciò può essere utile per indagare come si spende tempo nel caricare il file .vimrc, i plugin e per aprire il primo file.
Quando {nome-file} esiste già i nuovi messaggi sono aggiunti in fondo al file.
{disponibile solo se compilato con la funzionalità |+startuptime|}.

--literal
--literal Interpreta i nomi di file così come sono scritti, senza espandere i caratteri speciali. Questo non è necessario in Unix, perché Vim riceve sempre i nomi di file letteralmente (è la shell che espande i caratteri speciali).
La specifica vale per tutti i nomi, anche per quelli immessi prima di fornire questo parametro.

-+
+[num] Il cursore sarà posizionato sulla riga "num" del primo file da editare. Se "num" è assente, il cursore sarà posizionato sull'ultima riga del file.

-+/
+/{modello} Il cursore sarà posizionato sulla prima riga che contiene "modello" nel primo dei file da editare (vedere |pattern| per le espressioni di ricerca specificabili). La ricerca inizia dalla posizione del cursore, che può essere la prima riga, oppure la posizione del cursore usata per ultimo, e ricavata dalla lettura di |viminfo|. Per richiedere che una ricerca inizi dalla prima riga, usare "+1 +/modello".

+{comando} *-+c* *-c*
-c {comando} {comando} sarà eseguito dopo che il primo file è stato letto (e dopo che gli autocomandi e le modeline per quel file sono state elaborate). "comando" è interpretato come un comando Ex.
Se il "comando" contiene spazi, va incluso fra doppi apici (questo dipende dalla shell che si sta usando).
Esempio: >

```
vim "+set si" main.c
vim "+find stdio.h"
vim -c "set ff=dos" -c wq mine.mak
```

<

Nota: Si possono usare fino a 10 argomenti "+" o "-c" chiamando Vim. Gli stessi sono eseguiti nell'ordine in cui sono stati immessi. Un argomento "-S" è contato come se fosse un argomento "-c".

--cmd {comando} *--cmd*
{comando} sarà eseguito prima di elaborare qualsiasi file vimrc. A parte questo, il comportamento è lo stesso di -c {comando}. Si possono usare fino a 10 di questi argomenti, in maniera indipendente dai comandi "-c".

-S
-S {file} Il {file} [di comandi Vim] sarà elaborato dopo che il primo

file è stato letto. Questa è una maniera semplice per dare il comando equivalente: >
```
-c "source {file}"
```
< Si può dare insieme ad argomenti "-c" e si può ripetere come "-c". Il limite di 10 argomenti "-c" vale anche qui. {file} non può essere un nome che inizia con un "-".

Non usare questo metodo per eseguire uno script che, dopo aver fatto qualche lavoro, esca da Vim. I messaggi di errore non verrebbero visualizzati. Usare invece |**-u**|.

-S Equivale a specificare "-S Session.vim". Ma solamente quando è usato come ultimo argomento o quando è seguito da un'altra opzione che inizia con "-".

-r

-r Modo Recovery. Senza un argomento che sia un nome di file, viene elencata una lista di file di swap esistenti. Con un nome di file, un file di swap viene letto per ripristinare una sessione di editing finita male. Vedere |**crash-recovery**|.

-L

-L Come -r.

-R

-R modo Sola-lettura. L'opzione 'readonly' sarà impostata per tutti i file da editare. Si può comunque modificare il buffer, ma la sovrascrittura accidentale di un file sarà impedita. Se ci si dimentica di essere in modo View (sola lettura) e si fa qualche modifica, si può sovrascrivere il file aggiungendo un punto esclamativo al comando Ex, come in ":w!". L'opzione 'readonly' può essere annullata con ":set noro" (vedere il capitolo sulle opzioni, |**opzioni**|).
Le modifiche successive non verranno fatte in modo di sola lettura. Chiamare il modulo eseguibile "view" ha lo stesso effetto che specificare l'argomento -R.
L'opzione 'updatecount' sarà impostata a 10000, il che implica che il file di swap non sarà aggiornato automaticamente molto spesso.
Vedere |**-M**| per impedire modifiche.

-m

-m Eventuali modifiche non vanno scritte. L'opzione 'write' sarà messa a off, in modo da inibire la riscrittura dei file. Tuttavia l'opzione 'write' può essere cambiata in modo da consentire nuovamente la scrittura.

-M

-M Non sono consentite modifiche ai file. L'opzione 'modifiable' sarà messa a off, in modo da non consentire modifiche. Tuttavia le opzioni 'write' e 'modifiable' possono essere cambiate in modo da consentire nuovamente la modificazione e la riscrittura dei file.

-Z *restricted-mode* *E145* *E981*

-Z Modo Limitato. Tutti i comandi che utilizzano una shell esterna sono inibiti. Questo include la sospensione della sessione di edit con CTRL-Z, ":sh", i comandi di filtro, la funzione system(), la specifica di file sulla riga di comando invocando il sistema operativo con (`) e libcall(). Inoltre non è consentito usare delete(), rename(), mkdir(), writefile(), job_start(), etc.
Le interfacce, quali quelle verso Python, Ruby e Lua, sono pure inibite, poiché potrebbero essere usate per eseguire comandi della shell. L'interfaccio con Perl fa uso del modulo Safe (controlli sicurezza).
Va notato che l'utente potrebbe ancora trovare una maniera contorta di eseguire un comando della shell, gli è stata solo resa la vita più difficile.

-g

-g Esegue Vim in modo GUI. Vedere |**gui**|. Per il caso opposto vedere |**-v**|.

-v

-v Esegue Ex in modo Vi. Serve specificarlo solo quando il programma eseguibile si chiama "ex" o "gvim". Per gvim la GUI non è attivata, anche se sarebbe possibile farlo.

-e

-e Esegue Vim in modo Ex |**Q**|. Serve specificarlo solo quando il programma eseguibile non si chiama "ex".

-E

-E Esegue Vim in modo Ex migliorato |**gQ**|. Serve specificarlo solo quando il programma eseguibile non si chiama "exim".

-s-ex

-s Modo Silent o Batch. Solo quando Vim è stato eseguito come "ex" o quando è preceduto dall'argomento "-e". Negli altri casi, vedere |**-s**|, che richiede un argomento mentre questo uso di "-s" non lo richiede.
Da usare quando Vim serve a eseguire comandi Ex da un file invece che da terminale. Non vengono emessi molti dei messaggi di richiesta e di quelli informativi. Come pure gli avvisi e i messaggi di errore.
L'output dei seguenti comandi è visualizzato (in stdout):

```
:print
:list
:number
:set      per visualizzare i valori delle opzioni.
```

Se l'opzione 'verbose' è diversa da 0, i messaggi sono stampati (per permettere la correzione di errori, in stderr). 'term' e $TERM non sono usati.
Se Vim sembra bloccato, si provi a immettere "qa!<Enter>". Un prompt non è previsto, e quindi non è possibile sapere se Vim sta effettivamente aspettando un input dall'utente, se si immette qualcosa.
Le inizializzazioni sono omesse (tranne quelle fornite con l'argomento "-u").
Esempio: >

```
vim -e -s  < un_filtro  un_file
```

< Per il caso opposto, ossia per visuralizzari gli errori generati eseguendo lo script, occorre eseguire il file specificando il flag "-u": >

```
vim -u un_filtro un_file
```

<

-b

-b Modo Binary. La fase di I/O userà solo <NL> [ritorno a capo] per separare le righe. L'opzione 'expandtab' è messa a off. L'opzione 'textwidth' è impostata a 0. 'modeline' è messa a off. L'opzione 'binary' è impostata. Questo avviene dopo la lettura dei file vimrc/exrc ma prima di leggere qualsiasi file presente nella riga degli argomenti.
Vedere anche |**edit-binary**|.

-l

-l Modo Lisp. Imposta a on le opzioni 'lisp' e 'showmatch'.

-A

-A Modo Arabic. Imposta a on l'opzione 'arabic'. {Solo se compilato con le funzionalità |**+arabic**| (incluso |**+rightleft**|); negli altri casi Vim emette un messaggio di errore ed esce}.

-F

-F Quest'opzione serviva per utilizzare il linguaggio Farsi, che non è più supportato. Vedere |**farsi.txt**|.

-H

-H Modo Hebrew. Imposta a on le opzioni 'hkmap' e 'rightleft'. {Solo se compilato con la funzionalità |**+rightleft**|; negli altri casi Vim emette un messaggio di errore ed esce}.

-V *verbose*

-V[N] Verboso. Imposta l'opzione 'verbose' a [N] (default: 10).

Messaggi saranno dati per ogni file letto col comando
":source" e per leggere o scrivere un file viminfo. Può
essere usato per capire quel che succede nelle fasi di
inizio e di fine di una sessione di edit.
Esempio: >
	vim -V8 pippopluto

-V[N]{nome-file}
Come -V e imposta 'verbosefile' a {nome-file}. L'effetto è
che i messaggi non sono visualizzati ma scritti sul file
{nome-file}. Il {nome-file} non deve iniziare con una cifra.
Esempio: >
	vim -V20vimlog pippopluto
<

-D
-D Debugging. Entra in modo Debugging quando si esegue il primo
comando da uno script. |**debug-mode**|
{non disponibile se compilato senza la funzionalità |**+eval**|}

-C
-C Modo Compatible. Imposta l'opzione 'compatible'. Si può
usare per impostare 'compatible', anche se è presente un
file .vimrc.
Va tenuto presente che il comando ":set nocompatible" in
qualche plugin o script iniziale prevale, e che quindi si
può finire per trovarsi comunque in modo 'nocompatible'.
Per controllare, si può usare: >
	:verbose set compatible?
< Molti plugin non funzionano se 'compatible' è impostato.
Si può impostare dopo l'avvio di vim in questo modo: >
	vim "+set cp" nome-file
< Vedere anche |**compatible-default**|.

-N
-N Modo non Compatible. Mette a off l'opzione 'compatible'.
Si può usare per ottenere un comportamento 'nocompatible',
quando non sia presente un file .vimrc o se si usa "-u NONE".
Vedere anche |**compatible-default**|.

-y *easy*
-y Modo Easy (facile). Implicito se si esegue |**evim**| o |**eview**|.
Vim inizia con 'insertmode' impostata e si comporta come un
editor "clicca e scrivi" (p.es. Notepad).
Quest'opzione fa eseguire lo script $VIMRUNTIME/evim.vim.
Le mappature sono impostate in modo da lavorare come quasi
tutti gli editor di tipo "clicca e scrivi", vedere
|**evim-keys**|. La GUI è utilizzata se disponibile.

