```
'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
'relativenumber' 'rnu'    show relative line number in front of each line
'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'
'undodir'       'udir'    where to store undo files
'undofile'      'udf'     save undo information in a file
'undolevels'    'ul'      maximum number of changes that can be undone
'undoreload'    'ur'      max nr of lines to save for undo on a buffer reload
'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'   'wiv'     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
'wildignorecase' 'wic'    ignore case when completing file names
'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
'window'        'wi'      nr of lines to scroll for CTRL-F and CTRL-B
'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, CB0,
                                     CI350, 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 one

|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 seven are typed literally; these are not special keys!
|:<abuf>|    <abuf>      buffer number, for use in an autocommand (only where a
                             file name is expected)
|:<afile>|   <afile>     file name, for use in an autocommand (only where a
                             file name is expected)
|:<amatch>| <amatch>     what matched with the pattern, for use in an
                             autocommand (only where a file name is expected)
|:<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)
|:<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 have 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|  :ar[gument] 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 {rhs}
|:menu|          :menu {pri} {mpath} {rhs}
                                         idem, with priorities {pri}
|:menu|          :menu ToolBar.{name} {rhs}
                                         add toolbar item, giving {rhs}
|: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 'foldmarker'

|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'
```

top - main help file