```
*quickref.txt*  For Vim version 7.2.  Last change: 2009 Jan 22


                  VIM REFERENCE MANUAL    by Bram Moolenaar

                           Quick reference guide

                                                         *quickref* *Contents*
 tag      subject                        tag      subject       ~
|Q_ct|  list of help files              |Q_re|  Repeating commands
|Q_lr|  motion: Left-right              |Q_km|  Key mapping
|Q_ud|  motion: Up-down                 |Q_ab|  Abbreviations
|Q_tm|  motion: Text object             |Q_op|  Options
|Q_pa|  motion: Pattern searches        |Q_ur|  Undo/Redo commands
|Q_ma|  motion: Marks                   |Q_et|  External commands
|Q_vm|  motion: Various                 |Q_qf|  Quickfix commands
|Q_ta|  motion: Using tags              |Q_vc|  Various commands
|Q_sc|  Scrolling                       |Q_ce|  Ex: Command-line editing
|Q_in|  insert: Inserting text          |Q_ra|  Ex: Ranges
|Q_ai|  insert: Keys                    |Q_ex|  Ex: Special characters
|Q_ss|  insert: Special keys            |Q_st|  Starting VIM
|Q_di|  insert: Digraphs                |Q_ed|  Editing a file
|Q_si|  insert: Special inserts         |Q_fl|  Using the argument list
|Q_de|  change: Deleting text           |Q_wq|  Writing and quitting
|Q_cm|  change: Copying and moving      |Q_ac|  Automatic commands
|Q_ch|  change: Changing text           |Q_wi|  Multi-window commands
|Q_co|  change: Complex                 |Q_bu|  Buffer list commands
|Q_vi|  Visual mode                     |Q_sy|  Syntax highlighting
|Q_to|  Text objects                    |Q_gu|  GUI commands
                                        |Q_fo|  Folding

------------------------------------------------------------------------------
N is used to indicate an optional count that can be given before the command.
------------------------------------------------------------------------------
*Q_lr*          Left-right motions

|h|     N  h            left (also: CTRL-H, <BS>, or <Left> key)
|l|     N  l            right (also: <Space> or <Right> key)
|0|        0            to first character in the line (also: <Home> key)
|^|        ^            to first non-blank character in the line
|$|     N  $            to the last character in the line (N-1 lines lower)
                           (also: <End> key)
|g0|       g0           to first character in screen line (differs from "0"
                           when lines wrap)
|g^|       g^           to first non-blank character in screen line (differs
                           from "^" when lines wrap)
|g$|    N  g$           to last character in screen line (differs from "$"
                           when lines wrap)
|gm|       gm           to middle of the screen line
|bar|   N  |            to column N (default: 1)
|f|     N  f{char}      to the Nth occurrence of {char} to the right
|F|     N  F{char}      to the Nth occurrence of {char} to the left
|t|     N  t{char}      till before the Nth occurrence of {char} to the right
|T|     N  T{char}      till before the Nth occurrence of {char} to the left
```

```
|;|     N  ;            repeat the last "f", "F", "t", or "T" N times
|,|     N  ,            repeat the last "f", "F", "t", or "T" N times in
                           opposite direction
------------------------------------------------------------------------------
*Q_ud*          Up-down motions

|k|     N  k            up N lines (also: CTRL-P and <Up>)
|j|     N  j            down N lines (also: CTRL-J, CTRL-N, <NL>, and <Down>)
|-|     N  -            up N lines, on the first non-blank character
|+|     N  +            down N lines, on the first non-blank character (also:
                           CTRL-M and <CR>)
|_|     N  _            down N-1 lines, on the first non-blank character
|G|     N  G            goto line N (default: last line), on the first
                           non-blank character
|gg|    N  gg           goto line N (default: first line), on the first
                           non-blank character
|N%|    N  %            goto line N percentage down in the file.  N must be
                           given, otherwise it is the |%| command.
|gk|    N  gk           up N screen lines (differs from "k" when line wraps)
|gj|    N  gj           down N screen lines (differs from "j" when line wraps)
------------------------------------------------------------------------------
*Q_tm*          Text object motions

|w|     N  w            N words forward
|W|     N  W            N blank-separated |WORD|s forward
|e|     N  e            forward to the end of the Nth word
|E|     N  E            forward to the end of the Nth blank-separated |WORD|
|b|     N  b            N words backward
|B|     N  B            N blank-separated |WORD|s backward
|ge|    N  ge           backward to the end of the Nth word
|gE|    N  gE           backward to the end of the Nth blank-separated |WORD|

|)|     N  )            N sentences forward
|(|     N  (            N sentences backward
|}|     N  }            N paragraphs forward
|{|     N  {            N paragraphs backward
|]]|    N  ]]           N sections forward, at start of section
|[[|    N  [[           N sections backward, at start of section
|][|    N  ][           N sections forward, at end of section
|[]|    N  []           N sections backward, at end of section
|[(|    N  [(           N times back to unclosed '('
|[{|    N  [{           N times back to unclosed '{'
|[m|    N  [m           N times back to start of method (for Java)
|[M|    N  [M           N times back to end of method (for Java)
|])|    N  ])           N times forward to unclosed ')'
|]}|    N  ]}           N times forward to unclosed '}'
|]m|    N  ]m           N times forward to start of method (for Java)
|]M|    N  ]M           N times forward to end of method (for Java)
|[#|    N  [#           N times back to unclosed "#if" or "#else"
|]#|    N  ]#           N times forward to unclosed "#else" or "#endif"
|[star| N  [*           N times back to start of comment "/*"
|]star| N  ]*           N times forward to end of comment "*/"
------------------------------------------------------------------------------
*Q_pa*          Pattern searches
```

```
|/|     N  /{pattern}[/[offset]]<CR>
                        search forward for the Nth occurrence of {pattern}
|?|     N  ?{pattern}[?[offset]]<CR>
                        search backward for the Nth occurrence of {pattern}
|/<CR>| N  /<CR>        repeat last search, in the forward direction
|?<CR>| N  ?<CR>        repeat last search, in the backward direction
|n|     N  n            repeat last search
|N|     N  N            repeat last search, in opposite direction
|star|  N  *            search forward for the identifier under the cursor
|#|     N  #            search backward for the identifier under the cursor
|gstar| N  g*           like "*", but also find partial matches
|g#|    N  g#           like "#", but also find partial matches
|gd|       gd           goto local declaration of identifier under the cursor
|gD|       gD           goto global declaration of identifier under the cursor

|pattern|               Special characters in search patterns

                        meaning              magic   nomagic   ~
                matches any single character    .       \.
                       matches start of line    ^       ^
                               matches <EOL>    $       $
                       matches start of word    \<      \<
                         matches end of word    \>      \>
        matches a single char from the range    [a-z]   \[a-z]
      matches a single char not in the range    [^a-z]  \[^a-z]
                  matches an identifier char    \i      \i
                   idem but excluding digits    \I      \I
                 matches a keyword character    \k      \k
                   idem but excluding digits    \K      \K
               matches a file name character    \f      \f
                   idem but excluding digits    \F      \F
               matches a printable character    \p      \p
                   idem but excluding digits    \P      \P
             matches a white space character    \s      \s
         matches a non-white space character    \S      \S

                               matches <Esc>    \e      \e
                               matches <Tab>    \t      \t
                                matches <CR>    \r      \r
                                matches <BS>    \b      \b

     matches 0 or more of the preceding atom    *       \*
     matches 1 or more of the preceding atom    \+      \+
        matches 0 or 1 of the preceding atom    \=      \=
        matches 2 to 5 of the preceding atom    \{2,5}  \{2,5}
                  separates two alternatives    \|      \|
                group a pattern into an atom    \(\)    \(\)

|search-offset|         Offsets allowed after search command

    [num]       [num] lines downwards, in column 1
    +[num]      [num] lines downwards, in column 1
    -[num]      [num] lines upwards, in column 1
    e[+num]     [num] characters to the right of the end of the match
    e[-num]     [num] characters to the left of the end of the match
    s[+num]     [num] characters to the right of the start of the match
    s[-num]     [num] characters to the left of the start of the match
    b[+num]     [num] identical to s[+num] above (mnemonic: begin)
    b[-num]     [num] identical to s[-num] above (mnemonic: begin)
    ;{search-command}   execute {search-command} next
------------------------------------------------------------------------------
*Q_ma*          Marks and motions

|m|        m{a-zA-Z}    mark current position with mark {a-zA-Z}
|`a|       `{a-z}       go to mark {a-z} within current file
|`A|       `{A-Z}       go to mark {A-Z} in any file
|`0|       `{0-9}       go to the position where Vim was previously exited
|``|       ``           go to the position before the last jump
|`quote|   `"           go to the position when last editing this file
|`[|       `[           go to the start of the previously operated or put text
|`]|       `]           go to the end of the previously operated or put text
|`<|       `<           go to the start of the (previous) Visual area
|`>|       `>           go to the end of the (previous) Visual area
|`.|       `.           go to the position of the last change in this file
|'|        '{a-zA-Z0-9[]'"<>.}
                        same as `, but on the first non-blank in the line
|:marks|  :marks        print the active marks
|CTRL-O|  N  CTRL-O     go to Nth older position in jump list
|CTRL-I|  N  CTRL-I     go to Nth newer position in jump list
|:ju|     :ju[mps]      print the jump list
------------------------------------------------------------------------------
*Q_vm*          Various motions

|%|        %            find the next brace, bracket, comment, or "#if"/
                           "#else"/"#endif" in this line and go to its match
|H|     N  H            go to the Nth line in the window, on the first
                           non-blank
|M|        M            go to the middle line in the window, on the first
                           non-blank
|L|     N  L            go to the Nth line from the bottom, on the first
                           non-blank

|go|    N  go                   go to Nth byte in the buffer
|:go|   :[range]go[to] [off]    go to [off] byte in the buffer
------------------------------------------------------------------------------
*Q_ta*          Using tags

|:ta|      :ta[g][!] {tag}      Jump to tag {tag}
|:ta|      :[count]ta[g][!]     Jump to [count]'th newer tag in tag list
|CTRL-]|      CTRL-]            Jump to the tag under cursor, unless changes
                                   have been made
|:ts|      :ts[elect][!] [tag]  List matching tags and select one to jump to
|:tjump|   :tj[ump][!] [tag]    Jump to tag [tag] or select from list when
                                   there are multiple matches
|:ltag|    :lt[ag][!] [tag]     Jump to tag [tag] and add matching tags to the
                                   location list.