-n
-n Non usare un file di swap. Il ripristino dopo una caduta
del sistema sarà impossibile. Utile se si vuol visualizzare
o modificare un file contenuto su un supporto molto lento
(p.es., un floppy).
Si può anche ottenere immettendo ":set updatecount=0".
Si può ancora attivare l'uso del file di swap impostando
l'opzione 'updatecount' a qualche numero positivo, p.es.,
":set uc=100".
NOTA: Non aggiungere a -n l'opzione -b, ossia -nb, perché
specificherebbe un'altra opzione dal significato differente:
|**-nb**|.
'updatecount' è impostata a 0 DOPO aver eseguito comandi da
un file vimrc, ma prima delle inizializzazioni della GUI.
Quindi prevale su un'impostazione di 'updatecount' in un file
.vimrc, ma non in un una contenuta in un file .gvimrc.
Vedere |**startup**|. Volendo ridurre gli accessi a disco
(p.es., per un laptop), non si usi "-n", ma si imposti
'updatetime' e 'updatecount' a numeri molto elevati, e si
immetta il comando ":preserve" quando si vuole salvare il
file in elaborazione. In questo modo si ha la possibilità
di ripartire da quel punto dopo un'eventuale caduta di
sistema.

-o

-o[N] Apre N finestre, divise orizzontalmente. Se [N] non è specificato, una finestra è aperta per ogni file dato come argomento. Se lo spazio è limitato, solo i primi file specificati sono visibili in una finestra. Se ci sono più finestre che nomi di file, le finestre in eccesso si apriranno con un file vuoto.

-O
-O[N] Apre N finestre, divise verticalmente. Per il resto è come -o. Se si specificano sia -o che -O, l'ultima opzione che compare sulla riga di comando determina come saranno divise le finestre.

-p
-p[N] Apre N linguette. Se [N] non è specificato, una linguetta è aperta per ogni file dato come argomento. Il numero massimo possibile di linguette è impostato con 'tabpagemax' (default 10). Se ci sono più linguette specificate che nomi di file, le ultime linguette si apriranno con un file vuoto. Vedere anche |**tabpage**|.

-T
-T {terminale} Imposta il tipo di terminale a "terminale". Ciò determina i codici che Vim invia al terminale stesso. Di solito non è necessario impostare quest'opzione, perché Vim è in grado di riconoscere il tipo di terminale che si sta usando. (Vedere |**terminal-info**|.)

--not-a-term
--not-a-term Informa Vim che l'utente è consapevole che l'input e/o l'output non sono diretti a un terminale. Ciò evita l'invio di un messaggio di avvertimento e i relativi due secondi di attesa.
Si evita anche il messaggio "Leggo da 'stdin'...".
Si evita anche il messaggio "N file da elaborare".

--ttyfail
--ttyfail Quando lo stdin o lo stdout non sono un terminale (tty) termina subito l'esecuzione di Vim.

-d
-d Inizia in modo diff, come |**vimdiff**|.
{non disponibile se compilato senza la funzionalità |**+diff**|}

-d {dispositivo} Solo su Amiga e se Vim non è stato compilato con la funzionalità |**+diff**|. Equivale a specificare "-dev".
-dev
-dev {dispositivo} Solo su Amiga: il {dispositivo} è aperto per essere usato per l'editing.
Normalmente si può usare per impostare la posizione e la dimensione della finestra: "-d con:x/y/larghezza/altezza", p.es., "-d con:30/10/600/150". Ma si può anche usare per iniziare a editare su un altro dispositivo, p.es., AUX:.
-f
-f GUI: Non disconnettersi dal programma che ha iniziato Vim. 'f' sta per "foreground" ("in primo piano"). Se omesso, la GUI innesca un nuovo processo, ed esce da quello corrente. "-f" andrebbe usato se gvim è richiamato da un programma che deve attendere la fine della sessione di edit per continuare (p.es., mail o readnews). Se si desidera che gvim non inneschi mai un processo indipendente, si può includere 'f' nelle 'guioptions' contenute in |**gvimrc**|. Attenzione: Si può usare "-gf" per iniziare la GUI all'interno del processo corrente (in "foreground"), ma scrivendo "-fg" si specifica invece il colore del testo visualizzato. |**gui-fork**|

Amiga: Non si deve far ripartire Vim per aprire una nuova finestra. Quest'opzione andrebbe usata quando Vim è richiamato da un programma che deve attendere la fine della sessione di edit per proseguire (p.es., mail o readnews). Vedere |**amiga-window**|.

MS-Windows: Quest'opzione non è supportata. Tuttavia, se si esegue Vim invocandolo tramite gli script installati vim.bat o gvim.bat, funziona.

--nofork

--nofork GUI: Non usare la chiamata di sistema fork per eseguire Vim. Equivalente a |**-f**|.

-u *E282*

-u {vimrc} Il file {vimrc} è letto per inizializzazioni. Molte altre inizializzazioni sono omesse; vedere |**initialization**|.

Questo si può usare per eseguire Vim in una modalità speciale, con mappature e impostazioni particolari. Un alias della shell può essere usato per facilitare questa chiamata. Per esempio: >

```
alias vimc vim -u ~/.c_vimrc !*
```

< Si consideri anche la possibilità di usare autocomandi; vedere |**autocomando**|.

Quando {vimrc} ha il valore "NONE" (a lettere maiuscole), tutte le inizializzazioni da file e variabili d'ambiente sono omesse, compresa la lettura del file |**gvimrc**| quando viene fatta partire la GUI. Anche il caricamento di plugin è omesso. Quando {vimrc} ha il valore "NORC" (a lettere maiuscole), l'effetto è lo stesso che specificare "NONE", ma i plugin vengono caricati.

Quando {vimrc} ha il valore "DEFAULTS" (a lettere maiuscole), l'effetto è lo stesso che specificare "NONE", ma lo script |**defaults.vim**| viene eseguito, che imposta anche l'opzione 'nocompatible'. Vedere anche |**--clean**|.

Usare l'argomento "-u" ha l'effetto secondario di impostare l'opzione 'compatible' per default. Questo può avere effetti indesiderati. Vedere |**'compatible'**|.

-U *E230*

-U {gvimrc} Il file {gvimrc} è letto per effettuare delle inizializzazioni quando si attiva la GUI. Altre inizializzazioni della GUI sono omesse. Quando {gvimrc} ha il valore "NONE", nessun file è letto per l'inizializzazione della GUI. |**gui-init**| Eccezione: La lettura del file del menù a livello di sistema è sempre effettuata.

-i

-i {viminfo} Si usa il file "viminfo" specificato al posto di quello di default. Se il nome indicato è "NONE" (a lettere maiuscole), non viene letto e scritto alcun file , anche se l'opzione 'viminfo' è impostata o se si usano i comandi ":rv" o ":wv". Vedere anche |**viminfo-file**|.

--clean

--clean Simile alla specifica "-u DEFAULTS -U NONE -i NONE":
- le inizializzazioni da file o da variabili d'ambiente vengono saltate
- 'runtimepath' e 'packpath' sono impostate per escludere elementi della home directory (ciò non avviene quando si specifica "-u DEFAULTS").
- lo script |**defaults.vim**| è eseguito, il che implica l'impostazione di 'nocompatible': si usano i valori di default di Vim [non quelli di Vi]
- non viene usato nessun script |**gvimrc**|
- non si legge né si scrive un file viminfo

-x

-x La cifratura è usata nel leggere/scrivere file. Vim richiederà una chiave, che sarà utilizzata per assegnare un valore all'opzione 'key'. Tutte le operazioni di scrittura useranno questa chiave per cifrare il testo. Non è necessario specificare l'argomento '-x' quando si legge un file, perché viene effettuato un controllo

per verificare se il file in lettura era stato cifrato.
e Vim richiede automaticamente di specificare una chiave.
|**encryption**|

-X

-X Non tentare di connettersi a server X per ottenere il titolo della finestra corrente, e per effettuare copy/paste usando le relative funzioni X. Ciò evita un lungo tempo di partenza di Vim se si usa un simulatore di terminale, e se la connessione col server X è lenta.
Vedere |**--startuptime**| per valutare se la cosa ha importanza in qualche caso particolare.
Serve specificarlo solo se si è in Unix o VMS, e se Vim è stato compilato attivando la funzionalità |**+X11**|. Altrimenti viene ignorato.
Per disabilitare la connessione solo per qualche terminale particolare, vedere l'opzione 'clipboard'.
Quando il gestore X11 Session Management Protocol (XSMP) è stato incorporato, l'opzione -X disabilita anche quella connessione, perché anch'essa può generare ritardi indesiderabili.
Se in seguito si vuol comunque attivare la connessione (p.es., per messaggi client-server), si deve chiamare la funzione |**serverlist()**|.
Questo peraltro non abilita il gestore XSMP.

-s

-s {script-in} Lo script file "script-in" è letto. I caratteri nel file sono interpretati come se fossero stati immessi dalla tastiera. Lo stesso si può ottenere col comando ":source! {script-in}". Se la fine del file è raggiunta prima che la sessione di edit termini, l'input ulteriore è letto dalla tastiera. Ciò è possibile solo se non si è invocato Vim in modo Ex, vedere |**-s-ex**|. Vedere anche |**complex-repeat**|.

-w_nr

-w {numero}
-w{numero} Imposta l'opzione 'finestra' a {numero}.

-w

-w {script-out} Tutti i caratteri immessi da tastiera vengono registrai nel file "script-out", fino alla fine della sessione di Vim. Questo può tornare utile se si desidera creare uno script file da usare con "vim -s" o ":source!". Quando il file "script-out" esiste già, i nuovi caratteri sono aggiunti a fine file. Vedere anche |**complex-repeat**|.
Il nome del file {script-out} non può iniziare con una cifra.

-W

-W {script-out} Come -w, ma, invece di aggiungere alla fine di un file esistente, lo sovrascrive.

--remote [+{comando}] {file} ...
Apre il {file} in un altro Vim che funge da server.
Ogni argomento diverso da nomi di file deve essere specificato prima di questo. Vedere |**--remote**|.

--remote-silent [+{comando}] {file} ...
Come --remote, ma non emette messaggi se non viene trovato un server. Vedere |**--remote-silent**|.

--remote-wait [+{comando}] {file} ...
Come --remote, ma aspetta che il server abbia finito la sessione di edit del/dei file.
Vedere |**--remote-wait**|.

--remote-wait-silent [+{comando}] {file} ...
Come --remote-wait, ma non emette messaggi se non si trova un server. Vedere |**--remote-wait-silent**|.

--servername {nome}
Specifica il nome del server Vim a cui inviare, o quello

da usare se si vuole fungere da server Vim.
Vedere |**--servername**|.

--remote-send {chiavi}
Invia {chiavi} a un server Vim ed esce.
Vedere |**--remote-send**|.

--remote-expr {espressione}
Valuta {espressione} in un altro Vim che funge da server. Il risultato è stampato su stdout.
Vedere |**--remote-expr**|.

--serverlist Elenca una lista di nomi di server Vim ed esce.
Vedere |**--serverlist**|.

--socketid {id} *--socketid*
Solo per la GUI Vim GTK+. Chiede a gvim di tentare di usare il meccanismo GtkPlug, in modo da venir eseguito all'interno di un'altra finestra.
Vedere |**gui-gtk-socketid**| per dettagli.

--windowid {id} *--windowid*
Solo per la GUI Vim Win32. Chiede a gvim di tentare di usare la finestra {id} come "padre" (parent), in modo da essere eseguito all'interno di quella finestra.
Vedere |**gui-w32-windowid**| per dettagli.

--echo-wid *--echo-wid*
Solo per GUI Vim GTK+. Chiede a gvim di visualizzare l'ID della finestra sullo stdout, in modo da poterlo usare per eseguire Vim in un widget di kpart [una struttura della GUI KDE]. Il formato dell'output è: >
WID: 12345\n

--role {ruolo} *--role*
Solo per la GUI GTK+ 2. Imposta il ruolo della finestra principale a {ruolo}. Il ruolo della finestra può essere usato da un software di gestione della finestra per identificare univocamente una finestra, per poter, p.es. ripristinare il posizionamento della stessa sullo schermo. L'argomento --role è passato automaticamente quando si ripristini una sessione al momento del login.
Vedere |**gui-gnome-session**|.

-P {titolo-padre} *-P* *MDI* *E671* *E672*
Solo per Win32: Specifica il titolo dell'applicazione padre (parent). Se possibile, Vim sarà eseguito in una finestra MDI (Multiple Document Interface, più documenti editati un un'unica finestra) all'interno dell'applicazione. {titolo-padre} deve comparire nel titolo della finestra dell'applicazione padre. Accertatevi che sia abbastanza specifico.
Nota: l'implementazione è ancora primitiva. Non funziona con tutte le applicazioni e il menù non funziona.

-nb *-nb*
-nb={nome-file}
-nb:{hostname}:{indirizzo}:{password}
Tenta una connessione a Netbeans e diviene un editor server per quest'applicazione. La seconda forma specifica un file da cui leggere le informazioni relative alla connessione. La terza forma specifica il nome dell'host, l'indirizzo e la password per connettersi a Netbeans. |**netbeans-run**| {disponibile solo se compilato con la funzionalità |**+netbeans_intg**|; se non è questo il caso, specificando -nb si provocherà solo la chiusura di Vim}

Se il programma eseguibile è invocato come "view", Vim sarà eseguito in modo di sola-lettura. Ciò può tornare utile se è possibile creare un link "hard" (puntare a un unico file eseguibile, nella stessa directory, con un nome diverso) o un link "simbolico" (associare un nome di file in una directory con un file residente in qualsiasi altra parte del Sistema Operativo) da "view" a "vim".

Iniziare in modo di sola-lettura si può anche fare specificando "vim -R".

Se il programma eseguibile è chiamato "ex", Vim inizierà in modo Ex.
Ciò implica che verranno accettati solo comandi che iniziano con ":".
Ma quando si specifica l'argomento "-v", Vim inizierà comunque in modo
Normal.

Ulteriori argomenti sono disponibili su sistemi di tipo Unix, se Vim è
compilato con il supporto GUI X11. Vedere |**gui-resources**|.

==
2. Vim in Amiga *starting-amiga*

Iniziare Vim dal Workbench *workbench*

Vim può essere chiamato dal Workbench cliccando due volte sulla sua icona.
In tal caso Vim partirà con un buffer vuoto.

Si può chiamare Vim per una sessione di edit su uno o più file, usando
un'icona "Project". Lo "Strumento di default" (Default Tool) dell'icona
dev'essere il nome completo del programma eseguibile Vim.
Il nome del file ".info" dev'essere lo stesso del file di testo (da editare).
Cliccando due volte su quest'icona, Vim sarà richiamato avendo come argomento
il nome corrente di file, che sarà letto nel buffer (se il file esiste).
Si possono editare più file premendo il tasto delle maiuscole mentre si
clicca sulle icone, e cliccando due volte sull'ultima. Lo "Strumento di
default" per tutte queste icone dev'essere lo stesso.

Non è possibile passare argomenti a Vim, che non siano nomi di file, dal
Workbench.

Finestra Vim *amiga-finestra*

Vim sarà eseguito nella finestra CLI (Command Line Interface, ossia una
finestra in cui si immettono comandi una riga per volta) in cui è stato
richiamato. Se Vim è stato iniziato dando il comando "run" o "runback", o
se Vim è stato iniziato dal Workbench, aprirà una finestra sua propria.

Dettagli tecnici:
Per aprire una nuova finestra si usa un trucchetto. Appena Vim
determina di non essere eseguito in una normale finestra CLI,
crea uno script in "t:". Questo script file contiene lo stesso
comando con cui è stato richiamato Vim, e un comando "endcli".
Questo script file è eseguito con un comando "newcli" (i comandi
"c:run" e "c:newcli" sono necessari perché la cosa funzioni). Lo
file di script resta presente fino alla ripartenza del sistema, a meno
che non venga cancellato esplicitamente. Operare in questo modo è
necessario per far sì che i comandi (di Vim) ":sh" e ":!" funzionino
correttamente. Ma quando si chiama Vim con l'opzione -f (modo
foreground), non si usa questo metodo. Il motivo è che quando un
programma richiama Vim con l'opzione -f, rimane poi in attesa che
sia terminata la sessione di Vim. Con il trucchetto dello script,
il programma chiamante non sa quando Vim termina. L'opzione -f può
essere usata quando Vim è richiamato da un programma di posta
elettronica, il quale poi resta in attesa che la sessione di edit
finisca. Di conseguenza, i comandi ":sh" e ":!" non sono disponibili
quando si usi l'opzione -f.

Vim riconosce automaticamente la dimensione di una finestra, e la adatta
quando ne vengono modificate le dimensioni. Sotto Amiga DOS 1.3, è
consigliabile usare il programma fastfonts, "FF", per velocizzare la
visualizzazione della finestra ridimensionata.

==
3. Eseguire eVim *evim-keys*

EVim esegue Vim come un editor del tipo "clicca e scrivi". Ciò è molto
differente dall'idea che ha ispirato la nascita di Vi. Ma può essere utile
per chi non usa Vim abbastanza frequentemente da aver acquisito una
familiarità con i comandi. Ci si augura che, imparando a usare i comandi
in modo Normal, le sessioni di edit diventino molto più efficienti.

In Evim queste opzioni sono cambiate, rispetto al loro valore di default:

:set nocompatible	Usare miglioramenti Vim
:set insertmode	Rimanere prevalentemente in modo Insert
:set hidden	Mantenere caricati i buffer non visualizzati
:set backup	Mantenere file di backup (non in VMS)
:set backspace=2	Backspace non limitato alla riga corrente
:set autoindent	Rientranze automatiche per le righe nuove
:set history=50	Mantenere lista degli ultimi 50 comandi Ex
:set ruler	Mostrare la posizione del cursore
:set incsearch	Mostrare corrispondenze iniziali mentre si specifica un criterio di ricerca
:set mouse=a	Usare il mouse in tutti i modi di Vim
:set hlsearch	Evidenziare tutte le corrispondenze a una ricerca
:set whichwrap+=<,>,[,]	<Left> e <Right> possono oltrepassare il limite della riga corrente
:set guioptions-=a	Solo in ambienti non-Unix: non effettuare auto-selezioni

Mappature di tasti:

<Down>	scende righe di schermo, non righe di file
<Up>	idem, verso l'alto
Q	esegue "gq", formattando, invece di passare al modo Ex
<BS>	in modo Visual: cancella la selezione
CTRL-X	in modo Visual: taglia verso la clipboard
<S-Del>	idem
CTRL-C	in modo Visual: copia verso la clipboard
<C-Insert>	idem
CTRL-V	Incolla dalla clipboard (in ogni modo Vim)
<S-Insert>	idem
CTRL-Q	fa quel che si faceva con CTRL-V
CTRL-Z	undo
CTRL-Y	redo
<M-Space>	menù di systema
CTRL-A	seleziona tutto
<C-Tab>	prossima finestra, CTRL-W w
<C-F4>	chiudi finestra, CTRL-W c

Inoltre:
- ":behave mswin" è usata |**:behave**|
- l'evidenziazione sintattica è abilitata
- la determinazione del tipo di file è abilitata, i plugin per tipo file sono abilititati, come pure l'indentazione
- in un file di testo 'textwidth' è impostata a 78

Un suggerimento: Se si vuole passare al modo Normal per poter immettere una serie di comandi, si usi CTRL-L. |**i_CTRL-L**|

==========

4. Inizializzazione *initialization* *startup*

Questa sezione riguarda la versione non-GUI di Vim. Vedere |**gui-fork**| per ulteriori inizializzazioni al momento di attivazione della GUI.

Alla partenza, Vim controlla le variabili d'ambiente e i file e imposta i valori di conseguenza. Vim procede in quest'ordine:

1. Imposta le opzioni 'shell' e 'term' *SHELL* *COMSPEC* *TERM*
   La variabile d'ambiente SHELL, se esiste, è usata per impostare l'opzione 'shell'. In Win32, la variabile COMSPEC è usata se SHELL non è impostata.
   La variabile d'ambiente TERM, se esiste, è usata per impostare l'opzione 'term'. Tuttavia, 'term' sarà modificato al momento della partenza della GUI (passo 8 qui sotto).

2. Elabora gli argomenti
   Le opzioni e nomi di file dal comando che ha chiamato Vim sono ispezionati. Dei buffer sono creati per tutti i file (ma per il momento rimangono vuoti). L'argomento |**-V**| può essere usato per visualizzare o registrare quel che accade in seguito, il che può tornare utile in fase di

debug delle inizializzazioni.

3. Esegue comandi Ex, da variabili d'ambiente e/o file
   Una variabile d'ambiente è letta come se fosse una riga comando Ex, in cui eventuali comandi oltre al primo devono essere separati con '|' o "<NL>".

   vimrc *exrc*

   Un file che contiene comandi di inizializzazione è detto un file "vimrc". Ogni riga in un file .vimrc è eseguito come una riga comando Ex. Talora viene anche detto un file "exrc". Sono file dello stesso tipo, ma "exrc" è il file che Vi ha sempre usato, mentre "vimrc" è un nome usato solo da Vim. Vedere anche |**vimrc-intro**|.