|:tags|    :tags                Print tag list
|CTRL-T|   N  CTRL-T            Jump back from Nth older tag in tag list
|:po|      :[count]po[p][!]     Jump back from [count]'th older tag in tag list
```

```
|:tnext|   :[count]tn[ext][!]   Jump to [count]'th next matching tag
|:tp|      :[count]tp[revious][!] Jump to [count]'th previous matching tag
|:tr|      :[count]tr[ewind][!] Jump to [count]'th matching tag
|:tl|      :tl[ast][!]          Jump to last matching tag

|:ptag|    :pt[ag] {tag}        open a preview window to show tag {tag}
|CTRL-W_}|     CTRL-W }         like CTRL-] but show tag in preview window
|:pts|     :pts[elect]          like ":tselect" but show tag in preview window
|:ptjump|  :ptj[ump]            like ":tjump" but show tag in preview window
|:pclose|  :pc[lose]            close tag preview window
|CTRL-W_z|     CTRL-W z         close tag preview window
------------------------------------------------------------------------------
*Q_sc*          Scrolling

|CTRL-E|        N  CTRL-E       window N lines downwards (default: 1)
|CTRL-D|        N  CTRL-D       window N lines Downwards (default: 1/2 window)
|CTRL-F|        N  CTRL-F       window N pages Forwards (downwards)
|CTRL-Y|        N  CTRL-Y       window N lines upwards (default: 1)
|CTRL-U|        N  CTRL-U       window N lines Upwards (default: 1/2 window)
|CTRL-B|        N  CTRL-B       window N pages Backwards (upwards)
|z<CR>|            z<CR> or zt  redraw, current line at top of window
|z.|               z.    or zz  redraw, current line at center of window
|z-|               z-    or zb  redraw, current line at bottom of window

These only work when 'wrap' is off:
|zh|            N  zh           scroll screen N characters to the right
|zl|            N  zl           scroll screen N characters to the left
|zH|            N  zH           scroll screen half a screenwidth to the right
|zL|            N  zL           scroll screen half a screenwidth to the left
------------------------------------------------------------------------------
*Q_in*          Inserting text

|a|     N  a    append text after the cursor (N times)
|A|     N  A    append text at the end of the line (N times)
|i|     N  i    insert text before the cursor (N times) (also: <Insert>)
|I|     N  I    insert text before the first non-blank in the line (N times)
|gI|    N  gI   insert text in column 1 (N times)
|o|     N  o    open a new line below the current line, append text (N times)
|O|     N  O    open a new line above the current line, append text (N times)
|:startinsert|  :star[tinsert][!]  start Insert mode, append when [!] used
|:startreplace| :startr[eplace][!]  start Replace mode, at EOL when [!] used

in Visual block mode:
|v_b_I|    I    insert the same text in front of all the selected lines
|v_b_A|    A    append the same text after all the selected lines
------------------------------------------------------------------------------
*Q_ai*          Insert mode keys

|insert-index|  alphabetical index of Insert mode commands

leaving Insert mode:
|i_<Esc>|       <Esc>             end Insert mode, back to Normal mode
|i_CTRL-C|      CTRL-C            like <Esc>, but do not use an abbreviation
|i_CTRL-O|      CTRL-O {command}  execute {command} and return to Insert mode
```

```
moving around:
|i_<Up>|        cursor keys       move cursor left/right/up/down
|i_<S-Left>|    shift-left/right  one word left/right
|i_<S-Up>|      shift-up/down     one screenful backward/forward
|i_<End>|       <End>             cursor after last character in the line
|i_<Home>|      <Home>            cursor to first character in the line
------------------------------------------------------------------------------
*Q_ss*          Special keys in Insert mode

|i_CTRL-V|      CTRL-V {char}..   insert character literally, or enter decimal
                                     byte value
|i_<NL>|        <NL> or <CR> or CTRL-M or CTRL-J
                                  begin new line
|i_CTRL-E|      CTRL-E            insert the character from below the cursor
|i_CTRL-Y|      CTRL-Y            insert the character from above the cursor

|i_CTRL-A|      CTRL-A            insert previously inserted text
|i_CTRL-@|      CTRL-@            insert previously inserted text and stop
                                     Insert mode
|i_CTRL-R|      CTRL-R {0-9a-z%#:.-="}  insert the contents of a register

|i_CTRL-N|      CTRL-N            insert next match of identifier before the
                                     cursor
|i_CTRL-P|      CTRL-P            insert previous match of identifier before
                                     the cursor
|i_CTRL-X|      CTRL-X ...        complete the word before the cursor in
                                     various ways

|i_<BS>|        <BS> or CTRL-H    delete the character before the cursor
|i_<Del>|       <Del>             delete the character under the cursor
|i_CTRL-W|      CTRL-W            delete word before the cursor
|i_CTRL-U|      CTRL-U            delete all entered characters in the current
                                     line
|i_CTRL-T|      CTRL-T            insert one shiftwidth of indent in front of
                                       the current line
|i_CTRL-D|      CTRL-D            delete one shiftwidth of indent in front of
                                     the current line
|i_0_CTRL-D|    0 CTRL-D          delete all indent in the current line
|i_^_CTRL-D|    ^ CTRL-D          delete all indent in the current line,
                                     restore indent in next line
------------------------------------------------------------------------------
*Q_di*          Digraphs

|:dig|     :dig[raphs]          show current list of digraphs
|:dig|     :dig[raphs] {char1}{char2} {number} ...