   Posizione delle inizializzazioni personali:

Unix	$HOME/.vimrc o $HOME/.vim/vimrc
MS-Windows	$HOME/_vimrc, $HOME/vimfiles/vimrc o $VIM/_vimrc
Amiga	s:.vimrc, home:.vimrc, home:vimfiles:vimrc o $VIM/.vimrc
Haiku	$HOME/config/settings/vim/vimrc

   I file sono ricercati nell'ordine sopra elencato, e viene utilizzato solo il primo file trovato.

   CONSIGLIO: Tutti i file personalizzati di configurazione di Vim dovrebbero risiedere nella directory $HOME/.vim/ ($HOME/vimfiles/ per MS-Windows). Ciò rende più semplice copiarli da un sistema all'altro.

   Se Vim è stato invocato con "-u nome-file", il file "nome-file" è usato. Tutte le successive inizializzazioni, fino al passo 4, sono omesse. $MYVIMRC non viene impostato.
   "vim -u NORC" si può usare per omettere queste inizializzazioni senza leggere alcun file. "vim -u NONE" omette anche il caricamento dei plugin. |**-u**|

   Se Vim è stato invocato in modo Ex con l'argomento "-s", tutte le successive inizializzazioni, fino al passo 4, sono omesse. Solo l'opzione "-u" è elaborata.

   evim.vim

   a. Se Vim è stato invocato come |**evim**| o |**eview**| o con l'argomento |**-y**|, lo script $VIMRUNTIME/evim.vim sarà eseguito.

   system-vimrc

   b. Per Unix, MS-Windows, VMS, Macintosh, Amiga
      il file di sistema vimrc è letto per le inizializzazioni. Il percorso di questo file è visualizzato se si dà il comando ":version"
      Per lo più questo percorso è "$VIM/vimrc".
      Nota Questo file è SEMPRE letto in modalità 'compatible', poiché l'impostazione automatica dell'opzione 'compatible' è effettuata solo successivamente. Volendo, si può aggiungere un comando ":set nocp".
      Per il Macintosh viene letto il file $VIMRUNTIME/macmap.vim.

   VIMINIT *.vimrc* *_vimrc* *EXINIT* *.exrc* *_exrc* *$MYVIMRC*

   c. Cinque percorsi vengono esaminati per le inizializzazioni. Il primo che viene trovato viene usato, gli altri sono ignorati. La variabile d'ambiente $MYVIMRC è impostata con il percorso del primo file di inizializzazione trovato, a meno che $MYVIMRC sia già stato impostato, e quando si usa VIMINIT.
      I La variabile d'ambiente VIMINIT (vedere anche |**compatible-default**|) (*)
        Il valore di $VIMINIT è usato come una riga di comando Ex.
      II Il file utente .vimrc:

"$HOME/.vimrc"	(per Unix) (*)
"$HOME/.vim/vimrc"	(per Unix) (*)
"s:.vimrc"	(per Amiga) (*)
"home:.vimrc"	(per Amiga) (*)
"home:vimfiles:vimrc"	(per Amiga) (*)
"$VIM/.vimrc"	(per Amiga) (*)
"$HOME/_vimrc"	(per Win32) (*)
"$HOME/vimfiles/vimrc"	(per Win32) (*)
"$VIM/_vimrc"	(per Win32) (*)
"$HOME/config/settings/vim/vimrc"	(per Haiku) (*)

Nota: Per Unix, OS/2 e Amiga, quando ".vimrc" non esiste,
si cerca anche "_vimrc", per il caso in cui si stia usando
un file system compatibile con MS-DOS. Per MS-Windows
".vimrc" è cercato DOPO "_vimrc", nel caso si stiano usando
nomi di file lunghi.
Nota: Per Win32, "$HOME" è controllata per prima.
Se lì non si trova alcun "_vimrc" o ".vimrc", si prova con
"$VIM". Vedere |**$VIM**| per quando $VIM non è impostata.

III La variabile d'ambiente EXINIT.
Il valore di $EXINIT è usato come una riga di comando Ex.

IV Il/I file utente exrc. Vale il discorso fatto per il file utente
.vimrc, rimpiazzando però "vimrc" con "exrc". Ma solo uno dei
file ".exrc" e "_exrc" è usato, a seconda del sistema.
E senza (*)!

V Il file di default vimrc, $VIMRUNTIME/defaults.vim. Questo
file imposta alcuni valori di opzioni, fra cui i comandi
"syntax on" e "filetype on", che è quel che molti nuovi utenti
desiderano. Vedere |**defaults.vim**|.

d. Se l'opzione 'exrc' è a on (e questo NON è il default), la directory
corrente è esaminata alla ricerca di tre file. Il primo che viene
trovato è utilizzato, gli altri sono ignorati.
- Il file ".vimrc" (per Unix, Amiga) (*)
"_vimrc" (per Win32) (*)
- Il file "_vimrc" (per Unix, Amiga) (*)
".vimrc" (per Win32) (*)
- Il file ".exrc" (per Unix, Amiga)
"_exrc" (per Win32)

(*) L'utilizzo di questo file o variabile d'ambiente farà sì che
'compatible' sia impostato a off per default.
Vedere |**compatible-default**|.

Nota: Se si sta usando l'interfaccia |**mzscheme**|, questa è inizializzata
dopo il caricamento del file vimrc. Se si modifica 'mzschemedll'
successivamente, la cosa non ha alcun effetto.

4. Carica gli script plugin. *load-plugins*
Ciò ha lo stesso effetto che dare il comando: >
:runtime! plugin/**/*.vim
<
Il risultato è che tutte le directory elencate nell'opzione
'runtimepath' saranno esaminate cercando una sottodirectory
"plugin" e tutti i file il cui nome termina con ".vim"
saranno eseguiti (in ordine alfabetico all'interno di ogni
directory), e anche nelle (eventuali) sottodirectory.
Tuttavia, le directory in 'runtimepath' il cui nome termina
con "after" sono omesse in questo contesto, e caricate solo
dopo i pacchetti, vedere più sotto.
Il caricamento dei plugin non sarà effettuato quando:
- L'opzione 'loadplugins' sia stata messa a off in un file .vimrc.
- Sulla riga di comando sia stato usato l'argomento |**--noplugin**|.
- Sulla riga di comando sia stato usato l'argomento |**--clean**|.
- Sulla riga di comando sia stato usato l'argomento |**-u**|.
- Se Vim è stato compilato senza la funzionalità |**+eval**|.
Nota Usare "-c 'set noloadplugins'" non basta, perché i
comandi dalla riga di comando non sono stati ancora eseguiti. Si può
usare "--cmd 'set noloadplugins'" o "--cmd 'set loadplugins'" |**--cmd**|.

I pacchetti sono caricati. Questi sono ancora plugin, come sopra,
ma sono contenuti nelle directory "iniziali" specificate con l'opzione
'packpath'. Ogni directory di plugin trovata è aggiunta a quelle
contenute in 'runtimepath'. Vedere |**packages**|.

Gli script di plugin sono caricati, come sopra, ma a questo punto
sono considerate solo le directory il cui nome termina con "after".
Nota: 'runtimepath' sarà stato modificato, se sono stati trovati dei
pacchetti, ma la modifica non dovrebbe aggiungere delle directory il
cui nome termini per "after".

5. Imposta 'shellpipe' e 'shellredir'
Le opzioni 'shellpipe' e 'shellredir' sono impostate in congruenza
con il valore dell'opzione 'shell', a meno che siano già stati
impostati in precedenza.

Ciò significa che Vim determinerà i valori di 'shellpipe' e
'shellredir' automaticamente, se non sono già stati impostati
dall'utente.

6. Imposta 'updatecount' a zero, se è stato usato come argomento il comando
   "-n"

7. Imposta l'opzione 'binary'
   Se è stato specificato il flag "-b" richiamando Vim, le opzioni
   correlate all'editing in modo binario saranno impostate a questo
   punto. Vedere |**-b**|.

8. Effettua inizializzazioni GUI
   Solo se si invoca "gvim", saranno effettuate le inizializzazioni GUI.
   Vedere |**gui-init**|.

9. Legge il file viminfo
   Se l'opzione 'viminfo' non è nulla, viene letto il file viminfo.
   Vedere |**viminfo-file**|.

10. Legge il file quickfix
    Se è stato specificato il flag "-q" richiamando Vim, viene letto
    il file quickfix. Se questa lettura non riesce, Vim termina.

11. Apre tutte le finestre
    Se è stato specificato il flag "-o", le finestre saranno aperte (ma
    non ancora visualizzate).
    Se è stato specificato il flag "-p", le linguette saranno create (ma
    non ancora visualizzate).
    Se è necessario un cambio di schermata, viene effettuato a questo
    punto.
    Il ridisegno delle schermate ha inizio.
    Se è stato specificato il flag "-q", si salta al primo errore.
    I buffer per tutte le finestre vengono caricati, senza innescare
    gli autocomandi |**BufAdd**|.

12. Esegue i comandi iniziali
    Se è stato specificato il flag "-t" richiamando Vim, si salta al tag
    specificato.
    I comandi dati con gli argomenti |**-c**| e |**+cmd**| sono eseguiti.
    Il flag che segnala che Vim è in fase di inizializzazione è messo a
    0, has("vim_starting") restituisce zero da qui in poi.
    La variabile |**v:vim_did_enter**| è impostata a 1.
    Se l'opzione 'insertmode' è impostata, si entra in modo Insert.
    Gli autocomandi |**VimEnter**| sono eseguiti.

La variabile $MYVIMRC o $MYGVIMRC sarà impostata col nome del primo file
vimrc e/o gvimrc trovato.

Alcuni consigli sull'uso delle inizializzazioni ~

Impostazioni standard:
Si crei un file vimrc per impostare i valori e le mappature di default
per tutte le sessioni di edit. Lo si metta in un posto in cui sia
trovato al precedente passo 3.b:

```
~/.vimrc        (Unix)
s:.vimrc        (Amiga)
$VIM\_vimrc     (Win32)
~/config/settings/vim/vimrc (Haiku)
```

Nota La creazione di un file .vimrc farà sì che l'opzione 'compatible' sia
impostata a off per default. Vedere |**compatible-default**|.

Impostazioni locali:
Si mettano tutti i comandi che servono per editare (solo) all'interno di una
particolare directory in un file vimrc e lo si posizioni in quella directory
col nome ".vimrc" ("_vimrc" per Win32). NOTA: Per far sì che Vim elabori
questi file speciali, va attivata l'opzione 'exrc'.
Vedere anche |**trojan-horse**|.

Impostazioni di sistema:
Questo vale solo se si sta gestendo un sistema Unix con molti utenti si

desidera impostare dei valori di default che valgano per tutti gli utenti.
Occorre creare un file .vimrc con comandi per le impostazioni e le mappature
di default e metterlo nel posto indicato dall'output del comando ":version".

Salvare lo stato corrente di Vim in un file ~

Dopo aver cambiato valori di opzioni o aver creato una mappatura, può essere
desiderabile salvarle in un file .vimrc da usare in seguito. Vedere
|**save-settings**| su come salvare le impostazioni correnti in un file.

Evitare problemi di inizializzazione per utenti Vi ~

Vi usa la variabile EXINIT e il file "~/.exrc". Per questo motivo, se non
si desidera interferire con il comportamento di Vi, si dovranno usare la
variabile VIMINIT e il file "vimrc".

Variabili in ambiente Amiga ~

Sull'Amiga, esistono due tipi di variabili d'ambiente. Quelle impostata
con il comando "setenv" di DOS versione 1.3 (o successiva) sono riconosciuti.
Vedere il manuale AmigaDos 1.3. Le variabili d'ambiente impostate con il
vecchio comando "Manx Set" (prima della versione 5.0) non sono riconosciute.

Separatori di riga MS-Windows ~

In MS-Windwos, Vim si aspetta che tutti i file .vimrc abbiano come separatore
di riga la coppia di caratteri <CR><NL>. Ciò può essere causa di problemi se
si usa un file che abbia come separatore solo il carattere <NL> e una delle
righe contenga qualcosa del tipo ":map xx yy^M". Il carattere finale ^M
(<CR>) sarà ignorato.

Valore di default compatibile con Vi ~

compatible-default

Quando Vim viene richiamato, l'opzione 'compatible' è on. Essa sarà usata
mentre Vim effettua le sue inizializzazioni. Ma se:
- si individua un file utente vimrc, o
- si individua un file vimrc nella directory corrente, o
- la variabile d'ambiente "VIMINIT" è impostata, o
- è stato specificato l'argomento "-N" sulla riga di comando, o
- è stato specificato l'argomento "--clean" da riga di comando, o
- lo script |**defaults.vim**| è caricato, o
- si individua un file gvimrc,

l'opzione verrà impostata a 'nocompatible'.

Si noti che ciò NON succede qualora sia stato trovato un file vimrc
valido a livello dell'intero sistema.

Ciò ha l'effetto collaterale di abilitare o disabilitare altre opzioni
(vedere 'compatible'). Ma solo le opzioni che non sono già state
abilitate o disabilitate saranno modificate. Ciò produce lo stesso
effetto che se il valore 'compatible' fosse stato presente al momento di
richiamare Vim.

'compatible' non viene cambiato, e |**defaults.vim**| non viene caricato:
- quando Vim è stato chiamato con l'argomento |**-u**| sulla riga di comando,
  specie con "-u NONE", o
- quando è stato chiamato con l'argomento |**-C**| sulla riga di comando, o
- quando il nome con cui Vim è stato chiamato finisce per "ex". (Ciò è
  stato fatto per far sì che Vim si comporti come "ex", se è chiamato
  come "ex")

Ma c'è un effetto collaterale se si abilita o disabilita 'compatible' nel
momento in cui si trova un file .vimrc: Le mappature sono interpretate
nel momento in cui sono incontrate. Ciò fa differenze se in esse sono
presenti cose come "<CR>". Se le mappature dipendono da un certo valore
di 'compatible', quest'opzione va abilitata o disabilitata prima di
impostare la mappatura.

Valori di default se non è presente un file .vimrc ~

defaults.vim

Se Vim è chiamato normalmente e non viene trovato alcun file vimrc,
viene eseguito lo script $VIMRUNTIME/defaults.vim. Questo script
imposta 'compatible' a off, attiva l'evidenziazione sintattica, e
alcune altre cose. Vedere lo script per i dettagli. NOTA: Questo
comportamento è attivo per Vim 8.0, non per Vim 7.4. (Per l'esattezza
è stato aggiunto a Vim 7.4 solo dalla patch 7.4.2111).

Questo comportamento dovrebbe andare bene per dei nuovi utenti di Vim
Se si crea un file .vimrc personale, è consigliabile aggiungere
queste righe nella parte iniziale del file: >

```
unlet! skip_defaults_vim
source $VIMRUNTIME/defaults.vim
```

In questo modo Vim funzione come farebbe se non ci fosse stato alcun file
.vimrc personale. Copiare il file .vimrc da $VIMRUNTIME/vimrc_example
è un modo per fare questo. In alternativa, si può copiare il file
defaults.vim nel file .vimrc personale, e modificarlo come desiderato
(così facendo, peraltro, i futuri aggiornamenti al file non saranno
disponibili).

Se alcuni dei valori di default non sono quelli desiderati, si può sempre
eseguire il file defaults.vim e quindi modificare le impostazioni secondo
le proprie preferenze. Vedere il file defaults.vim per indicazioni su
come modificare ogni singola opzione.

skip_defaults_vim

Se si usa un file vimrc a livello di sistema, e non si vuole utilizzare
defaults.vim per modificare le impostazioni, occorre impostare a on
la variabile "skip_defaults_vim". Se questa variabile è stata impostata
a on, e si desidera caricare defaults.vim dall'interno di un file .vimrc
personale, occorre prima annullare skip_defaults_vim, come mostrato
nell'esempio precedente.

Evitare cavalli di Troia ~

trojan-horse

Mentre si leggono i file "vimrc" o "exrc" nella directory corrente, alcuni
comandi possono essere disabilitati per motivi di sicurezza, impostando
l'opzione 'secure'. Questo è sempre fatto quando il comando è eseguito da
un file di tag. In caso contrario sarebbe possibile finire per usare un file
vimrc o un file di tag creato da qualcun altro e che contiene comandi
pericolosi. I comandi disabilitati sono quelli che iniziano una shell,
quelli che scrivono su un file e ":autocmd". I comandi ":map" sono
visualizzati, in modo da poter controllare quali tasti vengono mappati.

Se si desidera che Vim esegua tutti i comandi in un file .vimrc
locale, si può impostare a off l'opzione 'secure' nelle variabili d'ambiente
EXINIT o VIMINIT o nei file "exrc" o "vimrc" globali. Ciò non è possibile
per i file "vimrc" o "exrc" contenuti nella directory corrente, per ovvi
motivi.

Nei sistemi Unix, ciò succede solo se non si è i proprietari del file
vimrc. Attenzione: Se si scompatta un archivio che contiene un file vimrc o
exrc, se ne erediterà la proprietà. In questo caso la protezione di
sicurezza non sarà presente. Si dovrà controllare il file .vimrc prima di
eseguire Vim su file di quella directory, oppure mettere a off l'opzione
'exrc'. Alcuni sistemi Unix consentono a un utente di usare il comando
"chown" per cambiare il proprietario di un file. Questo rende possibile a
un altro utente la creazione di un file vimrc pericoloso, assegnandone la
proprietà all'utente in uso. Prestare attenzione!

Quando si usano comandi di ricerca tag, l'esecuzione del comando di
ricerca (la parte finale della riga nel file di tag) è sempre eseguita in
modalità sicura. Questo comportamento è lo stesso adottato quando si
segue un comando da un file vimrc/exrc nella directory corrente.

Se la partenza di Vim è lenta ~

slow-start

Se Vim impiega molto tempo per partire, si usi l'argomento |**--startuptime**|
per accertarsi di quel che succede. Ci sono alcune cause frequenti:
- Se la versione Unix è stata compilata con la GUI e/o X11 (cercare,
  nell'output di ":version", "+GUI" e/o "+X11"), può essere necessario
  caricare delle librerie condivise e connettersi al server X11.
  Si provi a compilare una versione in cui GUI e X11 siano disabilitati.

  Questo dovrebbe anche rendere più piccolo il programma eseguibile.
  Si usi l'argomento |**-X**| sulla riga di comando per evitare di connettersi
  a un server X quando si sta lavorando da un terminale.
- Se "viminfo" è abilitata, il caricamento del file viminfo può richiedere
  del tempo. Si può determinare se questo è il problema disabilitando
  viminfo per una volta (usare l'argomento Vim "-i NONE", |**-i**|). Si può
  ridurre il numero di righe mantenute in un registro impostando:
  ":set viminfo='20,<50,s10". |**viminfo-file**|.

Messaggio introduttivo ~

:intro

Quando Vim inizia senza un nome di file, un messaggio introduttivo è visualizzato (per chi non sa che cos'è Vim). Il messaggio scompare non appena la videata è ridisegnata per qualsiasi motivo. Per visualizzare ancora il messaggio, si usi il comando ":intro" (se non c'è abbastanza spazio, solo una parte del messaggio è visualizzata).
   Per evitare il messaggio introduttivo alla partenza, si aggiunga il flag 'I' a 'shortmess'.

info-message

Gli argomenti |**--help**| e |**--version**| fanno sì che Vim stampi un messaggio ed esca subito dopo. Normalmente il messaggio è inviato allo stdout, e quindi può essere ridiretto a un file con: >

```
vim --help >file
```

Dall'interno di Vim: >

```
:read !vim --help
```

Se si sta usando gvim, il programma controlla se è stato iniziato dal desktop (scrivania) senza che ci sia un terminale a cui dirigere i messaggi. È questo il caso quando sia stdout che stderr non sono una tty ("tty" fa parte del nome file usato per effettuare I/O da/verso terminale). Ciò rende il comando ":read", usato nell'esempio precedente, impossibilitato a funzionare. Per ottenere che funzioni, occorre impostare 'shellredir' a ">" invece che al valore di default ">&": >

```
:set shellredir=>
:read !gvim --help
```

Questo, tuttavia, non funzionerà per sistemi in cui gvim non usa affatto stdout.

==

5. $VIM e $VIMRUNTIME

$VIM

La variabile d'ambiente "$VIM" è usata per individuare vari file utente usati da Vim, come lo script utente iniziale ".vimrc". Ciò dipende dal sistema, vedere |**startup**|.