                                add digraph(s) to the list

In Insert or Command-line mode:
|i_CTRL-K|      CTRL-K {char1} {char2}
                                  enter digraph
|i_digraph|     {char1} <BS> {char2}
                                  enter digraph if 'digraph' option set
------------------------------------------------------------------------------
*Q_si*          Special inserts
```

```
|:r|       :r [file]       insert the contents of [file] below the cursor
|:r!|      :r! {command}   insert the standard output of {command} below the
                              cursor
------------------------------------------------------------------------------
*Q_de*          Deleting text

|x|     N  x            delete N characters under and after the cursor
|<Del>| N  <Del>        delete N characters under and after the cursor
|X|     N  X            delete N characters before the cursor
|d|     N  d{motion}    delete the text that is moved over with {motion}
|v_d|   {visual}d       delete the highlighted text
|dd|    N  dd           delete N lines
|D|     N  D            delete to the end of the line (and N-1 more lines)
|J|     N  J            join N-1 lines (delete <EOL>s)
|v_J|   {visual}J       join the highlighted lines
|gJ|    N  gJ           like "J", but without inserting spaces
|v_gJ|  {visual}gJ      like "{visual}J", but without inserting spaces
|:d|    :[range]d [x]   delete [range] lines [into register x]
------------------------------------------------------------------------------
*Q_cm*          Copying and moving text

|quote|   "{char}       use register {char} for the next delete, yank, or put
|:reg|    :reg          show the contents of all registers
|:reg|    :reg {arg}    show the contents of registers mentioned in {arg}
|y|       N  y{motion}  yank the text moved over with {motion} into a register
|v_y|        {visual}y  yank the highlighted text into a register
|yy|      N  yy         yank N lines into a register
|Y|       N  Y          yank N lines into a register
|p|       N  p          put a register after the cursor position (N times)
|P|       N  P          put a register before the cursor position (N times)
|]p|      N  ]p         like p, but adjust indent to current line
|[p|      N  [p         like P, but adjust indent to current line
|gp|      N  gp         like p, but leave cursor after the new text
|gP|      N  gP         like P, but leave cursor after the new text
------------------------------------------------------------------------------
*Q_ch*          Changing text

|r|       N  r{char}    replace N characters with {char}
|gr|      N  gr{char}   replace N characters without affecting layout
|R|       N  R          enter Replace mode (repeat the entered text N times)
|gR|      N  gR         enter virtual Replace mode: Like Replace mode but
                           without affecting layout
|v_b_r|   {visual}r{char}
                        in Visual block mode: Replace each char of the
                           selected text with {char}

        (change = delete text and enter Insert mode)
|c|       N  c{motion}  change the text that is moved over with {motion}
|v_c|        {visual}c  change the highlighted text
|cc|      N  cc         change N lines
|S|       N  S          change N lines
|C|       N  C          change to the end of the line (and N-1 more lines)
|s|       N  s          change N characters
|v_b_c|      {visual}c  in Visual block mode: Change each of the selected
                           lines with the entered text
```

```
|v_b_C|      {visual}C  in Visual block mode: Change each of the selected
                           lines until end-of-line with the entered text

|~|       N  ~          switch case for N characters and advance cursor
|v_~|        {visual}~  switch case for highlighted text
|v_u|        {visual}u  make highlighted text lowercase
|v_U|        {visual}U  make highlighted text uppercase
|g~|         g~{motion} switch case for the text that is moved over with
                           {motion}
|gu|         gu{motion} make the text that is moved over with {motion}
                           lowercase
|gU|         gU{motion} make the text that is moved over with {motion}
                           uppercase
|v_g?|       {visual}g? perform rot13 encoding on highlighted text
|g?|         g?{motion} perform rot13 encoding on the text that is moved over
                           with {motion}

|CTRL-A|  N  CTRL-A     add N to the number at or after the cursor
|CTRL-X|  N  CTRL-X     subtract N from the number at or after the cursor

|<|       N  <{motion}  move the lines that are moved over with {motion} one
                           shiftwidth left
|<<|      N  <<         move N lines one shiftwidth left
|>|       N  >{motion}  move the lines that are moved over with {motion} one
                           shiftwidth right
|>>|      N  >>         move N lines one shiftwidth right
|gq|      N  gq{motion} format the lines that are moved over with {motion} to
                           'textwidth' length
|:ce|     :[range]ce[nter] [width]
                        center the lines in [range]
|:le|     :[range]le[ft] [indent]
                        left-align the lines in [range] (with [indent])
|:ri|     :[range]ri[ght] [width]
                        right-align the lines in [range]
------------------------------------------------------------------------------
*Q_co*          Complex changes

|!|        N  !{motion}{command}<CR>
                        filter the lines that are moved over through {command}
|!!|       N  !!{command}<CR>
                        filter N lines through {command}
|v_!|         {visual}!{command}<CR>
                        filter the highlighted lines through {command}
|:range!|  :[range]! {command}<CR>
                        filter [range] lines through {command}
|=|        N  ={motion}
                        filter the lines that are moved over through 'equalprg'
|==|       N  ==        filter N lines through 'equalprg'
|v_=|         {visual}=
                        filter the highlighted lines through 'equalprg'
|:s|       :[range]s[ubstitute]/{pattern}/{string}/[g][c]
                        substitute {pattern} by {string} in [range] lines;
                           with [g], replace all occurrences of {pattern};
                           with [c], confirm each replacement
|:s|       :[range]s[ubstitute] [g][c]
```

```
                        repeat previous ":s" with new range and options
|&|           &         Repeat previous ":s" on current line without options
|:ret|     :[range]ret[ab][!] [tabstop]
                        set 'tabstop' to new value and adjust white space
                           accordingly
------------------------------------------------------------------------------
*Q_vi*          Visual mode

|visual-index|  list of Visual mode commands.

|v|        v            start highlighting characters  }  move cursor and use
|V|        V            start highlighting linewise    }  operator to affect
|CTRL-V|   CTRL-V       start highlighting blockwise   }  highlighted text
|v_o|      o            exchange cursor position with start of highlighting
|gv|       gv           start highlighting on previous visual area
|v_v|      v            highlight characters or stop highlighting
|v_V|      V            highlight linewise or stop highlighting
|v_CTRL-V| CTRL-V       highlight blockwise or stop highlighting
------------------------------------------------------------------------------
*Q_to*          Text objects (only in Visual mode or after an operator)

|v_aw|     N  aw        Select "a word"
|v_iw|     N  iw        Select "inner word"
|v_aW|     N  aW        Select "a |WORD|"
|v_iW|     N  iW        Select "inner |WORD|"
|v_as|     N  as        Select "a sentence"
|v_is|     N  is        Select "inner sentence"
|v_ap|     N  ap        Select "a paragraph"
|v_ip|     N  ip        Select "inner paragraph"
|v_ab|     N  ab        Select "a block" (from "[(" to "])")
|v_ib|     N  ib        Select "inner block" (from "[(" to "])")
|v_aB|     N  aB        Select "a Block" (from "[{" to "]}")
|v_iB|     N  iB        Select "inner Block" (from "[{" to "]}")
|v_a>|     N  a>        Select "a <> block"
|v_i>|     N  i>        Select "inner <> block"
|v_at|     N  at        Select "a tag block" (from <aaa> to </aaa>)
|v_it|     N  it        Select "inner tag block" (from <aaa> to </aaa>)
|v_a'|     N  a'        Select "a single quoted string"
|v_i'|     N  i'        Select "inner single quoted string"
|v_aquote| N  a"        Select "a double quoted string"
|v_iquote| N  i"        Select "inner double quoted string"
|v_a`|     N  a`        Select "a backward quoted string"
|v_i`|     N  i`        Select "inner backward quoted string"

------------------------------------------------------------------------------
*Q_re*          Repeating commands

|.|        N  .         repeat last change (with count replaced with N)
|q|           q{a-z}    record typed characters into register {a-z}
|q|           q{A-Z}    record typed characters, appended to register {a-z}
|q|           q         stop recording
|@|        N  @{a-z}    execute the contents of register {a-z} (N times)
|@@|       N  @@           repeat previous @{a-z} (N times)
|:@|       :@{a-z}      execute the contents of register {a-z} as an Ex
                           command
```

```
|:@@|      :@@          repeat previous :@{a-z}
|:g|       :[range]g[lobal]/{pattern}/[cmd]
                        Execute Ex command [cmd] (default: ":p") on the lines
                           within [range] where {pattern} matches.
|:g|       :[range]g[lobal]!/{pattern}/[cmd]
                        Execute Ex command [cmd] (default: ":p") on the lines
                           within [range] where {pattern} does NOT match.
|:so|      :so[urce] {file}
                        Read Ex commands from {file}.
|:so|      :so[urce]! {file}
                        Read Vim commands from {file}.
|:sl|      :sl[eep] [sec]
                        don't do anything for [sec] seconds
|gs|       N  gs        Goto Sleep for N seconds
------------------------------------------------------------------------------
*Q_km*          Key mapping

|:map|       :ma[p] {lhs} {rhs}   Map {lhs} to {rhs} in Normal and Visual
                                     mode.
|:map!|      :ma[p]! {lhs} {rhs}  Map {lhs} to {rhs} in Insert and Command-line
                                     mode.
|:noremap|   :no[remap][!] {lhs} {rhs}
                                  Same as ":map", no remapping for this {rhs}
|:unmap|     :unm[ap] {lhs}       Remove the mapping of {lhs} for Normal and
                                     Visual mode.
|:unmap!|    :unm[ap]! {lhs}      Remove the mapping of {lhs} for Insert and
                                     Command-line mode.
|:map_l|     :ma[p] [lhs]         List mappings (starting with [lhs]) for
                                     Normal and Visual mode.
|:map_l!|    :ma[p]! [lhs]        List mappings (starting with [lhs]) for
                                     Insert and Command-line mode.
|:cmap|      :cmap/:cunmap/:cnoremap
                                  like ":map!"/":unmap!"/":noremap!" but for
                                     Command-line mode only
|:imap|      :imap/:iunmap/:inoremap
                                  like ":map!"/":unmap!"/":noremap!" but for
                                     Insert mode only
|:nmap|      :nmap/:nunmap/:nnoremap
                                  like ":map"/":unmap"/":noremap" but for
                                     Normal mode only
|:vmap|      :vmap/:vunmap/:vnoremap
                                  like ":map"/":unmap"/":noremap" but for
                                     Visual mode only
|:omap|      :omap/:ounmap/:onoremap
                                  like ":map"/":unmap"/":noremap" but only for
                                     when an operator is pending
|:mapc|      :mapc[lear]          remove mappings for Normal and Visual mode
|:mapc|      :mapc[lear]!         remove mappings for Insert and Cmdline mode
|:imapc|     :imapc[lear]         remove mappings for Insert mode
|:vmapc|     :vmapc[lear]         remove mappings for Visual mode
|:omapc|     :omapc[lear]         remove mappings for Operator-pending mode
|:nmapc|     :nmapc[lear]         remove mappings for Normal mode
|:cmapc|     :cmapc[lear]         remove mappings for Cmdline mode
|:mkexrc|    :mk[exrc][!] [file]  write current mappings, abbreviations, and
                                     settings to [file] (default: ".exrc";
                                       use ! to overwrite)
|:mkvimrc|   :mkv[imrc][!] [file]
                                    same as ":mkexrc", but with default ".vimrc"
|:mksession| :mks[ession][!] [file]
                                    like ":mkvimrc", but store current files,
                                       windows, etc. too, to be able to continue
                                       this session later.