Per evitare che a ogni utente sia richiesto di impostare la variabile d'ambiente $VIM, Vim cerca di ottenere il valore di $VIM seguendo quest'ordine:
1. Il valore definito dalla variabile d'ambiente $VIM. Si può usare per
   far sì che Vim cerchi i suoi file di supporto in una directory
   specifica. Esempio: >
        setenv VIM /home/paul/vim
2. Il percorso dell'opzione 'helpfile' è usato, ma solo se non contiene
   esso stesso delle variabili d'ambiente (il default è
   "$VIMRUNTIME/doc/help.txt": torniamo al dilemma se è nato prima l'uovo o
   la gallina). Il nome del file ("help.txt" o altro specificato) è rimosso.
   Sono poi rimossi i nomi di directory, in quest'ordine: "doc", "runtime" e
   "vim{versione}" (p.es., "vim82").
3. Per Win32 Vim prova a usare il nome di directory del file eseguibile.
   Se il nome termina per "/src", anch'esso è rimosso. Questo è utile se si
   è scompattato il file sorgente in formato .zip su una data directory,
   e si è modificato il percorso di ricerca per trovare il file eseguibile
   di Vim. Sono rimossi i nomi finali di directory, in quest'ordine:
   "runtime" e "vim{versione}" (p.es., "vim82").
4. In Unix la directory di installazione specificata al momento della
   compilazione è usata (vedere l'output di ":version").

Dopo aver fatto questa ricerca una sola volta, Vim imposterà la variabile
d'ambiente $VIM. Per modificarla in seguito, si usi un comando ":let",
di questo tipo:>
	:let $VIM = "/home/paul/vim/"
<

$VIMRUNTIME
La variabile d'ambiente "$VIMRUNTIME" è usata per individuare vari
file di supporto, quali la documentazione online e i file usati per
l'evidenziazione sintattica. Per esempio, il file di aiuto principale
è normalmente "$VIMRUNTIME/doc/help.txt".
Normalmente non si imposta esplicitamente $VIMRUNTIME, ma si lascia che
Vim lo determini. Questo è l'ordine usato per trovare il valore di
$VIMRUNTIME:
1. Se la variabile d'ambiente $VIMRUNTIME è impostata, è usata. Questo
   può servire quando i file di supporto all'esecuzione (runtime files)
   sono in una directory insolita.
2. Se "$VIM/vim{versione}" esiste, è usata. {versione} è il numero di
   versione di Vim, senza alcun '-' o '.'. Per esempio: "$VIM/vim82".
   Questo è il valore normale di $VIMRUNTIME.
3. Se "$VIM/runtime" esiste, è usata.
4. Il valore di $VIM è usato. Questo per mantenere la compatibilità
   all'indietro con vecchie versioni di Vim.
5. Quando l'opzione 'helpfile' è impostata e non contiene un '$', il suo
   valore è usato, dopo aver rimosso la stringa finale "doc/help.txt".

In Unix, se esiste un default stabilito al tempo della compilazione per
$VIMRUNTIME (si controlli l'output di ":version"), i passi 2, 3 e 4 sono
saltati, e il suddetto default è usato dopo il passo 5. Questo significa
che il default stabilito al tempo della compilazione prevale sul valore di
$VIM. Ciò può essere utile se $VIM vale "/etc" e i file da usare in fase
di esecuzione sono in "/usr/share/vim/vim82".

Dopo aver fatto questa ricerca una sola volta, Vim imposterà la variabile
d'ambiente $VIMRUNTIME. Per modificarla in seguito, si usi un comando ":let",
di questo tipo:>
	:let $VIMRUNTIME = "/home/piet/vim/vim82"

Nel caso vi serva il valore di $VIMRUNTIME in una shell (p.es., per uno script
che faccia ricerche nei file di help) potete ottenerlo così: >

	VIMRUNTIME=`vim -e -T dumb --cmd 'exe "set t_cm=\<C-M>"|echo $VIMRUNTIME|quit' | tr -d '\015' `

Non impostate $VIMRUNTIME a un valore nullo, alcune cose potrebbero
smettere di funzionare.

==
6. Sospendere *suspend*

iconize *iconise* *CTRL-Z* *v_CTRL-Z*
CTRL-Z		Sospende Vim, come ":stop".
		Questo accade nei modi Normal e Visual. Nei modi
		Insert e riga-comando, il CTRL-Z è immesso come un
		carattere normale. In modo Visual Vim ritorna al
		modo Normal.
		Nota: se CTRL-Z annulla una modifica, vedere
		|**mswin.vim**|.

:sus[pend][!]	o		*:sus* *:suspend* *:st* *:stop*
:st[op][!]		Sospende la sessione di Vim.
		Se il '!' non è specificato e 'autowrite' è impostata,
		ogni buffer con modifiche e il cui nome di file è noto
		viene riscritto su disco.
		Se si specifica '!' o 'autowrite' non è impostata, i
		buffer modificati non sono riscritti su disco, ed è
		importante ricordarsi di riportare Vim in esecuzione
		appena possibile!

Nella GUI, la sospensione è implementata riducendo Vim a icona. Nelle
finestre MS-Windows, gvim è minimizzato.

In molti sistemi Unix, è possibile sospendere Vim con CTRL-Z. Ciò è

possibile solo nei modi Normal e Visual (vedere il capitolo successivo, |**vim-modes**|). Vim tornerà in esecuzione se lo si riporta nuovamente in primo piano. Su altri sistemi, CTRL-Z inizierà una nuova shell. Questo equivale a dare il comando ":sh". Vim riprenderà l'esecuzione una volta che si esca dalla shell.

In X-Windows una parte di testo eventualmente copiata non sarà disponibile quando Vim è sospeso. Ciò vuol dire che non sarà possibile incollarla in un'altra applicazione (poiché Vim non è più in memoria, un tentativo di procurarsi il testo copiato bloccherebbe il programma richiedente).

==
7. Uscire *exiting*

Ci sono più modi per uscire da Vim:
- Chiudere l'ultima finestra con `:quit`. Solo non non ci sono modifiche.
- Chiudere l'ultima finestra con `:quit!`. Anche se ci sono modifiche.
- Chiudere tutte le finestre con `:qall`. Solo non non ci sono modifiche.
- Chiudere tutte le finestre con `:qall!`. Anche se ci sono modifiche.
- Usare `:cquit`. Anche se ci sono modifiche.

Se si usa `:cquit` o se c'erano messaggi di errore Vim esce con un codice di ritorno 1. Gli errori possono essere ignorati usando `:silent!` oppure `:catch`.

==
8. Salvare impostazioni *save-settings*

Per lo più i file .vimrc sono modificati manualmente dall'utente. Questo consente la massima flessibilità. Ci sono alcuni comandi per generare automaticamente un file .vimrc. Si possono usare questi file così come sono stati generati, o copiare/incollare righe da includere in un altro file vimrc.

:mk *:mkexrc*

:mk[exrc] [file] Scrive le correnti mappature di tastiera e le opzioni modificate su [file] (default ".exrc" nella directory corrente), a meno che il file non esista già.

:mk[exrc]! [file] Scrive (o sovrascrive) le correnti mappature di tastiera e le opzioni modificate su [file] (default ".exrc" nella directory corrente).

:mkv *:mkvi* *:mkvimrc*

:mkv[imrc][!] [file] Come ":mkexrc", ma il default è ".vimrc" nella directory corrente. Il comando ":version" è pure scritto nel file.

Questi comandi scrivono comandi ":map" e ":set" in un file, in modo che, quando tali comandi sono stati eseguiti, le mappature di tastiera e le opzioni siano impostate agli stessi valori. Le opzioni 'columns', 'endofline', 'fileformat', 'key', 'lines', 'modified', 'scroll', 'term', 'textmode', 'ttyfast' e 'ttymouse' non sono incluse, perché dipendono dal terminale o dal tipo di file. Nota Le opzioni 'binary', 'paste' e 'readonly' sono incluse, e questo potrebbe non essere sempre quel che si desiderava.

Se dei tasti speciali sono usati nelle mappature, l'opzione 'cpoptions' sarà temporaneamente impostata al valore di default di Vim, per evitare problemi di interpretazione delle mappature. Ciò rende il file incompatibile con Vi, ma ne rende possibile l'uso su terminali differenti.

Solo le mappature globali sono memorizzate, non quelle locali valide solo per un particolare buffer.

Un modo comune di procedere è di usare un file ".vimrc" di default, a cui apportare qualche modifica tramite comandi ":map" e ":set" per poi scrivere il file modificato. Per fare ciò, si deve dapprima leggere il file ".vimrc" di default con un comando come ":source ~piet/.vimrc.Cprogs", modificare le impostazioni e poi salvarlo nella directory corrente con ":mkvimrc!". Se si desidera usare questo file come .vimrc di default, va spostato nella home directory dell'utente (in Unix), in s: (Amiga) o nella directory $VIM

(MS-Windows). Si possono anche usare autocomandi |**autocommand**| e/o modeline |**modeline**|.

vimrc-option-example
Se si vuole solo aggiungere un unico insieme di opzioni al file vimrc, si può seguire questa procedura:
1. Editare il file .vimrc con Vim.
2. Sperimentare con le opzioni finché non si è soddisfatti. P.es., provare diversi valori per 'guifont'.
3. Aggiungere in fondo una riga per impostare il valore desiderato dell'opzione, usando il registro delle espressioni '=' per immettere il valore corrente. P.es., per l'opzione 'guifont': >
   o:set guifont=<C-R>=&guifont<CR><Esc>
< [<C-R> è il CTRL-R, <CR> è il ritorno a capo, <Esc> è il tasto escape]
   Occorre proteggere eventuali caratteri speciali, specie gli spazi.

Nota Quando si crea un file .vimrc, le modifiche fatte possono influenzare l'opzione 'compatible', e ciò può generare parecchi effetti collaterali. Vedere |**'compatible'**|.
":mkvimrc", ":mkexrc" e ":mksession" inseriscono i comandi necessari per impostare o annullare l'opzione 'compatible' all'inizio del file di output, per prevenire questi effetti collaterali.

==
9. Viste e Sessioni *views-sessions*

Questo argomento è trattato nelle sezioni |**21.4**| e |**21.5**| del manuale utente.

View *view-file*
Una Vista (View) è una collezione di impostazioni che si riferiscono a una finestra. Si può salvare una Vista e, se la si ripristina in un secondo tempo, il testo è visualizzato come al momento del salvataggio della Vista. Le opzioni e mappature della finestra originale saranno anche ripristinate in modo da consentire la continuazione dell'edit nelle stesse condizioni in cui la Vista era stata salvata.

Session *session-file*
Una Sessione (Session) mantiene le Viste (Views) di tutte le finestre, come pure le impostazioni globali. Si può salvare una Sessione, e se la si ripristina in una secondo tempo, la visualizzazione della finestra è la stessa di quando si era effettuato il salvataggio.
Si può utilizzare una Sessione per passare velocemente da un progetto a un altro, caricando automaticamente i file sui quali si stava lavorando in precedenza su quel progetto.

Viste e Sessioni sono una graziosa aggiunta ai file viminfo, i quali sono usati per ricordare informazioni relative all'insieme di tutte le Viste e Sessioni |**viminfo-file**|.

Si può iniziare velocemente a editare usando una Sessione o Vista, salvata in precedenza, specificando l'argomento |**-S**|: >
        vim -S Sessione.vim
<
Tutto ciò è {non disponibile se compilato senza la funzionalità |**+mksession**|}.

:mks *:mksession*
:mks[ession][!] [file] Scrive uno script Vim per ripristinare la sessione di edit corrente.
                        Quando si aggiunge [!], il file viene riscritto, se esiste già.
                        Quando si omette [file] il valore "Session.vim" è usato.

L'output di ":mksession" è come quello di ":mkvimrc", ma ulteriori comandi sono aggiunti al file. Quali siano questi comandi, dipende dall'opzione 'sessionoptions'. Il file risultante, quando eseguito con un comando ":source":
1. Ripristina mappature e opzioni globali, se 'sessionoptions' contiene "options". Le mappature per degli script locali non saranno ripristinate.
2. Ripristina le variabili globali il cui nome inizia con una lettera maiuscola e contiene almeno una lettera minuscola, se 'sessionoptions' contiene "globals".

3. Scarica tutti i buffer correntemente caricati.
4. Ripristina la directory corrente se 'sessionoptions' contiene "curdir", o imposta la directory corrente a quella in cui risiede il file di Sessione se 'sessionoptions' contiene "sesdir".
5. Ripristina la posizione della finestra GUI, se 'sessionoptions' contiene "winpos".
6. Ripristina la dimensione dello schermo, se 'sessionoptions' contiene "resize".
7. Ricarica la lista dei buffer, posizionati all'ultima posizione del cursore in ogni buffer. Se 'sessionoptions' contiene "buffers" tutti i buffer sono ripristinati, compresi quelli nascosti e quelli non presenti sullo schermo. In caso contrario, solo i buffer visibili nella finestra sono ripristinati.
8. Ripristina tutte le finestre con lo stesso formato. Se 'sessionoptions' contiene "help", le finestre di help sono ripristinate. Se 'sessionoptions' contiene "blank", le finestre che stanno editando un file che non ha ancora un nome saranno ripristinate.
   Se 'sessionoptions' contiene "winsize" e non ci sono finestre (help/blank) non ripristinate, anche le dimensioni originali delle finestre sono ripristinate (in base alle dimensioni dello schermo).
   In caso contrario, alle finestre sono assegnate dimensioni secondo buon senso.
9. Ripristina le Viste per tutte le finestre, come con |**:mkview**|. Ma in base alle specifiche di 'sessionoptions' e non di 'viewoptions'.
10. Se un file esiste con lo stesso nome del file di Sessione, ma che termina con "x.vim" (per eXtra), anche questo viene eseguito. Si può usare il file *x.vim per fornire ulteriori impostazioni e azioni associate con una data Sessione, come aggiungere elementi ai menù, nella versione GUI.

Dopo il ripristino della Sessione, il nome completo del file che descrive la Sessione corrente è disponibile nella variabile interna "v:this_session" |**this_session-variable**|.
Un esempio di mappatura: >
```
  :nmap <F2> :wa<Bar>exe "mksession! " . v:this_session<CR>:so ~/sessions/
```
Questa mappatura salva la Sessione corrente, e fa partire il comando per caricarne un'altra.

Una sessione include tutte le linguette, a meno che "tabpages" sia stata tolta da 'sessionoptions'. |**tab-page**|

L'autocomando per l'evento |**SessionLoadPost**| è richiamato quando un file di sessione è caricato/eseguito.

SessionLoad-variable
Durante il caricamento del file di sessione, la variabile globale SessionLoad global è impostata a 1.
Dei plugin potrebbero servirsi di questa informazione per posticipare qualche attività fino a dopo che viene eseguito l'autocomando per l'evento SessionLoadPost.

:mkvie *:mkview*

:mkvie[w][!] [file] Scrive uno script Vim per ripristinare i contenuti della Vista corrente.
Quando si aggiunge [!], il file viene riscritto, se esiste già.
Quando si omette [file], oppure si specifica un numero tra 1 e 9, un nome di file viene generato, nella directory specificata da 'viewdir'. Quando l'ultimo elemento del percorso di 'viewdir' non esiste, la directory con quel nome viene creata. P.es., quando 'viewdir' vale "$VIM/vimfiles/view" la directory "view" è creata in "$VIM/vimfiles".
Un file già esistente è in questo caso sovrascritto.
Usare |**:loadview**| per caricare nuovamente questa Vista.
Quando [file] è il nome di un file ('viewdir' non è usata), un comando per editare il file è aggiunto al file generato.

L'output di ":mkview" contiene i seguenti elementi:
1. La lista degli argomenti usati nella finestra. Quando si sta usando la lista globale degli argomenti, è questa quella che viene ripristinata [e sostituisce quella corrente].
   L'indice all'interno della lista degli argomenti è pure ripristinato.
2. Il file in modifica nella finestra. Se non c'è ancora un file, la

   finestra è lasciata vuota, al momento del ripristino.
3. Sono ripristinate mappature, abbreviazioni e opzioni locali alla finestra se 'viewoptions' contiene "options" o "localoptions". Per le opzioni, sono ripristinati solo i valori che sono locali per il buffer corrente e quelli che sono locali alla finestra.
   Quando Vista che viene salvata fa parte di una sessione e "options" è in 'sessionoptions', i valori globali delle opzioni locali vengono pure salvati.
4. Le piegature vengono ripristinate quando il metodo di piegatura usato è quello manuale, e se 'viewoptions' contiene "folds". Vengono ripristinate le piegature aperte e chiuse manualmente.
5. La posizione all'interno del file e la posizione del cursore. Non funziona molto bene quando si è in presenza di piegature chiuse.
6. La directory corrente locale, se è differente dalla directory corrente globale e 'viewoptions' contiene "curdir".

Nota I ripristini di Viste e Sessioni non sono perfetti:
- Non tutto viene ripristinato. Per esempio, le funzioni definite, gli autocomandi e ":syntax on" non sono inclusi. Cose come i contenuti dei registri e la storia della riga di comando sono contenuti in viminfo, non nelle Sessioni o Viste salvate.
- I valori delle opzioni globali vengono impostati quando sono diversi dal valore di default. Quando il valore corrente non è quello di default, se si carica una Sessione non verrà cambiato al valore di default. Le opzioni locali saranno peraltro impostate al valore di default.
- Mappature esistenti saranno sovrascritte senza segnalazione. Una mappatura già esistente potrebbe generare errori a causa di ambiguità.
- Quando si salvano delle piegature manuali e quando si salvano delle piegature aperte/chiuse manualmente, le possibile modifiche intervenute fra il salvataggio e la ricarica della Vista potrebbero alterarle.
- Lo script Vim non è molto efficiente. Ma resta comunque più veloce che ribattere tutti i comandi necessari manualmente!

:lo *:loadview*

:lo[adview] [nr] Carica la Vista per il file corrente. Quando [nr] è omesso, viene caricata la Vista immagazzinata tramite ":mkview".
Quando [nr] è specificato, viene caricata la Vista immagazzinata con ":mkview [nr]".

La combinazione di ":mkview" e ":loadview" può essere usata per salvare fino a 10 differenti Viste di un file. Le Viste sono ricordate nella directory specificata tramite l'opzione 'viewdir'. Le Viste sono immagazzinate usando nome di file. Se un file è rinominato o acceduto attraverso un link (simbolico) la Vista non verrà trovata.

È auspicabile pulire, di tanto in tanto, la directory 'viewdir'.

Per salvare a ripristinare automaticamente le Viste relative un file sorgente in linguaggio C: >

```
au BufWinLeave *.c mkview
au BufWinEnter *.c silent loadview
```

==========

10. Il file viminfo *viminfo* *viminfo-file* *E136* *E575* *E576* *E577*

Se uscite da Vim e ci ritornare in seguito, normalmente molte informazioni andrebbero perse. Il file viminfo si può usare per ricordare tali informazioni, consentendo così di ricominciare da dove ci si era fermati.

Questo è dettagliato nella sezione |**21.3**| del manuale utente.

Il file viminfo è usato per immagazzinare:
- La storia delle righe di comando.
- La storia delle stringhe di ricerca.
- La storia delle righe in input.
- Il contenuto dei registri non vuoti.
- Marcatori per parecchi file.
- Marcatori di file, che puntano a punti interni a un file.
- L'ultima espressione di ricerca/sostituzione (per 'n' e '&').
- La lista dei buffer.
- Le variabili globali.

Il file viminfo non è supportato quando la funzionalità |**+viminfo**| sia stata disabilitata al momento della compilazione.

Si potrebbe usare per tutto questo un file di Sessione. La differenza è che il file viminfo non dipende da ciò su cui si sta lavorando al momento. Di solito il file viminfo è uno solo. I file di Sessione sono usati per salvare lo status quo di una specifica sessione di edit. Si potrebbero avere più file di Sessione, una per ogni progetto a cui si sta lavorando.
Il file Viminfo e quelli di Sessione, presi insieme, possono essere usati per richiamare Vim e cominciare subito a lavorare nella modalità preferita.
|**session-file**|

viminfo-read
Quando Vim è richiamato e l'opzione 'viminfo' non è nulla, il contenuto del file viminfo viene letto e l'informazione può essere usata laddove serva. Alla variabile |**v:oldfiles**| è assegnato un valore. Le marcature non sono lette alla partenza di Vim (ma lo sono le marcature interne a un file). Vedere |**initialization**| per come impostare l'opzione 'viminfo' alla partenza di Vim.

viminfo-write
Quando Vim termina e 'viminfo' non è nulla, l'informazione è immagazzinata nel file viminfo (in realtà è integrata in quello che esiste già, se qualche informazione già esiste).
L'opzione 'viminfo' è una stringa contenente una lista di quali informazioni dovrebbero essere memorizzate, e contiene delle soglie che stabiliscono dei limiti a quanto deve essere immagazzinato (vedere 'viminfo').

L'integrazione che crea un nuovo file viminfo all'uscita da una sessione di Vim può avvenire in due modi.
La maggior parte degli elementi che sono stati modificati o impostati nella sessione corrente di Vim viene inserita nel file, e ciò che non è stato modificato viene riempito utilizzando ciò che è correntemente contenuto nel file viminfo. Per esempio:
- La sessione Vim A legge viminfo, che contiene la variabile START
- La sessione Vim B fa lo stesso
- La sessione Vim A imposta le variabili AAA e ENTRAMBE ed esce
- La sessione Vim B imposta le variabili BBB e ENTRAMBE ed esce

Il contenuto finale di viminfo avrà:

START	- era in viminfo e non è stata modificata dalle sessioni A e B
AAA	- valore dalla sessione A, la sessione B lo conserva
BBB	- valore dalla sessione B
ENTRAMBE	- valore dalla sessione B, il valore dalla sessione A è perso

viminfo-timestamp
Per alcuni elementi si usa l'indicazione del momento della modifica, al fine di conservare la versione modificata per ultima. In questo caso non importa in quale ordine terminano le sessioni di Vim, viene sempre tenuto l'elemento più recente. Questo metodo è usato per:
- La storia delle righe di comando usate.
- La storia delle stringhe usate come argomenti di ricerca.
- La storia delle righe in input.
- Il contenuto dei registri che non sono vuoti.
- La lista dei salti fatti durante le modifiche.
- Le marcature all'interno dei file.