------------------------------------------------------------------------------
*Q_ab*          Abbreviations

|:abbreviate|   :ab[breviate] {lhs} {rhs}  add abbreviation for {lhs} to {rhs}
|:abbreviate|   :ab[breviate] {lhs}        show abbr's that start with {lhs}
|:abbreviate|   :ab[breviate]              show all abbreviations
|:unabbreviate| :una[bbreviate] {lhs}      remove abbreviation for {lhs}
|:noreabbrev|   :norea[bbrev] [lhs] [rhs]  like ":ab", but don't remap [rhs]
|:iabbrev|      :iab/:iunab/:inoreab       like ":ab", but only for Insert mode
|:cabbrev|      :cab/:cunab/:cnoreab       like ":ab", but only for
                                                Command-line mode
|:abclear|      :abc[lear]                 remove all abbreviations
|:cabclear|     :cabc[lear]                remove all abbr's for Cmdline mode
|:iabclear|     :iabc[lear]                remove all abbr's for Insert mode
------------------------------------------------------------------------------
*Q_op*          Options

|:set|          :se[t]                    Show all modified options.
|:set|          :se[t] all                Show all non-termcap options.
|:set|          :se[t] termcap            Show all termcap options.
|:set|          :se[t] {option}           Set boolean option (switch it on),
                                          show string or number option.
|:set|          :se[t] no{option}         Reset boolean option (switch it off).
|:set|          :se[t] inv{option}        invert boolean option.
|:set|          :se[t] {option}={value}   Set string/number option to {value}.
|:set|          :se[t] {option}+={value}  append {value} to string option, add
                                          {value} to number option
|:set|          :se[t] {option}-={value}  remove {value} to string option,
                                          subtract {value} from number option
|:set|          :se[t] {option}?          Show value of {option}.
|:set|          :se[t] {option}&          Reset {option} to its default value.

|:setlocal|     :setl[ocal]               like ":set" but set the local value
                                          for options that have one
|:setglobal|    :setg[lobal]              like ":set" but set the global value
                                          of a local option

|:fix|          :fix[del]                 Set value of 't_kD' according to
                                          value of 't_kb'.
|:options|      :opt[ions]                Open a new window to view and set
                                          options, grouped by functionality,
                                          a one line explanation and links to
                                          the help.

Short explanation of each option:               *option-list*
'aleph'           'al'      ASCII code of the letter Aleph (Hebrew)
'allowrevins'     'ari'     allow CTRL-_ in Insert and Command-line mode
'altkeymap'       'akm'     for default second language (Farsi/Hebrew)
```

'ambiwidth'	'ambw'	what to do with Unicode chars of ambiguous width
'antialias'	'anti'	Mac OS X: use smooth, antialiased fonts
'autochdir'	'acd'	change directory to the file in the current window
'arabic'	'arab'	for Arabic as a default second language
'arabicshape'	'arshape'	do shaping for Arabic characters
'autoindent'	'ai'	take indent for new line from previous line
'autoread'	'ar'	autom. read file when changed outside of Vim
'autowrite'	'aw'	automatically write file if changed
'autowriteall'	'awa'	as 'autowrite', but works with more commands
'background'	'bg'	"dark" or "light", used for highlight colors
'backspace'	'bs'	how backspace works at start of line
'backup'	'bk'	keep backup file after overwriting a file
'backupcopy'	'bkc'	make backup as a copy, don't rename the file
'backupdir'	'bdir'	list of directories for the backup file
'backupext'	'bex'	extension used for the backup file
'backupskip'	'bsk'	no backup for files that match these patterns
'balloondelay'	'bdlay'	delay in mS before a balloon may pop up
'ballooneval'	'beval'	switch on balloon evaluation
'balloonexpr'	'bexpr'	expression to show in balloon
'binary'	'bin'	read/write/edit file in binary mode
'bioskey'	'biosk'	MS-DOS: use bios calls for input characters
'bomb'		prepend a Byte Order Mark to the file
'breakat'	'brk'	characters that may cause a line break
'browsedir'	'bsdir'	which directory to start browsing in
'bufhidden'	'bh'	what to do when buffer is no longer in window
'buflisted'	'bl'	whether the buffer shows up in the buffer list
'buftype'	'bt'	special type of buffer
'casemap'	'cmp'	specifies how case of letters is changed
'cdpath'	'cd'	list of directories searched with ":cd"
'cedit'		key used to open the command-line window
'charconvert'	'ccv'	expression for character encoding conversion
'cindent'	'cin'	do C program indenting
'cinkeys'	'cink'	keys that trigger indent when 'cindent' is set
'cinoptions'	'cino'	how to do indenting when 'cindent' is set
'cinwords'	'cinw'	words where 'si' and 'cin' add an indent
'clipboard'	'cb'	use the clipboard as the unnamed register
'cmdheight'	'ch'	number of lines to use for the command-line
'cmdwinheight'	'cwh'	height of the command-line window
'columns'	'co'	number of columns in the display
'comments'	'com'	patterns that can start a comment line
'commentstring'	'cms'	template for comments; used for fold marker
'compatible'	'cp'	behave Vi-compatible as much as possible
'complete'	'cpt'	specify how Insert mode completion works
'completefunc'	'cfu'	function to be used for Insert mode completion
'completeopt'	'cot'	options for Insert mode completion
'confirm'	'cf'	ask what to do about unsaved/read-only files
'conskey'	'consk'	get keys directly from console (MS-DOS only)
'copyindent'	'ci'	make 'autoindent' use existing indent structure
'cpoptions'	'cpo'	flags for Vi-compatible behavior
'cscopepathcomp'	'cspc'	how many components of the path to show
'cscopeprg'	'csprg'	command to execute cscope
'cscopequickfix'	'csqf'	use quickfix window for cscope results
'cscopetag'	'cst'	use cscope for tag commands
'cscopetagorder'	'csto'	determines ":cstag" search order
'cscopeverbose'	'csverb'	give messages when adding a cscope database

'cursorcolumn'	'cuc'	highlight the screen column of the cursor
'cursorline'	'cul'	highlight the screen line of the cursor
'debug'		set to "msg" to see all error messages
'define'	'def'	pattern to be used to find a macro definition
'delcombine'	'deco'	delete combining characters on their own
'dictionary'	'dict'	list of file names used for keyword completion
'diff'		use diff mode for the current window
'diffexpr'	'dex'	expression used to obtain a diff file
'diffopt'	'dip'	options for using diff mode
'digraph'	'dg'	enable the entering of digraphs in Insert mode
'directory'	'dir'	list of directory names for the swap file
'display'	'dy'	list of flags for how to display text
'eadirection'	'ead'	in which direction 'equalalways' works
'edcompatible'	'ed'	toggle flags of ":substitute" command
'encoding'	'enc'	encoding used internally
'endofline'	'eol'	write <EOL> for last line in file
'equalalways'	'ea'	windows are automatically made the same size
'equalprg'	'ep'	external program to use for "=" command
'errorbells'	'eb'	ring the bell for error messages
'errorfile'	'ef'	name of the errorfile for the QuickFix mode
'errorformat'	'efm'	description of the lines in the error file
'esckeys'	'ek'	recognize function keys in Insert mode
'eventignore'	'ei'	autocommand events that are ignored
'expandtab'	'et'	use spaces when <Tab> is inserted
'exrc'	'ex'	read .vimrc and .exrc in the current directory
'fileencoding'	'fenc'	file encoding for multi-byte text
'fileencodings'	'fencs'	automatically detected character encodings
'fileformat'	'ff'	file format used for file I/O
'fileformats'	'ffs'	automatically detected values for 'fileformat'
'filetype'	'ft'	type of file, used for autocommands
'fillchars'	'fcs'	characters to use for displaying special items
'fkmap'	'fk'	Farsi keyboard mapping
'foldclose'	'fcl'	close a fold when the cursor leaves it
'foldcolumn'	'fdc'	width of the column used to indicate folds
'foldenable'	'fen'	set to display all folds open
'foldexpr'	'fde'	expression used when 'foldmethod' is "expr"
'foldignore'	'fdi'	ignore lines when 'foldmethod' is "indent"
'foldlevel'	'fdl'	close folds with a level higher than this
'foldlevelstart'	'fdls'	'foldlevel' when starting to edit a file
'foldmarker'	'fmr'	markers used when 'foldmethod' is "marker"
'foldmethod'	'fdm'	folding type
'foldminlines'	'fml'	minimum number of lines for a fold to be closed
'foldnestmax'	'fdn'	maximum fold depth
'foldopen'	'fdo'	for which commands a fold will be opened
'foldtext'	'fdt'	expression used to display for a closed fold
'formatlistpat'	'flp'	pattern used to recognize a list header
'formatoptions'	'fo'	how automatic formatting is to be done
'formatprg'	'fp'	name of external program used with "gq" command
'formatexpr'	'fex'	expression used with "gq" command
'fsync'	'fs'	whether to invoke fsync() after file write
'gdefault'	'gd'	the ":substitute" flag 'g' is default on
'grepformat'	'gfm'	format of 'grepprg' output
'grepprg'	'gp'	program to use for ":grep"
'guicursor'	'gcr'	GUI: settings for cursor shape and blinking
'guifont'	'gfn'	GUI: Name(s) of font(s) to be used

```
'guifontset'      'gfs'     GUI: Names of multi-byte fonts to be used
'guifontwide'     'gfw'     list of font names for double-wide characters
'guiheadroom'     'ghr'     GUI: pixels room for window decorations
'guioptions'      'go'      GUI: Which components and options are used
'guipty'                    GUI: try to use a pseudo-tty for ":!" commands
'guitablabel'     'gtl'     GUI: custom label for a tab page
'guitabtooltip'   'gtt'     GUI: custom tooltip for a tab page
'helpfile'        'hf'      full path name of the main help file
'helpheight'      'hh'      minimum height of a new help window
'helplang'        'hlg'     preferred help languages
'hidden'          'hid'     don't unload buffer when it is |abandon|ed
'highlight'       'hl'      sets highlighting mode for various occasions
'hlsearch'        'hls'     highlight matches with last search pattern
'history'         'hi'      number of command-lines that are remembered
'hkmap'           'hk'      Hebrew keyboard mapping
'hkmapp'          'hkp'     phonetic Hebrew keyboard mapping
'icon'                      let Vim set the text of the window icon
'iconstring'                string to use for the Vim icon text
'ignorecase'      'ic'      ignore case in search patterns
'imactivatekey'   'imak'    key that activates the X input method
'imcmdline'       'imc'     use IM when starting to edit a command line
'imdisable'       'imd'     do not use the IM in any mode
'iminsert'        'imi'     use :lmap or IM in Insert mode
'imsearch'        'ims'     use :lmap or IM when typing a search pattern
'include'         'inc'     pattern to be used to find an include file
'includeexpr'     'inex'    expression used to process an include line
'incsearch'       'is'      highlight match while typing search pattern
'indentexpr'      'inde'    expression used to obtain the indent of a line
'indentkeys'      'indk'    keys that trigger indenting with 'indentexpr'
'infercase'       'inf'     adjust case of match for keyword completion
'insertmode'      'im'      start the edit of a file in Insert mode
'isfname'         'isf'     characters included in file names and pathnames
'isident'         'isi'     characters included in identifiers
'iskeyword'       'isk'     characters included in keywords
'isprint'         'isp'     printable characters
'joinspaces'      'js'      two spaces after a period with a join command
'key'                       encryption key
'keymap'          'kmp'     name of a keyboard mapping
'keymodel'        'km'      enable starting/stopping selection with keys
'keywordprg'      'kp'      program to use for the "K" command
'langmap'         'lmap'    alphabetic characters for other language mode
'langmenu'        'lm'      language to be used for the menus
'laststatus'      'ls'      tells when last window has status lines
'lazyredraw'      'lz'      don't redraw while executing macros
'linebreak'       'lbr'     wrap long lines at a blank
'lines'                     number of lines in the display
'linespace'       'lsp'     number of pixel lines to use between characters
'lisp'                      automatic indenting for Lisp
'lispwords'       'lw'      words that change how lisp indenting works
'list'                      show <Tab> and <EOL>
'listchars'       'lcs'     characters for displaying in list mode
'loadplugins'     'lpl'     load plugin scripts when starting up
'macatsui'                  Mac GUI: use ATSUI text drawing
'magic'                     changes special characters in search patterns
'makeef'          'mef'     name of the errorfile for ":make"
```

'makeprg'	'mp'	program to use for the ":make" command
'matchpairs'	'mps'	pairs of characters that "%" can match
'matchtime'	'mat'	tenths of a second to show matching paren
'maxcombine'	'mco'	maximum nr of combining characters displayed
'maxfuncdepth'	'mfd'	maximum recursive depth for user functions
'maxmapdepth'	'mmd'	maximum recursive depth for mapping
'maxmem'	'mm'	maximum memory (in Kbyte) used for one buffer
'maxmempattern'	'mmp'	maximum memory (in Kbyte) used for pattern search
'maxmemtot'	'mmt'	maximum memory (in Kbyte) used for all buffers
'menuitems'	'mis'	maximum number of items in a menu
'mkspellmem'	'msm'	memory used before |:mkspell| compresses the tree
'modeline'	'ml'	recognize modelines at start or end of file
'modelines'	'mls'	number of lines checked for modelines
'modifiable'	'ma'	changes to the text are not possible
'modified'	'mod'	buffer has been modified
'more'		pause listings when the whole screen is filled
'mouse'		enable the use of mouse clicks
'mousefocus'	'mousef'	keyboard focus follows the mouse
'mousehide'	'mh'	hide mouse pointer while typing
'mousemodel'	'mousem'	changes meaning of mouse buttons
'mouseshape'	'mouses'	shape of the mouse pointer in different modes
'mousetime'	'mouset'	max time between mouse double-click
'mzquantum'	'mzq'	the interval between polls for MzScheme threads
'nrformats'	'nf'	number formats recognized for CTRL-A command
'number'	'nu'	print the line number in front of each line
'numberwidth'	'nuw'	number of columns used for the line number
'omnifunc'	'ofu'	function for filetype-specific completion
'opendevice'	'odev'	allow reading/writing devices on MS-Windows
'operatorfunc'	'opfunc'	function to be called for |g@| operator
'osfiletype'	'oft'	operating system-specific filetype information
'paragraphs'	'para'	nroff macros that separate paragraphs
'paste'		allow pasting text
'pastetoggle'	'pt'	key code that causes 'paste' to toggle
'patchexpr'	'pex'	expression used to patch a file
'patchmode'	'pm'	keep the oldest version of a file
'path'	'pa'	list of directories searched with "gf" et.al.
'preserveindent'	'pi'	preserve the indent structure when reindenting
'previewheight'	'pvh'	height of the preview window
'previewwindow'	'pvw'	identifies the preview window
'printdevice'	'pdev'	name of the printer to be used for :hardcopy
'printencoding'	'penc'	encoding to be used for printing
'printexpr'	'pexpr'	expression used to print PostScript for :hardcopy
'printfont'	'pfn'	name of the font to be used for :hardcopy
'printheader'	'pheader'	format of the header used for :hardcopy
'printmbcharset'	'pmbcs'	CJK character set to be used for :hardcopy
'printmbfont'	'pmbfn'	font names to be used for CJK output of :hardcopy
'printoptions'	'popt'	controls the format of :hardcopy output
'pumheight'	'ph'	maximum height of the popup menu
'quoteescape'	'qe'	escape characters used in a string
'readonly'	'ro'	disallow writing the buffer
'redrawtime'	'rdt'	timeout for 'hlsearch' and |:match| highlighting
'remap'		allow mappings to work recursively
'report'		threshold for reporting nr. of lines changed
'restorescreen'	'rs'	Win32: restore screen when exiting
'revins'	'ri'	inserting characters will work backwards

'rightleft'	'rl'	window is right-to-left oriented
'rightleftcmd'	'rlc'	commands for which editing works right-to-left
'ruler'	'ru'	show cursor line and column in the status line
'rulerformat'	'ruf'	custom format for the ruler
'runtimepath'	'rtp'	list of directories used for runtime files
'scroll'	'scr'	lines to scroll with CTRL-U and CTRL-D
'scrollbind'	'scb'	scroll in window as other windows scroll
'scrolljump'	'sj'	minimum number of lines to scroll
'scrolloff'	'so'	minimum nr. of lines above and below cursor
'scrollopt'	'sbo'	how 'scrollbind' should behave
'sections'	'sect'	nroff macros that separate sections
'secure'		secure mode for reading .vimrc in current dir
'selection'	'sel'	what type of selection to use
'selectmode'	'slm'	when to use Select mode instead of Visual mode
'sessionoptions'	'ssop'	options for |:mksession|
'shell'	'sh'	name of shell to use for external commands
'shellcmdflag'	'shcf'	flag to shell to execute one command
'shellpipe'	'sp'	string to put output of ":make" in error file
'shellquote'	'shq'	quote character(s) for around shell command
'shellredir'	'srr'	string to put output of filter in a temp file
'shellslash'	'ssl'	use forward slash for shell file names
'shelltemp'	'stmp'	whether to use a temp file for shell commands
'shelltype'	'st'	Amiga: influences how to use a shell
'shellxquote'	'sxq'	like 'shellquote', but include redirection
'shiftround'	'sr'	round indent to multiple of shiftwidth
'shiftwidth'	'sw'	number of spaces to use for (auto)indent step
'shortmess'	'shm'	list of flags, reduce length of messages
'shortname'	'sn'	non-MS-DOS: Filenames assumed to be 8.3 chars
'showbreak'	'sbr'	string to use at the start of wrapped lines
'showcmd'	'sc'	show (partial) command in status line
'showfulltag'	'sft'	show full tag pattern when completing tag
'showmatch'	'sm'	briefly jump to matching bracket if insert one
'showmode'	'smd'	message on status line to show current mode
'showtabline'	'stal'	tells when the tab pages line is displayed
'sidescroll'	'ss'	minimum number of columns to scroll horizontal
'sidescrolloff'	'siso'	min. nr. of columns to left and right of cursor
'smartcase'	'scs'	no ignore case when pattern has uppercase
'smartindent'	'si'	smart autoindenting for C programs
'smarttab'	'sta'	use 'shiftwidth' when inserting <Tab>
'softtabstop'	'sts'	number of spaces that <Tab> uses while editing
'spell'		enable spell checking
'spellcapcheck'	'spc'	pattern to locate end of a sentence
'spellfile'	'spf'	files where |zg| and |zw| store words
'spelllang'	'spl'	language(s) to do spell checking for
'spellsuggest'	'sps'	method(s) used to suggest spelling corrections
'splitbelow'	'sb'	new window from split is below the current one
'splitright'	'spr'	new window is put right of the current one
'startofline'	'sol'	commands move cursor to first non-blank in line
'statusline'	'stl'	custom format for the status line
'suffixes'	'su'	suffixes that are ignored with multiple match
'suffixesadd'	'sua'	suffixes added when searching for a file
'swapfile'	'swf'	whether to use a swapfile for a buffer
'swapsync'	'sws'	how to sync the swap file
'switchbuf'	'swb'	sets behavior when switching to another buffer
'synmaxcol'	'smc'	maximum column to find syntax items

'syntax'	'syn'	syntax to be loaded for current buffer
'tabstop'	'ts'	number of spaces that <Tab> in file uses
'tabline'	'tal'	custom format for the console tab pages line
'tabpagemax'	'tpm'	maximum number of tab pages for |-p| and "tab all"
'tagbsearch'	'tbs'	use binary searching in tags files
'taglength'	'tl'	number of significant characters for a tag
'tagrelative'	'tr'	file names in tag file are relative
'tags'	'tag'	list of file names used by the tag command
'tagstack'	'tgst'	push tags onto the tag stack
'term'		name of the terminal
'termbidi'	'tbidi'	terminal takes care of bi-directionality
'termencoding'	'tenc'	character encoding used by the terminal
'terse'		shorten some messages
'textauto'	'ta'	obsolete, use 'fileformats'
'textmode'	'tx'	obsolete, use 'fileformat'
'textwidth'	'tw'	maximum width of text that is being inserted
'thesaurus'	'tsr'	list of thesaurus files for keyword completion
'tildeop'	'top'	tilde command "~" behaves like an operator
'timeout'	'to'	time out on mappings and key codes
'timeoutlen'	'tm'	time out time in milliseconds
'title'		let Vim set the title of the window
'titlelen'		percentage of 'columns' used for window title
'titleold'		old title, restored when exiting
'titlestring'		string to use for the Vim window title
'toolbar'	'tb'	GUI: which items to show in the toolbar
'toolbariconsize'	'tbis'	size of the toolbar icons (for GTK 2 only)
'ttimeout'		time out on mappings
'ttimeoutlen'	'ttm'	time out time for key codes in milliseconds
'ttybuiltin'	'tbi'	use built-in termcap before external termcap
'ttyfast'	'tf'	indicates a fast terminal connection
'ttymouse'	'ttym'	type of mouse codes generated
'ttyscroll'	'tsl'	maximum number of lines for a scroll
'ttytype'	'tty'	alias for 'term'
'undolevels'	'ul'	maximum number of changes that can be undone
'updatecount'	'uc'	after this many characters flush swap file
'updatetime'	'ut'	after this many milliseconds flush swap file
'verbose'	'vbs'	give informative messages
'verbosefile'	'vfile'	file to write messages in
'viewdir'	'vdir'	directory where to store files with :mkview
'viewoptions'	'vop'	specifies what to save for :mkview
'viminfo'	'vi'	use .viminfo file upon startup and exiting
'virtualedit'	've'	when to use virtual editing
'visualbell'	'vb'	use visual bell instead of beeping
'warn'		warn for shell command when buffer was changed
'weirdinvert'	'wi'	for terminals that have weird inversion method
'whichwrap'	'ww'	allow specified keys to cross line boundaries
'wildchar'	'wc'	command-line character for wildcard expansion
'wildcharm'	'wcm'	like 'wildchar' but also works when mapped
'wildignore'	'wig'	files matching these patterns are not completed
'wildmenu'	'wmnu'	use menu for command line completion
'wildmode'	'wim'	mode for 'wildchar' command-line expansion
'wildoptions'	'wop'	specifies how command line completion is done.
'winaltkeys'	'wak'	when the windows system handles ALT keys
'winheight'	'wh'	minimum number of lines for the current window
'winfixheight'	'wfh'	keep window height when opening/closing windows

```
'winfixwidth'     'wfw'     keep window width when opening/closing windows
'winminheight'    'wmh'     minimum number of lines for any window
'winminwidth'     'wmw'     minimal number of columns for any window
'winwidth'        'wiw'     minimal number of columns for current window
'wrap'                      long lines wrap and continue on the next line
'wrapmargin'      'wm'      chars from the right where wrapping starts
'wrapscan'        'ws'      searches wrap around the end of the file
'write'                     writing to a file is allowed
'writeany'        'wa'      write to file with no need for "!" override
'writebackup'     'wb'      make a backup before overwriting a file
'writedelay'      'wd'      delay this many msec for each char (for debug)
------------------------------------------------------------------------------
*Q_ur*          Undo/Redo commands

|u|       N  u          undo last N changes
|CTRL-R|  N  CTRL-R     redo last N undone changes
|U|          U          restore last changed line
------------------------------------------------------------------------------
*Q_et*          External commands

|:shell|        :sh[ell]        start a shell
|:!|            :!{command}     execute {command} with a shell
|K|                K            lookup keyword under the cursor with
                                   'keywordprg' program (default: "man")
------------------------------------------------------------------------------
*Q_qf*          Quickfix commands

|:cc|           :cc [nr]        display error [nr] (default is the same again)
|:cnext|        :cn             display the next error
|:cprevious|    :cp             display the previous error
|:clist|        :cl             list all errors
|:cfile|        :cf             read errors from the file 'errorfile'
|:cgetbuffer|   :cgetb          like :cbuffer but don't jump to the first error
|:cgetfile|     :cg             like :cfile but don't jump to the first error
|:cgetexpr|     :cgete          like :cexpr but don't jump to the first error
|:caddfile|     :caddf          add errors from the error file to the current
                                   quickfix list
|:caddexpr|     :cad            add errors from an expression to the current
                                   quickfix list
|:cbuffer|      :cb             read errors from text in a buffer
|:cexpr|        :cex            read errors from an expression
|:cquit|        :cq             quit without writing and return error code (to
                                   the compiler)
|:make|         :make [args]    start make, read errors, and jump to first
                                   error
|:grep|         :gr[ep] [args]  execute 'grepprg' to find matches and jump to
                                   the first one.
------------------------------------------------------------------------------
*Q_vc*          Various commands

|CTRL-L|           CTRL-L       Clear and redraw the screen.
|CTRL-G|           CTRL-G       show current file name (with path) and cursor
                                   position
|ga|               ga           show ascii value of character under cursor in
                                   decimal, hex, and octal
```

```
|g8|               g8           for utf-8 encoding: show byte sequence for
                                   character under cursor in hex.
|g_CTRL-G|         g CTRL-G     show cursor column, line, and character
                                   position
|CTRL-C|           CTRL-C       during searches: Interrupt the search
|dos-CTRL-Break|   CTRL-Break   MS-DOS: during searches: Interrupt the search
|<Del>|            <Del>        while entering a count: delete last character
|:version|      :ve[rsion]      show version information
|:mode|         :mode N         MS-DOS: set screen mode to N (number, C80,
                                   C4350, etc.)
|:normal|       :norm[al][!] {commands}
                                Execute Normal mode commands.
|Q|             Q               switch to "Ex" mode

|:redir|        :redir >{file}          redirect messages to {file}
|:silent|       :silent[!] {command}    execute {command} silently
|:confirm|      :confirm {command}      quit, write, etc., asking about
                                        unsaved changes or read-only files.
|:browse|       :browse {command}       open/read/write file, using a
                                        file selection dialog
------------------------------------------------------------------------------
*Q_ce*          Command-line editing

|c_<Esc>|       <Esc>              abandon command-line (if 'wildchar' is
                                      <Esc>, type it twice)

|c_CTRL-V|      CTRL-V {char}      insert {char} literally
|c_CTRL-V|      CTRL-V {number}    enter decimal value of character (up to
                                      three digits)
|c_CTRL-K|      CTRL-K {char1} {char2}
                                   enter digraph (See |Q_di|)
|c_CTRL-R|      CTRL-R {0-9a-z"%#:-=}
                                   insert the contents of a register

|c_<Left>|      <Left>/<Right>     cursor left/right
|c_<S-Left>|    <S-Left>/<S-Right> cursor one word left/right
|c_CTRL-B|      CTRL-B/CTRL-E      cursor to beginning/end of command-line

|c_<BS>|        <BS>               delete the character in front of the cursor
|c_<Del>|       <Del>              delete the character under the cursor
|c_CTRL-W|      CTRL-W             delete the word in front of the cursor
|c_CTRL-U|      CTRL-U             remove all characters

|c_<Up>|        <Up>/<Down>        recall older/newer command-line that starts
                                      with current command
|c_<S-Up>|      <S-Up>/<S-Down>    recall older/newer command-line from history
|:history|      :his[tory]         show older command-lines

Context-sensitive completion on the command-line:

|c_wildchar|    'wildchar'  (default: <Tab>)
                                do completion on the pattern in front of the
                                   cursor.  If there are multiple matches,
                                   beep and show the first one; further
                                   'wildchar' will show the next ones.
```

```
|c_CTRL-D|      CTRL-D          list all names that match the pattern in
                                   front of the cursor
|c_CTRL-A|      CTRL-A          insert all names that match pattern in front
                                   of cursor
|c_CTRL-L|      CTRL-L          insert longest common part of names that
                                   match pattern
|c_CTRL-N|      CTRL-N          after 'wildchar' with multiple matches: go
                                   to next match
|c_CTRL-P|      CTRL-P          after 'wildchar' with multiple matches: go
                                   to previous match
------------------------------------------------------------------------------
*Q_ra*          Ex ranges

|:range|        ,               separates two line numbers
|:range|        ;               idem, set cursor to the first line number
                                before interpreting the second one

|:range|        {number}        an absolute line number
|:range|        .               the current line
|:range|        $               the last line in the file
|:range|        %               equal to 1,$ (the entire file)
|:range|        *               equal to '<,'> (visual area)
|:range|        't              position of mark t
|:range|        /{pattern}[/]   the next line where {pattern} matches
|:range|        ?{pattern}[?]   the previous line where {pattern} matches

|:range|        +[num]          add [num] to the preceding line number
                                   (default: 1)
|:range|        -[num]          subtract [num] from the preceding line
                                   number (default: 1)
------------------------------------------------------------------------------
*Q_ex*          Special Ex characters

|:bar|      |           separates two commands (not for ":global" and ":!")
|:quote|    "           begins comment

|:_%|       %           current file name (only where a file name is expected)
|:_#|       #[num]      alternate file name [num] (only where a file name is
                           expected)
        Note: The next four are typed literally; these are not special keys!
|:<cword>|  <cword>     word under the cursor (only where a file name is
                           expected)
|:<cWORD>|  <cWORD>     WORD under the cursor (only where a file name is
                           expected) (see |WORD|)
|:<cfile>|  <cfile>     file name under the cursor (only where a file name is
                           expected)
|:<afile>|  <afile>     file name for autocommand (only where a file name is
                           expected)
|:<sfile>|  <sfile>     file name of a ":source"d file, within that file (only
                           where a file name is expected)

                After "%", "#", "<cfile>", "<sfile>" or "<afile>"
                |::p|       :p          full path
                |::h|       :h          head (file name removed)
                |::t|       :t          tail (file name only)
                |::r|       :r          root (extension removed)
                |::e|       :e          extension
                |::s|       :s/{pat}/{repl}/    substitute {pat} with {repl}
------------------------------------------------------------------------------
*Q_st*          Starting VIM

|-vim|     vim [options]                start editing with an empty buffer
|-file|    vim [options] {file} ..      start editing one or more files
|--|       vim [options] -              read file from stdin
|-tag|     vim [options] -t {tag}       edit the file associated with {tag}
|-qf|      vim [options] -q [fname]     start editing in QuickFix mode,
                                           display the first error

        Most useful Vim arguments (for full list see |startup-options|)

|-gui|  -g                  start GUI (also allows other options)

|-+|    +[num]              put the cursor at line [num] (default: last line)
|-+c|   +{command}          execute {command} after loading the file
|-+/|   +/{pat} {file} ..   put the cursor at the first occurrence of {pat}
|-v|    -v                  Vi mode, start ex in Normal mode
|-e|    -e                  Ex mode, start vim in Ex mode
|-R|    -R                  Read-only mode, implies -n
|-m|    -m                  modifications not allowed (resets 'write' option)
|-d|    -d                  diff mode |diff|
|-b|    -b                  binary mode
|-l|    -l                  lisp mode
|-A|    -A                  Arabic mode ('arabic' is set)
|-F|    -F                  Farsi mode ('fkmap' and 'rightleft' are set)
|-H|    -H                  Hebrew mode ('hkmap' and 'rightleft' are set)
|-V|    -V                  Verbose, give informative messages
|-C|    -C                  Compatible, set the 'compatible' option
|-N|    -N                  Nocompatible, reset the 'compatible' option
|-r|    -r                  give list of swap files
|-r|    -r {file} ..        recover aborted edit session
|-n|    -n                  do not create a swap file
|-o|    -o [num]            open [num] windows (default: one for each file)
|-f|    -f                  GUI: foreground process, don't fork
                            Amiga: do not restart VIM to open a window (for
                                e.g., mail)
|-s|    -s {scriptin}       first read commands from the file {scriptin}
|-w|    -w {scriptout}      write typed chars to file {scriptout} (append)
|-W|    -W {scriptout}      write typed chars to file {scriptout} (overwrite)
|-T|    -T {terminal}       set terminal name
|-d|    -d {device}         Amiga: open {device} to be used as a console
|-u|    -u {vimrc}          read inits from {vimrc} instead of other inits
|-U|    -U {gvimrc}         idem, for when starting the GUI
|-i|    -i {viminfo}        read info from {viminfo} instead of other files
|---|   --                  end of options, other arguments are file names
|--help|    --help          show list of arguments and exit
|--version| --version       show version info and exit
|--|    -                   Read file from stdin.
------------------------------------------------------------------------------
*Q_ed*          Editing a file
```

```
            Without !: Fail if changes has been made to the current buffer.
               With !: Discard any changes to the current buffer.
|:edit_f|  :e[dit][!] {file}    Edit {file}.
|:edit|    :e[dit][!]           Reload the current file.
|:enew|    :ene[w][!]           Edit a new, unnamed buffer.
|:find|    :fin[d][!] {file}    Find {file} in 'path' and edit it.

|CTRL-^|   N   CTRL-^           Edit alternate file N (equivalent to ":e #N").
|gf|           gf  or ]f        Edit the file whose name is under the cursor
|:pwd|     :pwd                 Print the current directory name.
|:cd|      :cd [path]           Change the current directory to [path].
|:cd-|     :cd -                Back to previous current directory.
|:file|    :f[ile]              Print the current file name and the cursor
                                   position.
|:file|    :f[ile] {name}       Set the current file name to {name}.
|:files|   :files               Show alternate file names.
------------------------------------------------------------------------------
*Q_fl*          Using the argument list                 |argument-list|

|:args|    :ar[gs]              Print the argument list, with the current file
                                   in "[]".
|:all|     :all  or :sall       Open a window for every file in the arg list.
|:wn|      :wn[ext][!]          Write file and edit next file.
|:wn|      :wn[ext][!] {file}   Write to {file} and edit next file, unless
                                   {file} exists.  With !, overwrite existing
                                   file.
|:wN|      :wN[ext][!] [file]   Write file and edit previous file.

             in current window    in new window ~
|:argument|  :argu[ment] N        :sar[gument] N       Edit file N
|:next|      :n[ext]              :sn[ext]             Edit next file
|:next_f|    :n[ext] {arglist}    :sn[ext] {arglist}   define new arg list
                                                          and edit first file
|:Next|      :N[ext]              :sN[ext]             Edit previous file
|:first|     :fir[st]             :sfir[st]            Edit first file
|:last|      :la[st]              :sla[st]             Edit last file
------------------------------------------------------------------------------
*Q_wq*          Writing and quitting

|:w|        :[range]w[rite][!]          Write to the current file.
|:w_f|      :[range]w[rite] {file}      Write to {file}, unless it already
                                           exists.
|:w_f|      :[range]w[rite]! {file}     Write to {file}.  Overwrite an existing
                                           file.
|:w_a|      :[range]w[rite][!] >>       Append to the current file.
|:w_a|      :[range]w[rite][!] >> {file}  Append to {file}.
|:w_c|      :[range]w[rite] !{cmd}      Execute {cmd} with [range] lines as
                                           standard input.
|:up|       :[range]up[date][!]         write to current file if modified
|:wall|     :wa[ll][!]                  write all changed buffers

|:q|        :q[uit]             Quit current buffer, unless changes have been
                                   made.  Exit Vim when there are no other
                                   non-help buffers
|:q|        :q[uit]!            Quit current buffer always, discard any
```

```
                                   changes.  Exit Vim when there are no other
                                   non-help buffers
|:qa|     :qa[ll]               Exit Vim, unless changes have been made.
|:qa|     :qa[ll]!              Exit Vim always, discard any changes.
|:cq|     :cq                   Quit without writing and return error code.

|:wq|     :wq[!]                Write the current file and exit.
|:wq|     :wq[!] {file}         Write to {file} and exit.
|:xit|    :x[it][!] [file]      Like ":wq" but write only when changes have
                                   been made
|ZZ|         ZZ                 Same as ":x".
|ZQ|         ZQ                 Same as ":q!".
|:xall|   :xa[ll][!]  or :wqall[!]
                                Write all changed buffers and exit

|:stop|   :st[op][!]            Suspend VIM or start new shell.  If 'aw' option
                                   is set and [!] not given write the buffer.
|CTRL-Z|     CTRL-Z             Same as ":stop"
------------------------------------------------------------------------------
*Q_ac*          Automatic Commands

|viminfo-file|  Read registers, marks, history at startup, save when exiting.

|:rviminfo|     :rv[iminfo] [file]      Read info from viminfo file [file]
|:rviminfo|     :rv[iminfo]! [file]     idem, overwrite existing info
|:wviminfo|     :wv[iminfo] [file]      Add info to viminfo file [file]
|:wviminfo|     :wv[iminfo]! [file]     Write info to viminfo file [file]

|modeline|      Automatic option setting when editing a file

|modeline|      vim:{set-arg}: ..       In the first and last lines of the
                                        file (see 'ml' option), {set-arg} is
                                        given as an argument to ":set"

|autocommand|   Automatic execution of commands on certain events.

|:autocmd|      :au                     List all autocommands
|:autocmd|      :au {event}             List all autocommands for {event}
|:autocmd|      :au {event} {pat}       List all autocommands for {event} with
                                        {pat}
|:autocmd|      :au {event} {pat} {cmd} Enter new autocommands for {event}
                                        with {pat}
|:autocmd|      :au!                    Remove all autocommands
|:autocmd|      :au! {event}            Remove all autocommands for {event}
|:autocmd|      :au! * {pat}            Remove all autocommands for {pat}
|:autocmd|      :au! {event} {pat}      Remove all autocommands for {event}
                                        with {pat}
|:autocmd|      :au! {event} {pat} {cmd}  Remove all autocommands for {event}
                                        with {pat} and enter new one
------------------------------------------------------------------------------
*Q_wi*          Multi-window commands

|CTRL-W_s|      CTRL-W s  or  :split    Split window into two parts
|:split_f|      :split {file}           Split window and edit {file} in one of
                                           them
```

```
|:vsplit|       :vsplit {file}          Same, but split vertically
|:vertical|     :vertical {cmd}         Make {cmd} split vertically

|:sfind|        :sf[ind] {file}         Split window, find {file} in 'path'
                                           and edit it.
|CTRL-W_]|      CTRL-W ]                Split window and jump to tag under
                                           cursor
|CTRL-W_f|      CTRL-W f                Split window and edit file name under
                                           the cursor
|CTRL-W_^|      CTRL-W ^                Split window and edit alternate file
|CTRL-W_n|      CTRL-W n  or  :new      Create new empty window
|CTRL-W_q|      CTRL-W q  or  :q[uit]   Quit editing and close window
|CTRL-W_c|      CTRL-W c  or  :cl[ose]  Make buffer hidden and close window
|CTRL-W_o|      CTRL-W o  or  :on[ly]   Make current window only one on the
                                           screen

|CTRL-W_j|      CTRL-W j                Move cursor to window below
|CTRL-W_k|      CTRL-W k                Move cursor to window above
|CTRL-W_CTRL-W| CTRL-W CTRL-W           Move cursor to window below (wrap)
|CTRL-W_W|      CTRL-W W                Move cursor to window above (wrap)
|CTRL-W_t|      CTRL-W t                Move cursor to top window
|CTRL-W_b|      CTRL-W b                Move cursor to bottom window
|CTRL-W_p|      CTRL-W p                Move cursor to previous active window

|CTRL-W_r|      CTRL-W r                Rotate windows downwards
|CTRL-W_R|      CTRL-W R                Rotate windows upwards
|CTRL-W_x|      CTRL-W x                Exchange current window with next one

|CTRL-W_=|      CTRL-W =                Make all windows equal height
|CTRL-W_-|      CTRL-W -                Decrease current window height
|CTRL-W_+|      CTRL-W +                Increase current window height
|CTRL-W__|      CTRL-W _                Set current window height (default:
                                           very high)
------------------------------------------------------------------------------
*Q_bu*          Buffer list commands

|:buffers|      :buffers  or  :files    list all known buffer and file names

|:ball|         :ball     or  :sball    edit all args/buffers
|:unhide|       :unhide   or  :sunhide  edit all loaded buffers

|:badd|         :badd {fname}           add file name {fname} to the list
|:bunload|      :bunload[!] [N]         unload buffer [N] from memory
|:bdelete|      :bdelete[!] [N]         unload buffer [N] and delete it from
                                           the buffer list

              in current window   in new window ~
|:buffer|     :[N]buffer [N]      :[N]sbuffer [N]     to arg/buf N
|:bnext|      :[N]bnext [N]       :[N]sbnext [N]      to Nth next arg/buf
|:bNext|      :[N]bNext [N]       :[N]sbNext [N]      to Nth previous arg/buf
|:bprevious|  :[N]bprevious [N]   :[N]sbprevious [N]  to Nth previous arg/buf
|:bfirst|     :bfirst             :sbfirst            to first arg/buf
|:blast|      :blast              :sblast             to last arg/buf
|:bmodified|  :[N]bmod [N]        :[N]sbmod [N]       to Nth modified buf
------------------------------------------------------------------------------
```

```
*Q_sy*          Syntax Highlighting

|:syn-on|       :syntax on              start using syntax highlighting
|:syn-off|      :syntax off             stop using syntax highlighting

|:syn-keyword|  :syntax keyword {group-name} {keyword} ..
                                        add a syntax keyword item
|:syn-match|    :syntax match {group-name} {pattern} ...
                                        add syntax match item
|:syn-region|   :syntax region {group-name} {pattern} ...
                                        add syntax region item
|:syn-sync|     :syntax sync [ccomment | lines {N} | ...]
                                        tell syntax how to sync
|:syntax|       :syntax [list]          list current syntax items
|:syn-clear|    :syntax clear           clear all syntax info

|:highlight|    :highlight clear        clear all highlight info
|:highlight|    :highlight {group-name} {key}={arg} ..
                                        set highlighting for {group-name}

|:filetype|     :filetype on            switch on file type detection, without
                                        syntax highlighting
|:filetype|     :filetype plugin indent on
                                        switch on file type detection, with
                                        automatic indenting and settings
------------------------------------------------------------------------------
*Q_gu*          GUI commands

|:gui|          :gui                    UNIX: start the GUI
|:gui|          :gui {fname} ..         idem, and edit {fname} ..

|:menu|         :menu                   list all menus
|:menu|         :menu {mpath}           list menus starting with {mpath}
|:menu|         :menu {mpath} {rhs}     add menu {mpath}, giving {lhs}
|:menu|         :menu {pri} {mpath} {rhs}
                                        idem, with priorities {pri}
|:menu|         :menu ToolBar.{name} {rhs}
                                        add toolbar item, giving {lhs}
|:tmenu|        :tmenu {mpath} {text}   add tooltip to menu {mpath}
|:unmenu|       :unmenu {mpath}         remove menu {mpath}
------------------------------------------------------------------------------
*Q_fo*          Folding

|'foldmethod'|  set foldmethod=manual   manual folding
                set foldmethod=indent   folding by indent
                set foldmethod=expr     folding by 'foldexpr'
                set foldmethod=syntax   folding by syntax regions
                set foldmethod=marker   folding by 'foldmarkers'

|zf|            zf{motion}              operator: Define a fold manually
|:fold|         :{range}fold            define a fold for {range} lines
|zd|            zd                      delete one fold under the cursor
|zD|            zD                      delete all folds under the cursor

|zo|            zo                      open one fold under the cursor
```

```
|zO|            zO                      open all folds under the cursor
|zc|            zc                      close one fold under the cursor
|zC|            zC                      close all folds under the cursor

|zm|            zm                      fold more: decrease 'foldlevel'
|zM|            zM                      close all folds: make 'foldlevel' zero
|zr|            zr                      reduce folding: increase 'foldlevel'
|zR|            zR                      open all folds: make 'foldlevel' max.

|zn|            zn                      fold none: reset 'foldenable'
|zN|            zN                      fold normal set 'foldenable'
|zi|            zi                      invert 'foldenable'

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