L'aggiunta del tempo di ultima modifica è stata aggiunta prima della versione Vim 8.0. Versioni meno recenti di Vim, a partire dalla versione 7.4.1131, aggiungono il tempo di modifica degli elementi, ma non ne fanno ancora uso. Quindi, se si sta usando sia una versione meno recente che una versione nuova di Vim, i dati più recenti saranno quelli utilizzati.

Note per Unix:
- la protezione del file viminfo sarà impostata in modo da impedire ad altri utenti di leggerlo, perché può contenere del testo o dei comandi utilizzati durante le sessioni di edit.
- Se si vuole condividere il file viminfo con altri utenti (p.es. quando si usa il comando "su" per cambiare utente), si può rendere il file scrivibile da parte del gruppo o da ogni utente. Vim rispetterà queste abilitazioni quando rimpiazzerà il file viminfo. Attenzione, non è il caso di permettere a tutti di accedere al vostro file viminfo!
- Vim non sovrascriverà un file viminfo per il quale l'utente "vero" corrente

non abbia la possibilità di riscriverlo. Questo può servire quando si usa il comando "su" per diventare l'utente root, ma la variabile $HOME è ancora impostata alla home directory dell'utente originale. Altrimenti Vim creerebbe un file viminfo il cui proprietario è root, e che nessun altro può più leggere.
- Il file viminfo non può essere un link simbolico. Ciò permette di evitare problemi di sicurezza.

Le marcature sono immagazzinate separatamente, per ogni singolo file. Quando un file è letto e 'viminfo' non è un file vuoto, le marcature relative a quel file sono lette dal the file viminfo. NOTA: Le marcature sono scritte solo quando si esce da Vim, il che va bene, perché a quel punto sono immagazzinate quelle di tutti i file aperti nella sessione di editing corrente, tranne nel caso in cui sia stato usato il comando ":bdel". Se si vogliono salvare le marcature per un file che si sta per rilasciare con ":bdel", va usato il comando ":wv". Le marcature '[' e ']' non sono immagazzinate, ma quella '"' lo è. La marcatura '"' è molto utile per saltare all'ultima posizione del cursore, quella in cui ci si trovava l'ultima volta che si è terminato di editare il file. Nessuna marcatura è salvata per file il cui nome inizi con una qualsiasi stringa di caratteri per la quale sia specificato il flag "r" in 'viminfo'. Ciò si può usare per evitare di salvare marcature per file contenuto in supporti rimovibili (per MS-Windows si potrebbe specificare "ra:,rb:", per Amiga "rdf0:,rdf1:,rdf2:").
La variabile |**v:oldfiles**| contiene una lista (separata da virgole) di tutti i nomi di file per i quali il file viminfo contiene delle marcature.

viminfo-file-marks
Le marcature a lettere maiuscole (da 'A a 'Z) sono immagazzinate quando si scrive il file viminfo. Le marcature numeriche (da '0 a '9) sono un po' speciali. Quando il file viminfo è scritto (all'uscita da Vim o con il comando :wviminfo"), '0 è impostata alla posizione corrente del cursore. Il vecchio '0 è trasferito in '1, '1 in '2, etc. Ciò è simile al quel che succede con i registri di cancellazione da "1 a "9. Se la posizione corrente del cursore è già presente in uno dei registri da '0 a '9, viene spostata in '0, per evitare di avere due volte la stessa posizione. Il risultato è che con digitando "'0", è possibile saltare all'indietro fino al file e alla riga in cui si è usciti da Vim. Per provarlo subito, si può provare a usare questo comando: >

```
vim -c "normal '0"
```

In una shell compatibile con csh si potrebbe definire un alias per farlo: >

```
alias lvim vim -c '"'normal "'"0'"'
```

Per una shell di tipo bash: >

```
alias lvim='vim -c "normal '\''0"'
```

Il flag "r" in 'viminfo' va usato per specificare per quali file le marcature non vanno ricordate.

NOME DEL FILE VIMINFO *viminfo-file-name*

- Il nome di default del file viminfo è "$HOME/.viminfo" per Unix, "s:.viminfo" per Amiga, "$HOME_viminfo" per Win32. Per Win32, quando $HOME non è impostata, si usa "$VIM_viminfo". Quando neppure $VIM è impostata, si usa "c:_viminfo".
- Il flag 'n' nell'opzione 'viminfo' si può usare per specificare un altro nome per il file |**'viminfo'**|.
- L'argomento Vim "-i" si può usare per impostare un altro nome di file, |**-i**|. Quando il nome di file specificato è "NONE" (tutto maiuscolo), nessun file viminfo è letto o scritto. Lo stesso vale per i comandi descritti più sotto!
- l'opzione 'viminfofile' si può usare anche come argomento "-i". In effetti, il valore dell'argomento "-i" è memorizzato nell'opzione 'viminfofile'.
- Per i comandi descritti più sotto, un altro nome di file può essere assegnato, prevalendo sul nome di default, e sul nome eventualmente specificato con 'viminfo' o "-i" (a meno che il nome sia NONE).

CHARACTER ENCODING *viminfo-encoding*

Il testo nel file viminfo è codificato come specificato dall'opzione 'encoding'. Normalmente si lavora sempre con la stessa codifica (con lo stesso valore di 'encoding'), e quindi non dovrebbero esserci problemi. In ogni caso, se si legge il file viminfo con un valore di 'encoding' differente da quello con cui era stato scritto, una parte del testo (i caratteri non-ASCII) possono essere non validi. Se questo non è accettabile, si deve aggiungere il flag 'c' all'opzione 'viminfo': >

```
:set viminfo+=c
```

Vim tenterà allora di convertire il testo nel file viminfo dal valore di 'encoding' in cui era stato scritto al valore di 'encoding' corrente. Ciò richiede che Vim sia stato compilato con la funzionalità |**+iconv**|. I nomi di file non sono convertiti.

LEGGERE E SCRIVERE MANUALMENTE VIMINFO *viminfo-read-write*

Due comandi possono essere usati per leggere e scrivere il file viminfo manualmente. Questa funzionalità può essere usata per passare i contenuti di registri fra due programmi Vim in esecuzione: dapprima si dà il comando ":wv" in uno di essi, e poi ":rv" nell'altro. Nota Se il registro in questione conteneva già qualcosa, occorre immettere ":rv!". Si noti anche che dare questi comandi significa sovrascrivere tutto con informazioni provenienti dal primo Vim, compresa la storia delle righe di comando, etc.

Il file viminfo stesso può anche essere modificato a mano, sebbene sia consigliabile partire da un file già esistente, per ottenere una formattazione corretta del file. Se lo si esamina, si vedrà che è ragionevolmente autoesplicativo. Ciò può essere utile per creare un secondo file, che potrebbe essere chiamato "~/.my_viminfo" che potrebbe contenere alcune impostazioni che si desidera avere alla partenza di Vim. Per esempio, si possono pre-caricare dei registri con dati particolari, o inserire certi comandi nella storia delle righe di comando. Una riga nel file .vimrc come >

```
:rviminfo! ~/.my_viminfo
```

si può usare per caricare queste informazioni. Si potrebbero tenere dei file viminfo differenti per differenti tipi di file (p.es., del sorgente C) e caricarli a seconda del nome del file in edit, usando il comando ":autocmd" (vedere |**:autocmd**|).

viminfo-errors

Quando Vim trova un errore mentre legge un file viminfo, non sovrascriverà quel file. Se ci sono più di 10 errori, Vim si ferma nella lettura del file viminfo. Ciò serve a evitare di cancellare per errore un file, nel caso che il nome del file viminfo sia sbagliato. Un caso in cui questo succede è se si immette per errore il comando "vim -i file", mentre si intendeva scrivere "vim -R file" (sì, è successo davvero, per errore!). Se si vuole sovrascrivere un file viminfo contenente un errore, si può correggere l'errore, oppure cancellare il file (mentre Vim è in esecuzione, in modo che buona parte delle informazioni vengano poi ripristinate).

:rv *:rviminfo* *E195*

:rv[iminfo][!] [file] Legge dal file viminfo [file] (default: vedere sopra). Se si specifica [!], qualsiasi informazione sia già stata impostata (registri, marcature, |**v:oldfiles**|, etc.) sarà sovrascritta.

:wv *:wviminfo* *E137* *E138* *E574* *E886* *E929*

:wv[iminfo][!] [file] Scrive il file viminfo [file] (default: vedere sopra). Le informazioni nel file sono dapprima lette, per fondere le nuove informazioni con quelle precedenti. Quando si usa [!], le vecchie informazioni non sono lette per nulla, solo le informazioni della sessione sono scritte. Se 'viminfo' è un file vuoto, saranno scritte le marcature relative a un massimo di 100 file.
Se si riceve un errore "E929: Troppi file temporanei viminfo" si controlli che non ci siano dei vecchi file temporanei ancora presenti (p.es. ~/.viminf*), e che si sia autorizzati a scrivere nella directory del

file viminfo.

:ol *:oldfiles*

:ol[dfiles] Lista i file che hanno marcature immagazzinate nel file viminfo. Questa lista è letta all'inizio della sessione di Vim ed è modificata in seguito solo con il comando `:rviminfo!`. Vedere anche |**v:oldfiles**|. Il numero si può usare con |**c_#<**|. L'output può essere filtrato usando |**:filter**|, p.es.: >

```
:filter /\.vim/ oldfiles
```

< Il filtro dell'output è effettuato sul nome del file. {disponibile solo quando compilato con la funzionalità |**+eval**|}

:bro[wse] ol[dfiles][!]
Lista nomi di file come fa |**:oldfiles**|, e quindi chiede di immettere un numero. Quando il numero è valido, quel file della lista viene editato. Se si riceve la richiesta di passare alla pagina seguente |**press-enter**| si può rispondere "q" e ottenere ugualmente la richiesta di fornire un numero di file. Si usi ! per lasciare (non salvare) un buffer modificato. |**abandon**| {non disponibile quando Vim è compilato con funzionalità tiny o small}

vim:tw=78:ts=8:noet:ft=help:norl: