General subjects ~
|intro.txt| general introduction to Vim; notation used in help files
|help.txt| overview and quick reference (this file)
|helphelp.txt| about using the help files
|index.txt| alphabetical index of all commands
|help-tags| all the tags you can jump to (index of tags)
|howto.txt| how to do the most common editing tasks
|tips.txt| various tips on using Vim
|message.txt| (error) messages and explanations
|quotes.txt| remarks from users of Vim
|todo.txt| known problems and desired extensions
|develop.txt| development of Vim
|debug.txt| debugging Vim itself
|uganda.txt| Vim distribution conditions and what to do with your money

==
intro.txt   For Vim version 8.0. Last change: 2017 Sep 04

                  VIM REFERENCE MANUAL   by Bram Moolenaar

Introduction to Vim   *ref* *reference*

==
1. Introduction   *intro*

Vim stands for Vi IMproved. It used to be Vi IMitation, but there are so many
improvements that a name change was appropriate. Vim is a text editor which
includes almost all the commands from the Unix program "Vi" and a lot of new
ones. It is very useful for editing programs and other plain text.
   All commands are given with the keyboard. This has the advantage that you
can keep your fingers on the keyboard and your eyes on the screen. For those
who want it, there is mouse support and a GUI version with scrollbars and
menus (see |gui.txt|).

An overview of this manual can be found in the file "help.txt", |help.txt|.
It can be accessed from within Vim with the <Help> or <F1> key and with the
|:help| command (just type ":help", without the bars or quotes).
   The 'helpfile' option can be set to the name of the help file, in case it
is not located in the default place. You can jump to subjects like with tags:
Use CTRL-] to jump to a subject under the cursor, use CTRL-T to jump back.

Throughout this manual the differences between Vi and Vim are mentioned in
curly braces, like this: {Vi does not have on-line help}. See |vi_diff.txt|
for a summary of the differences between Vim and Vi.

This manual refers to Vim on various machines. There may be small differences
between different computers and terminals. Besides the remarks given in this
document, there is a separate document for each supported system, see
|sys-file-list|.

                                                        pronounce

Vim is pronounced as one word, like Jim, not vi-ai-em. It's written with a
capital, since it's a name, again like Jim.

This manual is a reference for all the Vim commands and options. This is not
an introduction to the use of Vi or Vim, it gets a bit complicated here and
there. For beginners, there is a hands-on |tutor|. To learn using Vim, read
the user manual |usr_toc.txt|.

book
There are many books on Vi that contain a section for beginners. There are
two books I can recommend:

"Vim - Vi Improved" by Steve Oualline

This is the very first book completely dedicated to Vim. It is very good for
beginners. The most often used commands are explained with pictures and
examples. The less often used commands are also explained, the more advanced
features are summarized. There is a comprehensive index and a quick
reference. Parts of this book have been included in the user manual
|frombook|.
Published by New Riders Publishing. ISBN: 0735710015
For more information try one of these:
http://iccf-holland.org/click5.html
http://www.vim.org/iccf/click5.html

"Learning the Vi editor" by Linda Lamb and Arnold Robbins

This is a book about Vi that includes a chapter on Vim (in the sixth edition).
The first steps in Vi are explained very well. The commands that Vim adds are
only briefly mentioned. There is also a German translation.
Published by O'Reilly. ISBN: 1-56592-426-6.

==
2. Vim on the internet *internet*

www *WWW* *faq* *FAQ* *distribution* *download*
The Vim pages contain the most recent information about Vim. They also
contain links to the most recent version of Vim. The FAQ is a list of
Frequently Asked Questions. Read this if you have problems.

Vim home page: http://www.vim.org/
Vim FAQ: http://vimdoc.sf.net/
Downloading: ftp://ftp.vim.org/pub/vim/MIRRORS

Usenet News group where Vim is discussed: *news* *usenet*
comp.editors
This group is also for other editors. If you write about Vim, don't forget to
mention that.

mail-list *maillist*
There are several mailing lists for Vim:
<vim@vim.org> *vim-use* *vim_use*
For discussions about using existing versions of Vim: Useful mappings,
questions, answers, where to get a specific version, etc. There are
quite a few people watching this list and answering questions, also
for beginners. Don't hesitate to ask your question here.
<vim-dev@vim.org> *vim-dev* *vim_dev* *vimdev*
For discussions about changing Vim: New features, porting, patches,
beta-test versions, etc.
<vim-announce@vim.org> *vim-announce* *vim_announce*
Announcements about new versions of Vim; also for beta-test versions

        and ports to different systems. This is a read-only list.
<vim-mac@vim.org>   *vim-mac* *vim_mac*
        For discussions about using and improving the Macintosh version of
        Vim.

See http://www.vim.org/maillist.php for the latest information.

NOTE:
- You can only send messages to these lists if you have subscribed!
- You need to send the messages from the same location as where you subscribed
  from (to avoid spam mail).
- Maximum message size is 40000 characters.

                                                 subscribe-maillist
If you want to join, send a message to
        <vim-subscribe@vim.org>
Make sure that your "From:" address is correct. Then the list server will
give you help on how to subscribe.

                                                 maillist-archive
For more information and archives look on the Vim maillist page:
http://www.vim.org/maillist.php

Bug reports: *bugs* *bug-reports* *bugreport.vim*

There are two ways to report bugs, both work:
1. Send bug reports to: Vim Developers <vim-dev@vim.org>
   This is a maillist, you need to become a member first and many people will
   see the message. If you don't want that, e.g. because it is a security
   issue, send it to <bugs@vim.org>, this only goes to the Vim maintainer
   (that's Bram).
2. Open issue on GitHub: https://github.com/vim/vim/issues
   The text will be forwarded to the vim-dev maillist.

Please be brief; all the time that is spent on answering mail is subtracted
from the time that is spent on improving Vim! Always give a reproducible
example and try to find out which settings or other things trigger the bug.

Preferably start Vim with: >
        vim --clean -u reproduce.vim
Where reproduce.vim is a script that reproduces the problem. Try different
machines, if relevant (is this an MS-Windows specific bug perhaps?).

Send me patches if you can!

It will help to include information about the version of Vim you are using and
your setup. You can get the information with this command: >
   :so $VIMRUNTIME/bugreport.vim
This will create a file "bugreport.txt" in the current directory, with a lot
of information of your environment. Before sending this out, check if it
doesn't contain any confidential information!

If Vim crashes, please try to find out where. You can find help on this here:
|debug.txt|.

In case of doubt or when you wonder if the problem has already been fixed but
you can't find a fix for it, become a member of the vim-dev maillist and ask
your question there. |maillist|

                                                        year-2000 *Y2K*
Since Vim internally doesn't use dates for editing, there is no year 2000

problem to worry about. Vim does use the time in the form of seconds since January 1st 1970. It is used for a time-stamp check of the edited file and the swap file, which is not critical and should only cause warning messages.

There might be a year 2038 problem, when the seconds don't fit in a 32 bit int anymore. This depends on the compiler, libraries and operating system. Specifically, time_t and the ctime() function are used. And the time_t is stored in four bytes in the swap file. But that's only used for printing a file date/time for recovery, it will never affect normal editing.

The Vim strftime() function directly uses the strftime() system function. localtime() uses the time() system function. getftime() uses the time returned by the stat() system function. If your system libraries are year 2000 compliant, Vim is too.

The user may create scripts for Vim that use external commands. These might introduce Y2K problems, but those are not really part of Vim itself.

==
3. Credits *credits* *author* *Bram* *Moolenaar*

Most of Vim was written by Bram Moolenaar <Bram@vim.org>.

Parts of the documentation come from several Vi manuals, written by:
W.N. Joy
Alan P.W. Hewett
Mark Horton

The Vim editor is based on Stevie and includes (ideas from) other software, worked on by the people mentioned here. Other people helped by sending me patches, suggestions and giving feedback about what is good and bad in Vim.

Vim would never have become what it is now, without the help of these people!

Ron Aaron	Win32 GUI changes
Mohsin Ahmed	encryption
Zoltan Arpadffy	work on VMS port
Tony Andrews	Stevie
Gert van Antwerpen	changes for DJGPP on MS-DOS
Berkeley DB(3)	ideas for swap file implementation
Keith Bostic	Nvi
Walter Briscoe	Makefile updates, various patches
Ralf Brown	SPAWNO library for MS-DOS
Robert Colon	many useful remarks
Marcin Dalecki	GTK+ GUI port, toolbar icons, gettext()
Kayhan Demirel	sent me news in Uganda
Chris & John Downey	xvi (ideas for multi-windows version)
Henk Elbers	first VMS port
Daniel Elstner	GTK+ 2 port
Eric Fischer	Mac port, 'cindent', and other improvements
Benji Fisher	Answering lots of user questions
Bill Foster	Athena GUI port
Google	Lets me work on Vim one day a week
Loic Grenie	xvim (ideas for multi windows version)
Sven Guckes	Vim promoter and previous WWW page maintainer
Darren Hiebert	Exuberant ctags
Jason Hildebrand	GTK+ 2 port
Bruce Hunsaker	improvements for VMS port
Andy Kahn	Cscope support, GTK+ GUI port
Oezguer Kesim	Maintainer of Vim Mailing Lists
Axel Kielhorn	work on the Macintosh port
Steve Kirkendall	Elvis

```
        Roger Knobbe            original port to Windows NT
        Sergey Laskavy          Vim's help from Moscow
        Felix von Leitner       Previous maintainer of Vim Mailing Lists
        David Leonard           Port of Python extensions to Unix
        Avner Lottem            Edit in right-to-left windows
        Flemming Madsen         X11 client-server, various features and patches
        Tony Mechelynck         answers many user questions
        Paul Moore              Python interface extensions, many patches
        Katsuhito Nagano        Work on multi-byte versions
        Sung-Hyun Nam           Work on multi-byte versions
        Vince Negri             Win32 GUI and generic console enhancements
        Steve Oualline          Author of the first Vim book |frombook|
        Dominique Pelle         valgrind reports and many fixes
        A.Politz                Many bug reports and some fixes
        George V. Reilly        Win32 port, Win32 GUI start-off
        Stephen Riehm           bug collector
        Stefan Roemer           various patches and help to users
        Ralf Schandl            IBM OS/390 port
        Olaf Seibert            DICE and BeBox version, regexp improvements
        Mortaza Shiran          Farsi patches
        Peter da Silva          termlib
        Paul Slootman           OS/2 port
        Henry Spencer           regular expressions
        Dany St-Amant           Macintosh port
        Tim Thompson            Stevie
        G. R. (Fred) Walter     Stevie
        Sven Verdoolaege        Perl interface
        Robert Webb             Command-line completion, GUI versions, and
                                lots of patches
        Ingo Wilken             Tcl interface
        Mike Williams           PostScript printing
        Juergen Weigert         Lattice version, AUX improvements, UNIX and
                                MS-DOS ports, autoconf
        Stefan 'Sec' Zehl       Maintainer of vim.org
        Yasuhiro Matsumoto      many MS-Windows improvements
        Ken Takata              fixes and features
        Kazunobu Kuriyama       GTK 3
        Christian Brabandt      many fixes, features, user support, etc.

I wish to thank all the people that sent me bug reports and suggestions.  The
list is too long to mention them all here.  Vim would not be the same without
the ideas from all these people: They keep Vim alive!
*love* *peace* *friendship* *gross-national-happiness*


In this documentation there are several references to other versions of Vi:
                                                        *Vi* *vi*
Vi      "the original".  Without further remarks this is the version
        of Vi that appeared in Sun OS 4.x.  ":version" returns
        "Version 3.7, 6/7/85".  Sometimes other versions are referred
        to.  Only runs under Unix.  Source code only available with a
        license.  More information on Vi can be found through:
                http://vi-editor.org    [doesn't currently work...]
                                                        *Posix*
Posix   From the IEEE standard 1003.2, Part 2: Shell and utilities.
        Generally known as "Posix".  This is a textual description of
        how Vi is supposed to work.
        See |posix-compliance|.
                                                        *Nvi*
Nvi     The "New" Vi.  The version of Vi that comes with BSD 4.4 and FreeBSD.
        Very good compatibility with the original Vi, with a few extensions.
        The version used is 1.79.  ":version" returns "Version 1.79
```

(10/23/96)". There has been no release the last few years, although there is a development version 1.81.
Source code is freely available.

Elvis

Elvis Another Vi clone, made by Steve Kirkendall. Very compact but isn't as flexible as Vim.
The version used is 2.1. It is still being developed. Source code is freely available.

==========

4. Notation *notation*

When syntax highlighting is used to read this, text that is not typed literally is often highlighted with the Special group. These are items in [], {} and <>, and CTRL-X.

Note that Vim uses all possible characters in commands. Sometimes the [], {} and <> are part of what you type, the context should make this clear.

[] Characters in square brackets are optional.

count *[count]*

[count] An optional number that may precede the command to multiply or iterate the command. If no number is given, a count of one is used, unless otherwise noted. Note that in this manual the [count] is not mentioned in the description of the command, but only in the explanation. This was done to make the commands easier to look up. If the 'showcmd' option is on, the (partially) entered count is shown at the bottom of the window. You can use <Del> to erase the last digit (|N<Del>|).

[quotex]

["x] An optional register designation where text can be stored. See |registers|. The x is a single character between 'a' and 'z' or 'A' and 'Z' or '"', and in some cases (with the put command) between '0' and '9', '%', '#', or others. The uppercase and lowercase letter designate the same register, but the lowercase letter is used to overwrite the previous register contents, while the uppercase letter is used to append to the previous register contents. Without the ""x" or with """" the stored text is put into the unnamed register.

{}

{} Curly braces denote parts of the command which must appear, but which can take a number of different values. The differences between Vim and Vi are also given in curly braces (this will be clear from the context).

{char1-char2}

{char1-char2} A single character from the range char1 to char2. For example: {a-z} is a lowercase letter. Multiple ranges may be concatenated. For example, {a-zA-Z0-9} is any alphanumeric character.

{motion} *movement*

{motion} A command that moves the cursor. These are explained in |motion.txt|. Examples:

w	to start of next word
b	to begin of current word
4j	four lines down
/The<CR>	to next occurrence of "The"

```
                This is used after an |operator| command to move over the text
                that is to be operated upon.
                - If the motion includes a count and the operator also has a
                  count, the two counts are multiplied.  For example: "2d3w"
                  deletes six words.
                - The motion can be backwards, e.g. "db" to delete to the
                  start of the word.
                - The motion can also be a mouse click.  The mouse is not
                  supported in every terminal though.
                - The ":omap" command can be used to map characters while an
                  operator is pending.
                - Ex commands can be used to move the cursor.  This can be
                  used to call a function that does some complicated motion.
                  The motion is always characterwise exclusive, no matter
                  what ":" command is used.  This means it's impossible to
                  include the last character of a line without the line break
                  (unless 'virtualedit' is set).
                  If the Ex command changes the text before where the operator
                  starts or jumps to another buffer the result is
                  unpredictable.  It is possible to change the text further
                  down.  Jumping to another buffer is possible if the current
                  buffer is not unloaded.

                                                          *{Visual}*
{Visual}        A selected text area.  It is started with the "v", "V", or
                CTRL-V command, then any cursor movement command can be used
                to change the end of the selected text.
                This is used before an |operator| command to highlight the
                text that is to be operated upon.
                See |Visual-mode|.

                                                          *<character>*
<character>     A special character from the table below, optionally with
                modifiers, or a single ASCII character with modifiers.

                                                          *'character'*
'c'             A single ASCII character.

                                                          *CTRL-{char}*
CTRL-{char}     {char} typed as a control character; that is, typing {char}
                while holding the CTRL key down.  The case of {char} does not
                matter; thus CTRL-A and CTRL-a are equivalent.  But on some
                terminals, using the SHIFT key will produce another code,
                don't use it then.

                                                          *'option'*
'option'        An option, or parameter, that can be set to a value, is
                enclosed in single quotes.  See |options|.

                                                          *quotecommandquote*
"command"       A reference to a command that you can type is enclosed in
                double quotes.
`command`       New style command, this distinguishes it from other quoted
                text and strings.

                                        *key-notation* *key-codes* *keycodes*
These names for keys are used in the documentation.  They can also be used
with the ":map" command (insert the key name by pressing CTRL-K and then the
key you want the name for).

notation        meaning             equivalent  decimal value(s)        ~
-----------------------------------------------------------------------
```

```
<Nul>           zero                    CTRL-@    0 (stored as 10) *<Nul>*
<BS>            backspace               CTRL-H    8     *backspace*
<Tab>           tab                     CTRL-I    9     *tab* *Tab*
                                                        *linefeed*
<NL>            linefeed                CTRL-J   10 (used for <Nul>)
<FF>            formfeed                CTRL-L   12     *formfeed*
<CR>            carriage return         CTRL-M   13     *carriage-return*
<Return>        same as <CR>                             *<Return>*
<Enter>         same as <CR>                             *<Enter>*
<Esc>           escape                  CTRL-[   27     *escape* *<Esc>*
<Space>         space                            32     *space*
<lt>            less-than               <        60     *<lt>*
<Bslash>        backslash               \        92     *backslash* *<Bslash>*
<Bar>           vertical bar            |       124     *<Bar>*
<Del>           delete                          127
<CSI>           command sequence intro  ALT-Esc 155     *<CSI>*
<xCSI>          CSI when typed in the GUI                *<xCSI>*

<EOL>           end-of-line (can be <CR>, <LF> or <CR><LF>,
                depends on system and 'fileformat')     *<EOL>*

<Up>            cursor-up                       *cursor-up* *cursor_up*
<Down>          cursor-down                     *cursor-down* *cursor_down*
<Left>          cursor-left                     *cursor-left* *cursor_left*
<Right>         cursor-right                    *cursor-right* *cursor_right*
<S-Up>          shift-cursor-up
<S-Down>        shift-cursor-down
<S-Left>        shift-cursor-left
<S-Right>       shift-cursor-right
<C-Left>        control-cursor-left
<C-Right>       control-cursor-right
<F1> - <F12>    function keys 1 to 12           *function_key* *function-key*
<S-F1> - <S-F12> shift-function keys 1 to 12    *<S-F1>*
<Help>          help key
<Undo>          undo key
<Insert>        insert key
<Home>          home                            *home*
<End>           end                             *end*
<PageUp>        page-up                         *page_up* *page-up*
<PageDown>      page-down                       *page_down* *page-down*
<kHome>         keypad home (upper left)        *keypad-home*
<kEnd>          keypad end (lower left)         *keypad-end*
<kPageUp>       keypad page-up (upper right)    *keypad-page-up*
<kPageDown>     keypad page-down (lower right)  *keypad-page-down*
<kPlus>         keypad +                        *keypad-plus*
<kMinus>        keypad -                        *keypad-minus*
<kMultiply>     keypad *                        *keypad-multiply*
<kDivide>       keypad /                        *keypad-divide*
<kEnter>        keypad Enter                    *keypad-enter*
<kPoint>        keypad Decimal point            *keypad-point*
<k0> - <k9>     keypad 0 to 9                   *keypad-0* *keypad-9*
<S-...>         shift-key                       *shift* *<S-*
<C-...>         control-key                     *control* *ctrl* *<C-*
<M-...>         alt-key or meta-key             *meta* *alt* *<M-*
<A-...>         same as <M-...>                 *<A-*
<D-...>         command-key (Macintosh only)    *<D-*
<t_xx>          key with "xx" entry in termcap
-------------------------------------------------------------------------

Note: The shifted cursor keys, the help key, and the undo key are only
available on a few terminals.  On the Amiga, shifted function key 10 produces
a code (CSI) that is also used by key sequences.  It will be recognized only
```

```
after typing another key.

Note: There are two codes for the delete key.  127 is the decimal ASCII value
for the delete key, which is always recognized.  Some delete keys send another
value, in which case this value is obtained from the termcap entry "kD".  Both
values have the same effect.  Also see |:fixdel|.

Note: The keypad keys are used in the same way as the corresponding "normal"
keys.  For example, <kHome> has the same effect as <Home>.  If a keypad key
sends the same raw key code as its non-keypad equivalent, it will be
recognized as the non-keypad code.  For example, when <kHome> sends the same
code as <Home>, when pressing <kHome> Vim will think <Home> was pressed.
Mapping <kHome> will not work then.

                                                                      *<>*
Examples are often given in the <> notation.  Sometimes this is just to make
clear what you need to type, but often it can be typed literally, e.g., with
the ":map" command.  The rules are:
 1.  Any printable characters are typed directly, except backslash and '<'
 2.  A backslash is represented with "\\", double backslash, or "<Bslash>".
 3.  A real '<' is represented with "\<" or "<lt>".  When there is no
     confusion possible, a '<' can be used directly.
 4.  "<key>" means the special key typed.  This is the notation explained in
     the table above.  A few examples:
           <Esc>               Escape key
           <C-G>               CTRL-G
           <Up>                cursor up key
           <C-LeftMouse>       Control- left mouse click
           <S-F11>             Shifted function key 11
           <M-a>               Meta- a  ('a' with bit 8 set)
           <M-A>               Meta- A  ('A' with bit 8 set)
           <t_kd>              "kd" termcap entry (cursor down key)

If you want to use the full <> notation in Vim, you have to make sure the '<'
flag is excluded from 'cpoptions' (when 'compatible' is not set, it already is
by default). >
        :set cpo-=<
The <> notation uses <lt> to escape the special meaning of key names.  Using a
backslash also works, but only when 'cpoptions' does not include the 'B' flag.

Examples for mapping CTRL-H to the six characters "<Home>": >
        :imap <C-H> \<Home>
        :imap <C-H> <lt>Home>
The first one only works when the 'B' flag is not in 'cpoptions'.  The second
one always works.
To get a literal "<lt>" in a mapping: >
        :map <C-L> <lt>lt>

For mapping, abbreviation and menu commands you can then copy-paste the
examples and use them directly.  Or type them literally, including the '<' and
'>' characters.  This does NOT work for other commands, like ":set" and
":autocmd"!

==============================================================================
5. Modes, introduction                          *vim-modes-intro* *vim-modes*

Vim has seven BASIC modes:

                                          *Normal* *Normal-mode* *command-mode*
Normal mode             In Normal mode you can enter all the normal editor
                        commands.  If you start the editor you are in this
                        mode (unless you have set the 'insertmode' option,
```

```
                        see below).  This is also known as command mode.

Visual mode             This is like Normal mode, but the movement commands
                        extend a highlighted area.  When a non-movement
                        command is used, it is executed for the highlighted
                        area.  See |Visual-mode|.
                        If the 'showmode' option is on "-- VISUAL --" is shown
                        at the bottom of the window.

Select mode             This looks most like the MS-Windows selection mode.
                        Typing a printable character deletes the selection
                        and starts Insert mode.  See |Select-mode|.
                        If the 'showmode' option is on "-- SELECT --" is shown
                        at the bottom of the window.

Insert mode             In Insert mode the text you type is inserted into the
                        buffer.  See |Insert-mode|.
                        If the 'showmode' option is on "-- INSERT --" is shown
                        at the bottom of the window.

Command-line mode       In Command-line mode (also called Cmdline mode) you
Cmdline mode            can enter one line of text at the bottom of the
                        window.  This is for the Ex commands, ":", the pattern
                        search commands, "?" and "/", and the filter command,
                        "!".  |Cmdline-mode|

Ex mode                 Like Command-line mode, but after entering a command
                        you remain in Ex mode.  Very limited editing of the
                        command line.  |Ex-mode|

Terminal-Job mode       Interacting with a job in a terminal window.  Typed
                        keys go to the job and the job output is displayed in
                        the terminal window.  See |terminal| about how to
                        switch to other modes.

There are seven ADDITIONAL modes.  These are variants of the BASIC modes:

                                *Operator-pending* *Operator-pending-mode*
Operator-pending mode   This is like Normal mode, but after an operator
                        command has started, and Vim is waiting for a {motion}
                        to specify the text that the operator will work on.

Replace mode            Replace mode is a special case of Insert mode.  You
                        can do the same things as in Insert mode, but for
                        each character you enter, one character of the existing
                        text is deleted.  See |Replace-mode|.
                        If the 'showmode' option is on "-- REPLACE --" is
                        shown at the bottom of the window.

Virtual Replace mode    Virtual Replace mode is similar to Replace mode, but
                        instead of file characters you are replacing screen
                        real estate.  See |Virtual-Replace-mode|.
                        If the 'showmode' option is on "-- VREPLACE --" is
                        shown at the bottom of the window.

Insert Normal mode      Entered when CTRL-O given in Insert mode.  This is
                        like Normal mode, but after executing one command Vim
                        returns to Insert mode.
                        If the 'showmode' option is on "-- (insert) --" is
                        shown at the bottom of the window.

Terminal-Normal mode    Using Normal mode in a terminal window.  Making
```

changes is impossible. Use an insert command, such as "a" or "i", to return to Terminal-Job mode.

Insert Visual mode — Entered when starting a Visual selection from Insert mode, e.g., by using CTRL-O and then "v", "V" or CTRL-V. When the Visual selection ends, Vim returns to Insert mode.
If the 'showmode' option is on "-- (insert) VISUAL --" is shown at the bottom of the window.

Insert Select mode — Entered when starting Select mode from Insert mode. E.g., by dragging the mouse or <S-Right>.
When the Select mode ends, Vim returns to Insert mode.
If the 'showmode' option is on "-- (insert) SELECT --" is shown at the bottom of the window.

==
6. Switching from mode to mode *mode-switching*

If for any reason you do not know which mode you are in, you can always get back to Normal mode by typing <Esc> twice. This doesn't work for Ex mode though, use ":visual".
You will know you are back in Normal mode when you see the screen flash or hear the bell after you type <Esc>. However, when pressing <Esc> after using CTRL-O in Insert mode you get a beep but you are still in Insert mode, type <Esc> again.

i_esc

FROM mode \ TO mode	Normal	Visual	Select	Insert	Replace	Cmd-line	Ex
Normal		v V ^V	*4	*1	R gR	: / ? !	Q
Visual	*2		^G	c C	--	:	--
Select	*5	^O ^G		*6	--	--	--
Insert	<Esc>	--	--		<Insert>	--	--
Replace	<Esc>	--	--	<Insert>		--	--
Command-line	*3	--	--	:start	--		--
Ex	:vi	--	--	--	--	--	

-- not possible

*1 Go from Normal mode to Insert mode by giving the command "i", "I", "a", "A", "o", "O", "c", "C", "s" or S".
*2 Go from Visual mode to Normal mode by giving a non-movement command, which causes the command to be executed, or by hitting <Esc> "v", "V" or "CTRL-V" (see |v_v|), which just stops Visual mode without side effects.
*3 Go from Command-line mode to Normal mode by:
- Hitting <CR> or <NL>, which causes the entered command to be executed.
- Deleting the complete line (e.g., with CTRL-U) and giving a final <BS>.
- Hitting CTRL-C or <Esc>, which quits the command-line without executing the command.

In the last case <Esc> may be the character defined with the 'wildchar' option, in which case it will start command-line completion. You can ignore that and type <Esc> again. {Vi: when hitting <Esc> the command-line is executed. This is unexpected for most people; therefore it was changed in Vim. But when the <Esc> is part of a mapping, the command-line is executed. If you want the Vi behaviour also when typing <Esc>, use ":cmap ^V<Esc> ^V^M"}
*4 Go from Normal to Select mode by:
- use the mouse to select text while 'selectmode' contains "mouse"
- use a non-printable command to move the cursor while keeping the Shift key pressed, and the 'selectmode' option contains "key"

```
   - use "v", "V" or "CTRL-V" while 'selectmode' contains "cmd"
   - use "gh", "gH" or "g CTRL-H"  |g_CTRL-H|
*5 Go from Select mode to Normal mode by using a non-printable command to move
   the cursor, without keeping the Shift key pressed.
*6 Go from Select mode to Insert mode by typing a printable character.  The
   selection is deleted and the character is inserted.

If the 'insertmode' option is on, editing a file will start in Insert mode.

        *CTRL-\_CTRL-N* *i_CTRL-\_CTRL-N* *c_CTRL-\_CTRL-N* *v_CTRL-\_CTRL-N*
Additionally the command CTRL-\ CTRL-N or <C-\><C-N> can be used to go to
Normal mode from any other mode.  This can be used to make sure Vim is in
Normal mode, without causing a beep like <Esc> would.  However, this does not
work in Ex mode.  When used after a command that takes an argument, such as
|f| or |m|, the timeout set with 'ttimeoutlen' applies.
When focus is in a terminal window, CTRL-\ CTRL-N goes to Normal mode for only
one command, see |t_CTRL-\_CTRL-N|.

        *CTRL-\_CTRL-G* *i_CTRL-\_CTRL-G* *c_CTRL-\_CTRL-G* *v_CTRL-\_CTRL-G*
The command CTRL-\ CTRL-G or <C-\><C-G> can be used to go to Insert mode when
'insertmode' is set.  Otherwise it goes to Normal mode.  This can be used to
make sure Vim is in the mode indicated by 'insertmode', without knowing in
what mode Vim currently is.

                                *Q* *mode-Ex* *Ex-mode* *Ex* *EX* *E501*
Q                       Switch to "Ex" mode.  This is a bit like typing ":"
                        commands one after another, except:
                        - You don't have to keep pressing ":".
                        - The screen doesn't get updated after each command.
                        - There is no normal command-line editing.
                        - Mappings and abbreviations are not used.
                        In fact, you are editing the lines with the "standard"
                        line-input editing commands (<Del> or <BS> to erase,
                        CTRL-U to kill the whole line).
                        Vim will enter this mode by default if it's invoked as
                        "ex" on the command-line.
                        Use the ":vi" command |:visual| to exit "Ex" mode.
                        Note: In older versions of Vim "Q" formatted text,
                        that is now done with |gq|.  But if you use the
                        |vimrc_example.vim| script "Q" works like "gq".

                                *gQ*
gQ                      Switch to "Ex" mode like with "Q", but really behave
                        like typing ":" commands after another.  All command
                        line editing, completion etc. is available.
                        Use the ":vi" command |:visual| to exit "Ex" mode.
                        {not in Vi}

==============================================================================
7. The window contents                                  *window-contents*

In Normal mode and Insert/Replace mode the screen window will show the current
contents of the buffer: What You See Is What You Get.  There are two
exceptions:
- When the 'cpoptions' option contains '$', and the change is within one line,
  the text is not directly deleted, but a '$' is put at the last deleted
  character.
- When inserting text in one window, other windows on the same text are not
  updated until the insert is finished.
{Vi: The screen is not always updated on slow terminals}

Lines longer than the window width will wrap, unless the 'wrap' option is off
```

(see below). The 'linebreak' option can be set to wrap at a blank character.

If the window has room after the last line of the buffer, Vim will show '~' in the first column of the last lines in the window, like this:

```
+----------------------+
|some line             |
|last line             |
|~                     |
|~                     |
+----------------------+
```

Thus the '~' lines indicate that the end of the buffer was reached.

If the last line in a window doesn't fit, Vim will indicate this with a '@' in the first column of the last lines in the window, like this:

```
+----------------------+
|first line            |
|second line           |
|@                     |
|@                     |
+----------------------+
```

Thus the '@' lines indicate that there is a line that doesn't fit in the window.

When the "lastline" flag is present in the 'display' option, you will not see '@' characters at the left side of window. If the last line doesn't fit completely, only the part that fits is shown, and the last three characters of the last line are replaced with "@@@", like this:

```
+----------------------+
|first line            |
|second line           |
|a very long line that d|
|oesn't fit in the wi@@@|
+----------------------+
```

If there is a single line that is too long to fit in the window, this is a special situation. Vim will show only part of the line, around where the cursor is. There are no special characters shown, so that you can edit all parts of this line.
{Vi: gives an "internal error" on lines that do not fit in the window}

The '@' occasion in the 'highlight' option can be used to set special highlighting for the '@' and '~' characters. This makes it possible to distinguish them from real characters in the buffer.

The 'showbreak' option contains the string to put in front of wrapped lines.

wrap-off
If the 'wrap' option is off, long lines will not wrap. Only the part that fits on the screen is shown. If the cursor is moved to a part of the line that is not shown, the screen is scrolled horizontally. The advantage of this method is that columns are shown as they are and lines that cannot fit on the screen can be edited. The disadvantage is that you cannot see all the characters of a line at once. The 'sidescroll' option can be set to the minimal number of columns to scroll. {Vi: has no 'wrap' option}

All normal ASCII characters are displayed directly on the screen. The <Tab> is replaced with the number of spaces that it represents. Other non-printing

characters are replaced with "^{char}", where {char} is the non-printing
character with 64 added. Thus character 7 (bell) will be shown as "^G".
Characters between 127 and 160 are replaced with "~{char}", where {char} is
the character with 64 subtracted. These characters occupy more than one
position on the screen. The cursor can only be positioned on the first one.

If you set the 'number' option, all lines will be preceded with their
number. Tip: If you don't like wrapping lines to mix with the line numbers,
set the 'showbreak' option to eight spaces:
        ":set showbreak=\ \ \ \ \ \ \ "

If you set the 'list' option, <Tab> characters will not be shown as several
spaces, but as "^I". A '$' will be placed at the end of the line, so you can
find trailing blanks.

In Command-line mode only the command-line itself is shown correctly. The
display of the buffer contents is updated as soon as you go back to Command
mode.

The last line of the window is used for status and other messages. The
status messages will only be used if an option is on:

status message	option	default	Unix default	~
current mode	'showmode'	on	on	
command characters	'showcmd'	on	off	
cursor position	'ruler'	off	off	

The current mode is "-- INSERT --" or "-- REPLACE --", see |'showmode'|. The
command characters are those that you typed but were not used yet. {Vi: does
not show the characters you typed or the cursor position}

If you have a slow terminal you can switch off the status messages to speed
up editing:
        :set nosc noru nosm

If there is an error, an error message will be shown for at least one second
(in reverse video). {Vi: error messages may be overwritten with other
messages before you have a chance to read them}

Some commands show how many lines were affected. Above which threshold this
happens can be controlled with the 'report' option (default 2).

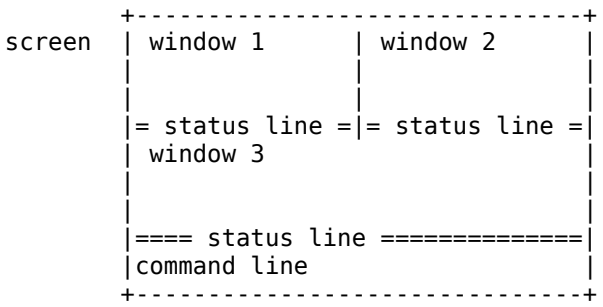
On the Amiga Vim will run in a CLI window. The name Vim and the full name of
the current file name will be shown in the title bar. When the window is
resized, Vim will automatically redraw the window. You may make the window as
small as you like, but if it gets too small not a single line will fit in it.
Make it at least 40 characters wide to be able to read most messages on the
last line.

On most Unix systems, resizing the window is recognized and handled correctly
by Vim. {Vi: not ok}

==
8. Definitions *definitions*

buffer	Contains lines of text, usually read from a file.
screen	The whole area that Vim uses to work in. This can be a terminal emulator window. Also called "the Vim window".
window	A view on a buffer. There can be multiple windows for one buffer.

A screen contains one or more windows, separated by status lines and with the
command line at the bottom.



The command line is also used for messages. It scrolls up the screen when
there is not enough room in the command line.

A difference is made between four types of lines:

buffer lines	The lines in the buffer. This is the same as the lines as they are read from/written to a file. They can be thousands of characters long.
logical lines	The buffer lines with folding applied. Buffer lines in a closed fold are changed to a single logical line: "+-- 99 lines folded". They can be thousands of characters long.
window lines	The lines displayed in a window: A range of logical lines with wrapping, line breaks, etc. applied. They can only be as long as the width of the window allows, longer lines are wrapped or truncated.
screen lines	The lines of the screen that Vim uses. Consists of the window lines of all windows, with status lines and the command line added. They can only be as long as the width of the screen allows. When the command line gets longer it wraps and lines are scrolled to make room.

buffer lines	logical lines	window lines	screen lines ~
1. one	1. one	1. +-- folded	1. +-- folded
2. two	2. +-- folded	2. five	2. five
3. three	3. five	3. six	3. six
4. four	4. six	4. seven	4. seven
5. five	5. seven		5. === status line ===
6. six			6. aaa
7. seven			7. bbb
			8. ccc ccc c
1. aaa	1. aaa	1. aaa	9. cc
2. bbb	2. bbb	2. bbb	10. ddd
3. ccc ccc ccc	3. ccc ccc ccc	3. ccc ccc c	11. ~
4. ddd	4. ddd	4. cc	12. === status line ===
		5. ddd	13. (command line)
		6. ~	

==
vim:tw=78:ts=8:ft=help:norl:
help.txt For Vim version 8.0. Last change: 2016 Sep 12

VIM - main help file

k
Move around: Use the cursor keys, or "h" to go left, h l

```
                    "j" to go down, "k" to go up, "l" to go right.       j
Close this window:  Use ":q<Enter>".
   Get out of Vim:  Use ":qa!<Enter>" (careful, all changes are lost!).

Jump to a subject:  Position the cursor on a tag (e.g. |bars|) and hit CTRL-].
   With the mouse:  ":set mouse=a" to enable the mouse (in xterm or GUI).
                    Double-click the left mouse button on a tag, e.g. |bars|.
        Jump back:  Type CTRL-T or CTRL-O.  Repeat to go further back.

Get specific help:  It is possible to go directly to whatever you want help
                    on, by giving an argument to the |:help| command.
                    Prepend something to specify the context:  *help-context*

                         WHAT                  PREPEND    EXAMPLE      ~
                      Normal mode command                  :help x
                      Visual mode command         v_       :help v_u
                      Insert mode command         i_       :help i_<Esc>
                      Command-line command        :        :help :quit
                      Command-line editing        c_       :help c_<Del>
                      Vim command argument        -        :help -r
                      Option                      '        :help 'textwidth'
                      Regular expression          /        :help /[
                    See |help-summary| for more contexts and an explanation.

  Search for help:  Type ":help word", then hit CTRL-D to see matching
                    help entries for "word".
                    Or use ":helpgrep word". |:helpgrep|

Vim stands for Vi IMproved.  Most of Vim was made by Bram Moolenaar, but only
through the help of many others.  See |credits|.
------------------------------------------------------------------------------
                                                    *doc-file-list* *Q_ct*
BASIC:
|quickref|      Overview of the most common commands you will use
|tutor|         30 minutes training course for beginners
|copying|       About copyrights
|iccf|          Helping poor children in Uganda
|sponsor|       Sponsor Vim development, become a registered Vim user
|www|           Vim on the World Wide Web
|bugs|          Where to send bug reports

USER MANUAL: These files explain how to accomplish an editing task.

|usr_toc.txt|   Table Of Contents

Getting Started ~
|usr_01.txt|  About the manuals
|usr_02.txt|  The first steps in Vim
|usr_03.txt|  Moving around
|usr_04.txt|  Making small changes
|usr_05.txt|  Set your settings
|usr_06.txt|  Using syntax highlighting
|usr_07.txt|  Editing more than one file
|usr_08.txt|  Splitting windows
|usr_09.txt|  Using the GUI
|usr_10.txt|  Making big changes
|usr_11.txt|  Recovering from a crash
|usr_12.txt|  Clever tricks

Editing Effectively ~
|usr_20.txt|  Typing command-line commands quickly
|usr_21.txt|  Go away and come back
```

```
|usr_22.txt|  Finding the file to edit
|usr_23.txt|  Editing other files
|usr_24.txt|  Inserting quickly
|usr_25.txt|  Editing formatted text
|usr_26.txt|  Repeating
|usr_27.txt|  Search commands and patterns
|usr_28.txt|  Folding
|usr_29.txt|  Moving through programs
|usr_30.txt|  Editing programs
|usr_31.txt|  Exploiting the GUI
|usr_32.txt|  The undo tree

Tuning Vim ~
|usr_40.txt|  Make new commands
|usr_41.txt|  Write a Vim script
|usr_42.txt|  Add new menus
|usr_43.txt|  Using filetypes
|usr_44.txt|  Your own syntax highlighted
|usr_45.txt|  Select your language

Making Vim Run ~
|usr_90.txt|  Installing Vim


REFERENCE MANUAL: These files explain every detail of Vim.      *reference_toc*

General subjects ~
|intro.txt|     general introduction to Vim; notation used in help files
|help.txt|      overview and quick reference (this file)
|helphelp.txt|  about using the help files
|index.txt|     alphabetical index of all commands
|help-tags|     all the tags you can jump to (index of tags)
|howto.txt|     how to do the most common editing tasks
|tips.txt|      various tips on using Vim
|message.txt|   (error) messages and explanations
|quotes.txt|    remarks from users of Vim
|todo.txt|      known problems and desired extensions
|develop.txt|   development of Vim
|debug.txt|     debugging Vim itself
|uganda.txt|    Vim distribution conditions and what to do with your money

Basic editing ~
|starting.txt|  starting Vim, Vim command arguments, initialisation
|editing.txt|   editing and writing files
|motion.txt|    commands for moving around
|scroll.txt|    scrolling the text in the window
|insert.txt|    Insert and Replace mode
|change.txt|    deleting and replacing text
|indent.txt|    automatic indenting for C and other languages
|undo.txt|      Undo and Redo
|repeat.txt|    repeating commands, Vim scripts and debugging
|visual.txt|    using the Visual mode (selecting a text area)
|various.txt|   various remaining commands
|recover.txt|   recovering from a crash

Advanced editing ~
|cmdline.txt|   Command-line editing
|options.txt|   description of all options
|pattern.txt|   regexp patterns and search commands
|map.txt|       key mapping and abbreviations
|tagsrch.txt|   tags and special searches
|quickfix.txt|  commands for a quick edit-compile-fix cycle
```

```
|windows.txt|   commands for using multiple windows and buffers
|tabpage.txt|   commands for using multiple tab pages
|syntax.txt|    syntax highlighting
|spell.txt|     spell checking
|diff.txt|      working with two to four versions of the same file
|autocmd.txt|   automatically executing commands on an event
|filetype.txt|  settings done specifically for a type of file
|eval.txt|      expression evaluation, conditional commands
|channel.txt|   Jobs, Channels, inter-process communication
|fold.txt|      hide (fold) ranges of lines

Special issues ~
|print.txt|     printing
|remote.txt|    using Vim as a server or client
|term.txt|      using different terminals and mice
|digraph.txt|   list of available digraphs
|mbyte.txt|     multi-byte text support
|mlang.txt|     non-English language support
|arabic.txt|    Arabic language support and editing
|farsi.txt|     Farsi (Persian) editing
|hebrew.txt|    Hebrew language support and editing
|russian.txt|   Russian language support and editing
|ft_ada.txt|    Ada (the programming language) support
|ft_sql.txt|    about the SQL filetype plugin
|hangulin.txt|  Hangul (Korean) input mode
|rileft.txt|    right-to-left editing mode

GUI ~
|gui.txt|       Graphical User Interface (GUI)
|gui_w32.txt|   Win32 GUI
|gui_x11.txt|   X11 GUI

Interfaces ~
|if_cscop.txt|  using Cscope with Vim
|if_lua.txt|    Lua interface
|if_mzsch.txt|  MzScheme interface
|if_perl.txt|   Perl interface
|if_pyth.txt|   Python interface
|if_tcl.txt|    Tcl interface
|if_ole.txt|    OLE automation interface for Win32
|if_ruby.txt|   Ruby interface
|debugger.txt|  Interface with a debugger
|workshop.txt|  Sun Visual Workshop interface
|netbeans.txt|  NetBeans External Editor interface
|sign.txt|      debugging signs

Versions ~
|vi_diff.txt|   Main differences between Vim and Vi
|version4.txt|  Differences between Vim version 3.0 and 4.x
|version5.txt|  Differences between Vim version 4.6 and 5.x
|version6.txt|  Differences between Vim version 5.7 and 6.x
|version7.txt|  Differences between Vim version 6.4 and 7.x
|version8.txt|  Differences between Vim version 7.4 and 8.x
                                                *sys-file-list*
Remarks about specific systems ~
|os_390.txt|    OS/390 Unix
|os_amiga.txt|  Amiga
|os_beos.txt|   BeOS and BeBox
|os_dos.txt|    MS-DOS and MS-Windows NT/95 common items
|os_mac.txt|    Macintosh
|os_mint.txt|   Atari MiNT
|os_msdos.txt|  MS-DOS (plain DOS and DOS box under Windows)
```

```
|os_os2.txt|    OS/2
|os_qnx.txt|    QNX
|os_risc.txt|   RISC-OS
|os_unix.txt|   Unix
|os_vms.txt|    VMS
|os_win32.txt|  MS-Windows 95/98/NT
                                                *standard-plugin-list*
Standard plugins ~
|pi_getscript.txt| Downloading latest version of Vim scripts
|pi_gzip.txt|      Reading and writing compressed files
|pi_logipat.txt|   Logical operators on patterns
|pi_netrw.txt|     Reading and writing files over a network
|pi_paren.txt|     Highlight matching parens
|pi_tar.txt|       Tar file explorer
|pi_vimball.txt|   Create a self-installing Vim script
|pi_zip.txt|       Zip archive explorer

LOCAL ADDITIONS:                                *local-additions*

------------------------------------------------------------------------------
*bars*          Bars example

Now that you've jumped here with CTRL-] or a double mouse click, you can use
CTRL-T, CTRL-O, g<RightMouse>, or <C-RightMouse> to go back to where you were.

Note that tags are within | characters, but when highlighting is enabled these
characters are hidden.  That makes it easier to read a command.

Anyway, you can use CTRL-] on any word, also when it is not within |, and Vim
will try to find help for it.  Especially for options in single quotes, e.g.
'compatible'.

------------------------------------------------------------------------------
 vim:tw=78:fo=tcq2:isk=!-~,^*,^\|,^\":ts=8:ft=help:norl:
*helphelp.txt*  For Vim version 8.0.  Last change: 2017 Mar 19


                  VIM REFERENCE MANUAL    by Bram Moolenaar


Help on help files                                      *helphelp*

1. Help commands                |online-help|
2. Translated help files        |help-translated|
3. Writing help files           |help-writing|

==============================================================================
1. Help commands                                        *online-help*

                        *help* *<Help>* *:h* *:help* *<F1>* *i_<F1>* *i_<Help>*
<Help>          or
:h[elp]                 Open a window and display the help file in read-only
                        mode.  If there is a help window open already, use
                        that one.  Otherwise, if the current window uses the
                        full width of the screen or is at least 80 characters
                        wide, the help window will appear just above the
                        current window.  Otherwise the new window is put at
                        the very top.
                        The 'helplang' option is used to select a language, if
                        the main help file is available in several languages.
                        {not in Vi}
```

```
                                        *{subject}* *E149* *E661*
:h[elp] {subject}       Like ":help", additionally jump to the tag {subject}.
                        For example:  >
                                :help options

<                       {subject} can include wildcards such as "*", "?" and
                        "[a-z]":
                           :help z?     jump to help for any "z" command
                           :help z.     jump to the help for "z."
                        But when a tag exists it is taken literally:
                           :help :?     jump to help for ":?"

                        If there is no full match for the pattern, or there
                        are several matches, the "best" match will be used.
                        A sophisticated algorithm is used to decide which
                        match is better than another one.  These items are
                        considered in the computation:
                        - A match with same case is much better than a match
                          with different case.
                        - A match that starts after a non-alphanumeric
                          character is better than a match in the middle of a
                          word.
                        - A match at or near the beginning of the tag is
                          better than a match further on.
                        - The more alphanumeric characters match, the better.
                        - The shorter the length of the match, the better.

                        The 'helplang' option is used to select a language, if
                        the {subject} is available in several languages.
                        To find a tag in a specific language, append "@ab",
                        where "ab" is the two-letter language code.  See
                        |help-translated|.

                        Note that the longer the {subject} you give, the less
                        matches will be found.  You can get an idea how this
                        all works by using commandline completion (type CTRL-D
                        after ":help subject" |c_CTRL-D|).
                        If there are several matches, you can have them listed
                        by hitting CTRL-D.  Example: >
                                :help cont<Ctrl-D>

<                       Instead of typing ":help CTRL-V" to search for help
                        for CTRL-V you can type: >
                                :help ^V
<                       This also works together with other characters, for
                        example to find help for CTRL-V in Insert mode: >
                                :help i^V
<
                        It is also possible to first do ":help" and then
                        use ":tag {pattern}" in the help window.  The
                        ":tnext" command can then be used to jump to other
                        matches, "tselect" to list matches and choose one. >
                                :help index
                                :tselect /.*mode

<                       When there is no argument you will see matches for
                        "help", to avoid listing all possible matches (that
                        would be very slow).
                        The number of matches displayed is limited to 300.

                        The `:help` command can be followed by '|' and another
                        command, but you don't need to escape the '|' inside a
```

help command. So these both work: >
:help |
:help k| only
< Note that a space before the '|' is seen as part of the ":help" argument.
You can also use <LF> or <CR> to separate the help command from a following command. You need to type CTRL-V first to insert the <LF> or <CR>. Example: >
:help so<C-V><CR>only
< {not in Vi}

:h[elp]! [subject] Like ":help", but in non-English help files prefer to find a tag in a file with the same language as the current file. See |help-translated|.

:helpc *:helpclose*
:helpc[lose] Close one help window, if there is one.

:helpg *:helpgrep*
:helpg[rep] {pattern}[@xx]
Search all help text files and make a list of lines in which {pattern} matches. Jumps to the first match.
The optional [@xx] specifies that only matches in the "xx" language are to be found.
You can navigate through the matches with the |quickfix| commands, e.g., |:cnext| to jump to the next one. Or use |:cwindow| to get the list of matches in the quickfix window.
{pattern} is used as a Vim regexp |pattern|.
'ignorecase' is not used, add "\c" to ignore case.
Example for case sensitive search: >
:helpgrep Uganda
< Example for case ignoring search: >
:helpgrep uganda\c
< Example for searching in French help: >
:helpgrep backspace@fr
< The pattern does not support line breaks, it must match within one line. You can use |:grep| instead, but then you need to get the list of help files in a complicated way.
Cannot be followed by another command, everything is used as part of the pattern. But you can use |:execute| when needed.
Compressed help files will not be searched (Fedora compresses the help files).
{not in Vi}

:lh *:lhelpgrep*
:lh[elpgrep] {pattern}[@xx]
Same as ":helpgrep", except the location list is used instead of the quickfix list. If the help window is already opened, then the location list for that window is used. Otherwise, a new help window is opened and the location list for that window is set. The location list for the current window is not changed then.

:exu *:exusage*
:exu[sage] Show help on Ex commands. Added to simulate the Nvi command. {not in Vi}

:viu *:viusage*

```
:viu[sage]              Show help on Normal mode commands.  Added to simulate
                        the Nvi command. {not in Vi}

When no argument is given to |:help| the file given with the 'helpfile' option
will be opened.  Otherwise the specified tag is searched for in all "doc/tags"
files in the directories specified in the 'runtimepath' option.

The initial height of the help window can be set with the 'helpheight' option
(default 20).

Jump to specific subjects by using tags.  This can be done in two ways:
- Use the "CTRL-]" command while standing on the name of a command or option.
  This only works when the tag is a keyword.  "<C-Leftmouse>" and
  "g<LeftMouse>" work just like "CTRL-]".
- use the ":ta {subject}" command.  This also works with non-keyword
  characters.

Use CTRL-T or CTRL-O to jump back.
Use ":q" to close the help window.

If there are several matches for an item you are looking for, this is how you
can jump to each one of them:
1. Open a help window
2. Use the ":tag" command with a slash prepended to the tag.  E.g.: >
        :tag /min
3. Use ":tnext" to jump to the next matching tag.

It is possible to add help files for plugins and other items.  You don't need
to change the distributed help files for that.  See |add-local-help|.

To write a local help file, see |write-local-help|.

Note that the title lines from the local help files are automagically added to
the "LOCAL ADDITIONS" section in the "help.txt" help file |local-additions|.
This is done when viewing the file in Vim, the file itself is not changed.  It
is done by going through all help files and obtaining the first line of each
file.  The files in $VIMRUNTIME/doc are skipped.

                                                        *help-xterm-window*
If you want to have the help in another xterm window, you could use this
command: >
        :!xterm -e vim +help &
<

                        *:helpfind* *:helpf*
:helpf[ind]             Like |:help|, but use a dialog to enter the argument.
                        Only for backwards compatibility.  It now executes the
                        ToolBar.FindHelp menu entry instead of using a builtin
                        dialog.  {only when compiled with |+GUI_GTK|}
                        {not in Vi}

                                        *:helpt* *:helptags*
                                *E154* *E150* *E151* *E152* *E153* *E670*
:helpt[ags] [++t] {dir}
                        Generate the help tags file(s) for directory {dir}.
                        When {dir} is ALL then all "doc" directories in
                        'runtimepath' will be used.

                        All "*.txt" and "*.??x" files in the directory and
                        sub-directories are scanned for a help tag definition
                        in between stars.  The "*.??x" files are for
                        translated docs, they generate the "tags-??" file, see
```

```
                        |help-translated|.  The generated tags files are
                        sorted.
                        When there are duplicates an error message is given.
                        An existing tags file is silently overwritten.

                        The optional "++t" argument forces adding the
                        "help-tags" tag.  This is also done when the {dir} is
                        equal to $VIMRUNTIME/doc.

                        To rebuild the help tags in the runtime directory
                        (requires write permission there): >
                                :helptags $VIMRUNTIME/doc
<                       {not in Vi}


==============================================================================
2. Translated help files                                *help-translated*

It is possible to add translated help files, next to the original English help
files.  Vim will search for all help in "doc" directories in 'runtimepath'.
This is only available when compiled with the |+multi_lang| feature.

At this moment translations are available for:
        Chinese  - multiple authors
        French   - translated by David Blanchet
        Italian  - translated by Antonio Colombo
        Japanese - multiple authors
        Polish   - translated by Mikolaj Machowski
        Russian  - translated by Vassily Ragosin
See the Vim website to find them: http://www.vim.org/translations.php

A set of translated help files consists of these files:

        help.abx
        howto.abx
        ...
        tags-ab

"ab" is the two-letter language code.  Thus for Italian the names are:

        help.itx
        howto.itx
        ...
        tags-it

The 'helplang' option can be set to the preferred language(s).  The default is
set according to the environment.  Vim will first try to find a matching tag
in the preferred language(s).  English is used when it cannot be found.

To find a tag in a specific language, append "@ab" to a tag, where "ab" is the
two-letter language code.  Example: >
        :he user-manual@it
        :he user-manual@en
The first one finds the Italian user manual, even when 'helplang' is empty.
The second one finds the English user manual, even when 'helplang' is set to
"it".

When using command-line completion for the ":help" command, the "@en"
extension is only shown when a tag exists for multiple languages.  When the
tag only exists for English "@en" is omitted.  When the first candidate has an
"@ab" extension and it matches the first language in 'helplang' "@ab" is also
omitted.
```

When using |CTRL-]| or ":help!" in a non-English help file Vim will try to
find the tag in the same language. If not found then 'helplang' will be used
to select a language.

Help files must use latin1 or utf-8 encoding. Vim assumes the encoding is
utf-8 when finding non-ASCII characters in the first line. Thus you must
translate the header with "For Vim version".

The same encoding must be used for the help files of one language in one
directory. You can use a different encoding for different languages and use
a different encoding for help files of the same language but in a different
directory.

Hints for translators:
- Do not translate the tags. This makes it possible to use 'helplang' to
  specify the preferred language. You may add new tags in your language.
- When you do not translate a part of a file, add tags to the English version,
  using the "tag@en" notation.
- Make a package with all the files and the tags file available for download.
  Users can drop it in one of the "doc" directories and start use it.
  Report this to Bram, so that he can add a link on www.vim.org.
- Use the |:helptags| command to generate the tags files. It will find all
  languages in the specified directory.

==
3. Writing help files   *help-writing*

For ease of use, a Vim help file for a plugin should follow the format of the
standard Vim help files. If you are writing a new help file it's best to copy
one of the existing files and use it as a template.

The first line in a help file should have the following format:

helpfile_name.txt   For Vim version 7.3   Last change: 2010 June 4

The first field is a link to the help file name. The second field describes
the applicable Vim version. The last field specifies the last modification
date of the file. Each field is separated by a tab.

At the bottom of the help file, place a Vim modeline to set the 'textwidth'
and 'tabstop' options and the 'filetype' to "help". Never set a global option
in such a modeline, that can have consequences undesired by whoever reads that
help.

TAGS

To define a help tag, place the name between asterisks (*tag-name*). The
tag-name should be different from all the Vim help tag names and ideally
should begin with the name of the Vim plugin. The tag name is usually right
aligned on a line.

When referring to an existing help tag and to create a hot-link, place the
name between two bars (|) eg. |help-writing|.

When referring to a Vim command and to create a hot-link, place the
name between two backticks, eg. inside `:filetype`. You will see this is
highlighted as a command, like a code block (see below).

When referring to a Vim option in the help file, place the option name between
two single quotes, eg. 'statusline'

```
HIGHLIGHTING

To define a column heading, use a tilde character at the end of the line.
This will highlight the column heading in a different color.  E.g.

Column heading~

To separate sections in a help file, place a series of '=' characters in a
line starting from the first column.  The section separator line is highlighted
differently.

To quote a block of ex-commands verbatim, place a greater than (>) character
at the end of the line before the block and a less than (<) character as the
first non-blank on a line following the block.  Any line starting in column 1
also implicitly stops the block of ex-commands before it.  E.g. >
    function Example_Func()
        echo "Example"
    endfunction
<

The following are highlighted differently in a Vim help file:
  - a special key name expressed either in <> notation as in <PageDown>, or
    as a Ctrl character as in CTRL-X
  - anything between {braces}, e.g. {lhs} and {rhs}

The word "Note", "Notes" and similar automagically receive distinctive
highlighting.  So do these:
        *Todo   something to do
        *Error  something wrong

You can find the details in $VIMRUNTIME/syntax/help.vim

 vim:tw=78:ts=8:ft=help:norl:
*index.txt*     For Vim version 8.0.  Last change: 2017 Aug 02


                  VIM REFERENCE MANUAL    by Bram Moolenaar

                                                                  *index*
This file contains a list of all commands for each mode, with a tag and a
short description.  The lists are sorted on ASCII value.

Tip: When looking for certain functionality, use a search command.  E.g.,
to look for deleting something, use: "/delete".

1. Insert mode                          |insert-index|
2. Normal mode                          |normal-index|
   2.1. Text objects                    |objects|
   2.2. Window commands                 |CTRL-W|
   2.3. Square bracket commands         |[|
   2.4. Commands starting with 'g'      |g|
   2.5. Commands starting with 'z'      |z|
3. Visual mode                          |visual-index|
4. Command-line editing                 |ex-edit-index|
5. EX commands                          |ex-cmd-index|

For an overview of options see help.txt |option-list|.
For an overview of built-in functions see |functions|.
For a list of Vim variables see |vim-variable|.
For a complete listing of all help items see |help-tags|.
```

```
==============================================================================
1. Insert mode                                              *insert-index*

tag             char            action in Insert mode   ~
------------------------------------------------------------------------------
|i_CTRL-@|      CTRL-@           insert previously inserted text and stop
                                insert
|i_CTRL-A|      CTRL-A           insert previously inserted text
                CTRL-B           not used |i_CTRL-B-gone|
|i_CTRL-C|      CTRL-C           quit insert mode, without checking for
                                abbreviation, unless 'insertmode' set.
|i_CTRL-D|      CTRL-D           delete one shiftwidth of indent in the current
                                line
|i_CTRL-E|      CTRL-E           insert the character which is below the cursor
                CTRL-F           not used (but by default it's in 'cinkeys' to
                                re-indent the current line)
|i_CTRL-G_j|    CTRL-G CTRL-J    line down, to column where inserting started
|i_CTRL-G_j|    CTRL-G j         line down, to column where inserting started
|i_CTRL-G_j|    CTRL-G <Down>    line down, to column where inserting started
|i_CTRL-G_k|    CTRL-G CTRL-K    line up, to column where inserting started
|i_CTRL-G_k|    CTRL-G k         line up, to column where inserting started
|i_CTRL-G_k|    CTRL-G <Up>      line up, to column where inserting started
|i_CTRL-G_u|    CTRL-G u         start new undoable edit
|i_CTRL-G_U|    CTRL-G U         don't break undo with next cursor movement
|i_<BS>|        <BS>             delete character before the cursor
|i_digraph|     {char1}<BS>{char2}
                                enter digraph (only when 'digraph' option set)
|i_CTRL-H|      CTRL-H           same as <BS>
|i_<Tab>|       <Tab>            insert a <Tab> character
|i_CTRL-I|      CTRL-I           same as <Tab>
|i_<NL>|        <NL>             same as <CR>
|i_CTRL-J|      CTRL-J           same as <CR>
|i_CTRL-K|      CTRL-K {char1} {char2}
                                enter digraph
|i_CTRL-L|      CTRL-L           when 'insertmode' set: Leave Insert mode
|i_<CR>|        <CR>             begin new line
|i_CTRL-M|      CTRL-M           same as <CR>
|i_CTRL-N|      CTRL-N           find next match for keyword in front of the
                                cursor
|i_CTRL-O|      CTRL-O           execute a single command and return to insert
                                mode
|i_CTRL-P|      CTRL-P           find previous match for keyword in front of
                                the cursor
|i_CTRL-Q|      CTRL-Q           same as CTRL-V, unless used for terminal
                                control flow
|i_CTRL-R|      CTRL-R {0-9a-z"%#*:=}
                                insert the contents of a register
|i_CTRL-R_CTRL-R| CTRL-R CTRL-R {0-9a-z"%#*:=}
                                insert the contents of a register literally
|i_CTRL-R_CTRL-O| CTRL-R CTRL-O {0-9a-z"%#*:=}
                                insert the contents of a register literally
                                and don't auto-indent
|i_CTRL-R_CTRL-P| CTRL-R CTRL-P {0-9a-z"%#*:=}
                                insert the contents of a register literally
                                and fix indent.
                CTRL-S           (used for terminal control flow)
|i_CTRL-T|      CTRL-T           insert one shiftwidth of indent in current
                                line
|i_CTRL-U|      CTRL-U           delete all entered characters in the current
                                line
|i_CTRL-V|      CTRL-V {char}    insert next non-digit literally
```

```
|i_CTRL-V_digit| CTRL-V {number} insert three digit decimal number as a single
                                byte.
|i_CTRL-W|      CTRL-W          delete word before the cursor
|i_CTRL-X|      CTRL-X {mode}   enter CTRL-X sub mode, see |i_CTRL-X_index|
|i_CTRL-Y|      CTRL-Y          insert the character which is above the cursor
|i_CTRL-Z|      CTRL-Z          when 'insertmode' set: suspend Vim
|i_<Esc>|       <Esc>           end insert mode (unless 'insertmode' set)
|i_CTRL-[|      CTRL-[          same as <Esc>
|i_CTRL-\_CTRL-N| CTRL-\ CTRL-N go to Normal mode
|i_CTRL-\_CTRL-G| CTRL-\ CTRL-G go to mode specified with 'insertmode'
                CTRL-\ a - z    reserved for extensions
                CTRL-\ others   not used
|i_CTRL-]|      CTRL-]          trigger abbreviation
|i_CTRL-^|      CTRL-^          toggle use of |:lmap| mappings
|i_CTRL-_|      CTRL-_          When 'allowrevins' set: change language
                                (Hebrew, Farsi) {only when compiled with
                                the |+rightleft| feature}

                <Space> to '~'  not used, except '0' and '^' followed by
                                CTRL-D

|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
                                it in the next line

|i_<Del>|       <Del>           delete character under the cursor

                Meta characters (0x80 to 0xff, 128 to 255)
                                not used

|i_<Left>|      <Left>          cursor one character left
|i_<S-Left>|    <S-Left>        cursor one word left
|i_<C-Left>|    <C-Left>        cursor one word left
|i_<Right>|     <Right>         cursor one character right
|i_<S-Right>|   <S-Right>       cursor one word right
|i_<C-Right>|   <C-Right>       cursor one word right
|i_<Up>|        <Up>            cursor one line up
|i_<S-Up>|      <S-Up>          same as <PageUp>
|i_<Down>|      <Down>          cursor one line down
|i_<S-Down>|    <S-Down>        same as <PageDown>
|i_<Home>|      <Home>          cursor to start of line
|i_<C-Home>|    <C-Home>        cursor to start of file
|i_<End>|       <End>           cursor past end of line
|i_<C-End>|     <C-End>         cursor past end of file
|i_<PageUp>|    <PageUp>        one screenful backward
|i_<PageDown>|  <PageDown>      one screenful forward
|i_<F1>|        <F1>            same as <Help>
|i_<Help>|      <Help>          stop insert mode and display help window
|i_<Insert>|    <Insert>        toggle Insert/Replace mode
|i_<LeftMouse>| <LeftMouse>     cursor at mouse click
|i_<ScrollWheelDown>|   <ScrollWheelDown>       move window three lines down
|i_<S-ScrollWheelDown>| <S-ScrollWheelDown>     move window one page down
|i_<ScrollWheelUp>|     <ScrollWheelUp>         move window three lines up
|i_<S-ScrollWheelUp>|   <S-ScrollWheelUp>       move window one page up
|i_<ScrollWheelLeft>|   <ScrollWheelLeft>       move window six columns left
|i_<S-ScrollWheelLeft>| <S-ScrollWheelLeft>     move window one page left
|i_<ScrollWheelRight>|  <ScrollWheelRight>      move window six columns right
|i_<S-ScrollWheelRight>| <S-ScrollWheelRight>   move window one page right

commands in CTRL-X submode                              *i_CTRL-X_index*

|i_CTRL-X_CTRL-D|       CTRL-X CTRL-D   complete defined identifiers
```

```
|i_CTRL-X_CTRL-E|       CTRL-X CTRL-E   scroll up
|i_CTRL-X_CTRL-F|       CTRL-X CTRL-F   complete file names
|i_CTRL-X_CTRL-I|       CTRL-X CTRL-I   complete identifiers
|i_CTRL-X_CTRL-K|       CTRL-X CTRL-K   complete identifiers from dictionary
|i_CTRL-X_CTRL-L|       CTRL-X CTRL-L   complete whole lines
|i_CTRL-X_CTRL-N|       CTRL-X CTRL-N   next completion
|i_CTRL-X_CTRL-O|       CTRL-X CTRL-O   omni completion
|i_CTRL-X_CTRL-P|       CTRL-X CTRL-P   previous completion
|i_CTRL-X_CTRL-S|       CTRL-X CTRL-S   spelling suggestions
|i_CTRL-X_CTRL-T|       CTRL-X CTRL-T   complete identifiers from thesaurus
|i_CTRL-X_CTRL-Y|       CTRL-X CTRL-Y   scroll down
|i_CTRL-X_CTRL-U|       CTRL-X CTRL-U   complete with 'completefunc'
|i_CTRL-X_CTRL-V|       CTRL-X CTRL-V   complete like in : command line
|i_CTRL-X_CTRL-]|       CTRL-X CTRL-]   complete tags
|i_CTRL-X_s|            CTRL-X s        spelling suggestions
{not available when compiled without the |+insert_expand| feature}

==============================================================================
2. Normal mode                                          *normal-index*

CHAR     any non-blank character
WORD     a sequence of non-blank characters
N        a number entered before the command
{motion} a cursor movement command
Nmove    the text that is moved over with a {motion}
SECTION  a section that possibly starts with '}' instead of '{'

note: 1 = cursor movement command; 2 = can be undone/redone

tag             char          note action in Normal mode        ~
------------------------------------------------------------------------------
                CTRL-@           not used
|CTRL-A|        CTRL-A         2  add N to number at/after cursor
|CTRL-B|        CTRL-B         1  scroll N screens Backwards
|CTRL-C|        CTRL-C            interrupt current (search) command
|CTRL-D|        CTRL-D            scroll Down N lines (default: half a screen)
|CTRL-E|        CTRL-E            scroll N lines upwards (N lines Extra)
|CTRL-F|        CTRL-F         1  scroll N screens Forward
|CTRL-G|        CTRL-G            display current file name and position
|<BS>|          <BS>           1  same as "h"
|CTRL-H|        CTRL-H           1  same as "h"
|<Tab>|         <Tab>           1  go to N newer entry in jump list
|CTRL-I|        CTRL-I           1  same as <Tab>
|<NL>|          <NL>            1  same as "j"
|CTRL-J|        CTRL-J           1  same as "j"
                CTRL-K            not used
|CTRL-L|        CTRL-L            redraw screen
|<CR>|          <CR>            1  cursor to the first CHAR N lines lower
|CTRL-M|        CTRL-M           1  same as <CR>
|CTRL-N|        CTRL-N           1  same as "j"
|CTRL-O|        CTRL-O           1  go to N older entry in jump list
|CTRL-P|        CTRL-P           1  same as "k"
                CTRL-Q            (used for terminal control flow)
|CTRL-R|        CTRL-R         2  redo changes which were undone with 'u'
                CTRL-S            (used for terminal control flow)
|CTRL-T|        CTRL-T            jump to N older Tag in tag list
|CTRL-U|        CTRL-U            scroll N lines Upwards (default: half a
                                  screen)
|CTRL-V|        CTRL-V            start blockwise Visual mode
|CTRL-W|        CTRL-W {char}     window commands, see |CTRL-W|
|CTRL-X|        CTRL-X         2  subtract N from number at/after cursor
|CTRL-Y|        CTRL-Y            scroll N lines downwards
```

```
|CTRL-Z|        CTRL-Z             suspend program (or start new shell)
                CTRL-[ <Esc>        not used
|CTRL-\_CTRL-N| CTRL-\ CTRL-N      go to Normal mode (no-op)
|CTRL-\_CTRL-G| CTRL-\ CTRL-G      go to mode specified with 'insertmode'
                CTRL-\ a - z        reserved for extensions
                CTRL-\ others       not used
|CTRL-]|        CTRL-]             :ta to ident under cursor
|CTRL-^|        CTRL-^             edit Nth alternate file (equivalent to
                                    ":e #N")
                CTRL-_              not used

|<Space>|       <Space>          1  same as "l"
|!|            !{motion}{filter}
                                 2  filter Nmove text through the {filter}
                                    command
|!!|           !!{filter}       2  filter N lines through the {filter} command
|quote|        "{a-zA-Z0-9.%#:-"}  use register {a-zA-Z0-9.%#:-"} for next
                                    delete, yank or put (uppercase to append)
                                    ({.%#:} only work with put)
|#|            #                1  search backward for the Nth occurrence of
                                    the ident under the cursor
|$|            $                1  cursor to the end of Nth next line
|%|            %                1  find the next (curly/square) bracket on
                                    this line and go to its match, or go to
                                    matching comment bracket, or go to matching
                                    preprocessor directive.
|N%|           {count}%          1  go to N percentage in the file
|&|            &                2  repeat last :s
|'|            '{a-zA-Z0-9}     1  cursor to the first CHAR on the line with
                                    mark {a-zA-Z0-9}
|''|           ''               1  cursor to the first CHAR of the line where
                                    the cursor was before the latest jump.
|'(|           '(               1  cursor to the first CHAR on the line of the
                                    start of the current sentence
|')|           ')               1  cursor to the first CHAR on the line of the
                                    end of the current sentence
|'<|           '<               1  cursor to the first CHAR of the line where
                                    highlighted area starts/started in the
                                    current buffer.
|'>|           '>               1  cursor to the first CHAR of the line where
                                    highlighted area ends/ended in the current
                                    buffer.
|'[|           '[               1  cursor to the first CHAR on the line of the
                                    start of last operated text or start of put
                                    text
|']|           ']               1  cursor to the first CHAR on the line of the
                                    end of last operated text or end of put
                                    text
|'{|           '{               1  cursor to the first CHAR on the line of the
                                    start of the current paragraph
|'}|           '}               1  cursor to the first CHAR on the line of the
                                    end of the current paragraph
|(|            (                1  cursor N sentences backward
|)|            )                1  cursor N sentences forward
|star|         *                1  search forward for the Nth occurrence of
                                    the ident under the cursor
|+|            +                1  same as <CR>
|,|            ,                1  repeat latest f, t, F or T in opposite
                                    direction N times
|-|            -                1  cursor to the first CHAR N lines higher
|.|            .                2  repeat last change with count replaced with
                                    N
```

```
|/|             /{pattern}<CR>  1  search forward for the Nth occurrence of
                                    {pattern}
|/<CR>|         /<CR>           1  search forward for {pattern} of last search
|count|         0                1  cursor to the first char of the line
|count|         1                   prepend to command to give a count
|count|         2                        "
|count|         3                        "
|count|         4                        "
|count|         5                        "
|count|         6                        "
|count|         7                        "
|count|         8                        "
|count|         9                        "
|:|             :                1  start entering an Ex command
|N:|            {count}:             start entering an Ex command with range
                                    from current line to N-1 lines down
|;|             ;                1  repeat latest f, t, F or T N times
|<|             <{motion}       2  shift Nmove lines one 'shiftwidth'
                                    leftwards
|<<|            <<               2  shift N lines one 'shiftwidth' leftwards
|=|             ={motion}       2  filter Nmove lines through "indent"
|==|            ==               2  filter N lines through "indent"
|>|             >{motion}       2  shift Nmove lines one 'shiftwidth'
                                    rightwards
|>>|            >>               2  shift N lines one 'shiftwidth' rightwards
|?|             ?{pattern}<CR>  1  search backward for the Nth previous
                                    occurrence of {pattern}
|?<CR>|         ?<CR>           1  search backward for {pattern} of last search
|@|             @{a-z}          2  execute the contents of register {a-z}
                                    N times
|@:|            @:                  repeat the previous ":" command N times
|@@|            @@               2  repeat the previous @{a-z} N times
|A|             A                2  append text after the end of the line N times
|B|             B                1  cursor N WORDS backward
|C|             ["x]C           2  change from the cursor position to the end
                                    of the line, and N-1 more lines [into
                                    register x]; synonym for "c$"
|D|             ["x]D           2  delete the characters under the cursor
                                    until the end of the line and N-1 more
                                    lines [into register x]; synonym for "d$"
|E|             E                1  cursor forward to the end of WORD N
|F|             F{char}         1  cursor to the Nth occurrence of {char} to
                                    the left
|G|             G                1  cursor to line N, default last line
|H|             H                1  cursor to line N from top of screen
|I|             I                2  insert text before the first CHAR on the
                                    line N times
|J|             J                2  Join N lines; default is 2
|K|             K                   lookup Keyword under the cursor with
                                    'keywordprg'
|L|             L                1  cursor to line N from bottom of screen
|M|             M                1  cursor to middle line of screen
|N|             N                1  repeat the latest '/' or '?' N times in
                                    opposite direction
|O|             O                2  begin a new line above the cursor and
                                    insert text, repeat N times
|P|             ["x]P           2  put the text [from register x] before the
                                    cursor N times
|Q|             Q                   switch to "Ex" mode
|R|             R                2  enter replace mode: overtype existing
                                    characters, repeat the entered text N-1
                                    times
```

```
|S|             ["x]S          2  delete N lines [into register x] and start
                                   insert; synonym for "cc".
|T|             T{char}        1  cursor till after Nth occurrence of {char}
                                   to the left
|U|             U              2  undo all latest changes on one line
|V|             V                 start linewise Visual mode
|W|             W              1  cursor N WORDS forward
|X|             ["x]X          2  delete N characters before the cursor [into
                                   register x]
|Y|             ["x]Y             yank N lines [into register x]; synonym for
                                   "yy"
|ZZ|            ZZ                store current file if modified, and exit
|ZQ|            ZQ                exit current file always
|[|             [{char}           square bracket command (see |[| below)
                \                  not used
|]|             ]{char}           square bracket command (see |]| below)
|^|             ^              1  cursor to the first CHAR of the line
|_|             _              1  cursor to the first CHAR N - 1 lines lower
|`|             `{a-zA-Z0-9}    1  cursor to the mark {a-zA-Z0-9}
|``(|           ``(             1  cursor to the start of the current sentence
|``)|           ``)             1  cursor to the end of the current sentence
|``<|           ``<             1  cursor to the start of the highlighted area
|``>|           ``>             1  cursor to the end of the highlighted area
|``[|           ``[             1  cursor to the start of last operated text
                                   or start of putted text
|``]|           ``]             1  cursor to the end of last operated text or
                                   end of putted text
|````|          ````            1  cursor to the position before latest jump
|``{|           ``{             1  cursor to the start of the current paragraph
|``}|           ``}             1  cursor to the end of the current paragraph
|a|             a              2  append text after the cursor N times
|b|             b              1  cursor N words backward
|c|             ["x]c{motion}  2  delete Nmove text [into register x] and
                                   start insert
|cc|            ["x]cc         2  delete N lines [into register x] and start
                                   insert
|d|             ["x]d{motion}  2  delete Nmove text [into register x]
|dd|            ["x]dd         2  delete N lines [into register x]
|do|            do             2  same as ":diffget"
|dp|            dp             2  same as ":diffput"
|e|             e              1  cursor forward to the end of word N
|f|             f{char}        1  cursor to Nth occurrence of {char} to the
                                   right
|g|             g{char}           extended commands, see |g| below
|h|             h              1  cursor N chars to the left
|i|             i              2  insert text before the cursor N times
|j|             j              1  cursor N lines downward
|k|             k              1  cursor N lines upward
|l|             l              1  cursor N chars to the right
|m|             m{A-Za-z}         set mark {A-Za-z} at cursor position
|n|             n              1  repeat the latest '/' or '?' N times
|o|             o              2  begin a new line below the cursor and
                                   insert text, repeat N times
|p|             ["x]p          2  put the text [from register x] after the
                                   cursor N times
|q|             q{0-9a-zA-Z"}     record typed characters into named register
                                   {0-9a-zA-Z"} (uppercase to append)
|q|             q                (while recording) stops recording
|q:|            q:                edit : command-line in command-line window
|q/|            q/                edit / command-line in command-line window
|q?|            q?                edit ? command-line in command-line window
|r|             r{char}        2  replace N chars with {char}
```

```
|s|                 ["x]s           2  (substitute) delete N characters [into
                                       register x] and start insert
|t|                 t{char}         1  cursor till before Nth occurrence of {char}
                                       to the right
|u|                 u               2  undo changes
|v|                 v                  start characterwise Visual mode
|w|                 w               1  cursor N words forward
|x|                 ["x]x           2  delete N characters under and after the
                                       cursor [into register x]
|y|                 ["x]y{motion}      yank Nmove text [into register x]
|yy|                ["x]yy             yank N lines [into register x]
|z|                 z{char}            commands starting with 'z', see |z| below
|{|                 {               1  cursor N paragraphs backward
|bar|               |               1  cursor to column N
|}|                 }               1  cursor N paragraphs forward
|~|                 ~               2  'tildeop' off: switch case of N characters
                                       under cursor and move the cursor N
                                       characters to the right
|~|                 ~{motion}          'tildeop' on: switch case of Nmove text
|<C-End>|           <C-End>         1  same as "G"
|<C-Home>|          <C-Home>        1  same as "gg"
|<C-Left>|          <C-Left>        1  same as "b"
|<C-LeftMouse>|     <C-LeftMouse>      ":ta" to the keyword at the mouse click
|<C-Right>|         <C-Right>       1  same as "w"
|<C-RightMouse>|    <C-RightMouse>    same as "CTRL-T"
|<Del>|             ["x]<Del>       2  same as "x"
|N<Del>|            {count}<Del>       remove the last digit from {count}
|<Down>|            <Down>          1  same as "j"
|<End>|             <End>           1  same as "$"
|<F1>|              <F1>                same as <Help>
|<Help>|            <Help>              open a help window
|<Home>|            <Home>          1  same as "0"
|<Insert>|          <Insert>        2  same as "i"
|<Left>|            <Left>          1  same as "h"
|<LeftMouse>|       <LeftMouse>     1  move cursor to the mouse click position
|<MiddleMouse>|     <MiddleMouse>   2  same as "gP" at the mouse click position
|<PageDown>|        <PageDown>          same as CTRL-F
|<PageUp>|          <PageUp>            same as CTRL-B
|<Right>|           <Right>         1  same as "l"
|<RightMouse>|      <RightMouse>       start Visual mode, move cursor to the mouse
                                       click position
|<S-Down>|          <S-Down>        1  same as CTRL-F
|<S-Left>|          <S-Left>        1  same as "b"
|<S-LeftMouse>|     <S-LeftMouse>      same as "*" at the mouse click position
|<S-Right>|         <S-Right>       1  same as "w"
|<S-RightMouse>|    <S-RightMouse>    same as "#" at the mouse click position
|<S-Up>|            <S-Up>          1  same as CTRL-B
|<Undo>|            <Undo>          2  same as "u"
|<Up>|              <Up>            1  same as "k"
|<ScrollWheelDown>|     <ScrollWheelDown>       move window three lines down
|<S-ScrollWheelDown>|   <S-ScrollWheelDown>     move window one page down
|<ScrollWheelUp>|       <ScrollWheelUp>         move window three lines up
|<S-ScrollWheelUp>|     <S-ScrollWheelUp>       move window one page up
|<ScrollWheelLeft>|     <ScrollWheelLeft>       move window six columns left
|<S-ScrollWheelLeft>|   <S-ScrollWheelLeft>     move window one page left
|<ScrollWheelRight>|    <ScrollWheelRight>      move window six columns right
|<S-ScrollWheelRight>|  <S-ScrollWheelRight>    move window one page right

==============================================================================
2.1 Text objects                                                *objects*

These can be used after an operator or in Visual mode to select an object.
```

```
tag             command           action in op-pending and Visual mode ~
-------------------------------------------------------------------------------
|v_aquote|      a"                double quoted string
|v_a'|          a'                single quoted string
|v_a(|          a(                same as ab
|v_a)|          a)                same as ab
|v_a<|          a<                "a <>" from '<' to the matching '>'
|v_a>|          a>                same as a<
|v_aB|          aB                "a Block" from "[{" to "]}" (with brackets)
|v_aW|          aW                "a WORD" (with white space)
|v_a[|          a[                "a []" from '[' to the matching ']'
|v_a]|          a]                same as a[
|v_a`|          a`                string in backticks
|v_ab|          ab                "a block" from "[(" to "])" (with braces)
|v_ap|          ap                "a paragraph" (with white space)
|v_as|          as                "a sentence" (with white space)
|v_at|          at                "a tag block" (with white space)
|v_aw|          aw                "a word" (with white space)
|v_a{|          a{                same as aB
|v_a}|          a}                same as aB
|v_iquote|      i"                double quoted string without the quotes
|v_i'|          i'                single quoted string without the quotes
|v_i(|          i(                same as ib
|v_i)|          i)                same as ib
|v_i<|          i<                "inner <>" from '<' to the matching '>'
|v_i>|          i>                same as i<
|v_iB|          iB                "inner Block" from "[{" and "]}"
|v_iW|          iW                "inner WORD"
|v_i[|          i[                "inner []" from '[' to the matching ']'
|v_i]|          i]                same as i[
|v_i`|          i`                string in backticks without the backticks
|v_ib|          ib                "inner block" from "[(" to "])"
|v_ip|          ip                "inner paragraph"
|v_is|          is                "inner sentence"
|v_it|          it                "inner tag block"
|v_iw|          iw                "inner word"
|v_i{|          i{                same as iB
|v_i}|          i}                same as iB

===============================================================================
2.2 Window commands                                             *CTRL-W*

tag             command           action in Normal mode       ~
-------------------------------------------------------------------------------
|CTRL-W_CTRL-B| CTRL-W CTRL-B     same as "CTRL-W b"
|CTRL-W_CTRL-C| CTRL-W CTRL-C     same as "CTRL-W c"
|CTRL-W_CTRL-D| CTRL-W CTRL-D     same as "CTRL-W d"
|CTRL-W_CTRL-F| CTRL-W CTRL-F     same as "CTRL-W f"
                CTRL-W CTRL-G     same as "CTRL-W g .."
|CTRL-W_CTRL-H| CTRL-W CTRL-H     same as "CTRL-W h"
|CTRL-W_CTRL-I| CTRL-W CTRL-I     same as "CTRL-W i"
|CTRL-W_CTRL-J| CTRL-W CTRL-J     same as "CTRL-W j"
|CTRL-W_CTRL-K| CTRL-W CTRL-K     same as "CTRL-W k"
|CTRL-W_CTRL-L| CTRL-W CTRL-L     same as "CTRL-W l"
|CTRL-W_CTRL-N| CTRL-W CTRL-N     same as "CTRL-W n"
|CTRL-W_CTRL-O| CTRL-W CTRL-O     same as "CTRL-W o"
|CTRL-W_CTRL-P| CTRL-W CTRL-P     same as "CTRL-W p"
|CTRL-W_CTRL-Q| CTRL-W CTRL-Q     same as "CTRL-W q"
|CTRL-W_CTRL-R| CTRL-W CTRL-R     same as "CTRL-W r"
|CTRL-W_CTRL-S| CTRL-W CTRL-S     same as "CTRL-W s"
|CTRL-W_CTRL-T| CTRL-W CTRL-T     same as "CTRL-W t"
```

```
|CTRL-W_CTRL-V| CTRL-W CTRL-V      same as "CTRL-W v"
|CTRL-W_CTRL-W| CTRL-W CTRL-W      same as "CTRL-W w"
|CTRL-W_CTRL-X| CTRL-W CTRL-X      same as "CTRL-W x"
|CTRL-W_CTRL-Z| CTRL-W CTRL-Z      same as "CTRL-W z"
|CTRL-W_CTRL-]| CTRL-W CTRL-]      same as "CTRL-W ]"
|CTRL-W_CTRL-^| CTRL-W CTRL-^      same as "CTRL-W ^"
|CTRL-W_CTRL-_| CTRL-W CTRL-_      same as "CTRL-W _"
|CTRL-W_quote|  CTRL-W "           terminal window: paste register
|CTRL-W_+|      CTRL-W +           increase current window height N lines
|CTRL-W_-|      CTRL-W -           decrease current window height N lines
|CTRL-W_.|      CTRL-W .           terminal window: type CTRL-W
|CTRL-W_:|      CTRL-W :           same as |:|, edit a command line
|CTRL-W_<|      CTRL-W <           decrease current window width N columns
|CTRL-W_=|      CTRL-W =           make all windows the same height & width
|CTRL-W_>|      CTRL-W >           increase current window width N columns
|CTRL-W_H|      CTRL-W H           move current window to the far left
|CTRL-W_J|      CTRL-W J           move current window to the very bottom
|CTRL-W_K|      CTRL-W K           move current window to the very top
|CTRL-W_L|      CTRL-W L           move current window to the far right
|CTRL-W_N|      CTRL-W N           terminal window: go to Terminal Normal mode
|CTRL-W_P|      CTRL-W P           go to preview window
|CTRL-W_R|      CTRL-W R           rotate windows upwards N times
|CTRL-W_S|      CTRL-W S           same as "CTRL-W s"
|CTRL-W_T|      CTRL-W T           move current window to a new tab page
|CTRL-W_W|      CTRL-W W           go to N previous window (wrap around)
|CTRL-W_]|      CTRL-W ]           split window and jump to tag under cursor
|CTRL-W_^|      CTRL-W ^           split current window and edit alternate
                                   file N
|CTRL-W__|      CTRL-W _           set current window height to N (default:
                                   very high)
|CTRL-W_b|      CTRL-W b           go to bottom window
|CTRL-W_c|      CTRL-W c           close current window (like |:close|)
|CTRL-W_d|      CTRL-W d           split window and jump to definition under
                                   the cursor
|CTRL-W_f|      CTRL-W f           split window and edit file name under the
                                   cursor
|CTRL-W_F|      CTRL-W F           split window and edit file name under the
                                   cursor and jump to the line number
                                   following the file name.
|CTRL-W_g_CTRL-]| CTRL-W g CTRL-] split window and do |:tjump| to tag under
                                   cursor
|CTRL-W_g]|     CTRL-W g ]         split window and do |:tselect| for tag
                                   under cursor
|CTRL-W_g}|     CTRL-W g }         do a |:ptjump| to the tag under the cursor
|CTRL-W_gf|     CTRL-W g f         edit file name under the cursor in a new
                                   tab page
|CTRL-W_gF|     CTRL-W g F         edit file name under the cursor in a new
                                   tab page and jump to the line number
                                   following the file name.
|CTRL-W_h|      CTRL-W h           go to Nth left window (stop at first window)
|CTRL-W_i|      CTRL-W i           split window and jump to declaration of
                                   identifier under the cursor
|CTRL-W_j|      CTRL-W j           go N windows down (stop at last window)
|CTRL-W_k|      CTRL-W k           go N windows up (stop at first window)
|CTRL-W_l|      CTRL-W l           go to Nth right window (stop at last window)
|CTRL-W_n|      CTRL-W n           open new window, N lines high
|CTRL-W_o|      CTRL-W o           close all but current window (like |:only|)
|CTRL-W_p|      CTRL-W p           go to previous (last accessed) window
|CTRL-W_q|      CTRL-W q           quit current window (like |:quit|)
|CTRL-W_r|      CTRL-W r           rotate windows downwards N times
|CTRL-W_s|      CTRL-W s           split current window in two parts, new
                                   window N lines high
```

```
|CTRL-W_t|      CTRL-W t           go to top window
|CTRL-W_v|      CTRL-W v           split current window vertically, new window
                                   N columns wide
|CTRL-W_w|      CTRL-W w           go to N next window (wrap around)
|CTRL-W_x|      CTRL-W x           exchange current window with window N
                                   (default: next window)
|CTRL-W_z|      CTRL-W z           close preview window
|CTRL-W_bar|    CTRL-W |           set window width to N columns
|CTRL-W_}|      CTRL-W }           show tag under cursor in preview window
|CTRL-W_<Down>| CTRL-W <Down>      same as "CTRL-W j"
|CTRL-W_<Up>|   CTRL-W <Up>        same as "CTRL-W k"
|CTRL-W_<Left>| CTRL-W <Left>      same as "CTRL-W h"
|CTRL-W_<Right>| CTRL-W <Right>    same as "CTRL-W l"

==============================================================================
2.3 Square bracket commands                                    *[* *]*

tag             char          note action in Normal mode        ~
------------------------------------------------------------------------------
|[_CTRL-D|      [ CTRL-D           jump to first #define found in current and
                                   included files matching the word under the
                                   cursor, start searching at beginning of
                                   current file
|[_CTRL-I|      [ CTRL-I           jump to first line in current and included
                                   files that contains the word under the
                                   cursor, start searching at beginning of
                                   current file
|[#|            [#              1  cursor to N previous unmatched #if, #else
                                   or #ifdef
|['|            ['              1  cursor to previous lowercase mark, on first
                                   non-blank
|[(|            [(              1  cursor N times back to unmatched '('
|[star|         [*              1  same as "[/"
|[`|            [`              1  cursor to previous lowercase mark
|[/|            [/              1  cursor to N previous start of a C comment
|[D|            [D                 list all defines found in current and
                                   included files matching the word under the
                                   cursor, start searching at beginning of
                                   current file
|[I|            [I                 list all lines found in current and
                                   included files that contain the word under
                                   the cursor, start searching at beginning of
                                   current file
|[P|            [P              2  same as "[p"
|[[|            [[              1  cursor N sections backward
|[]|            []              1  cursor N SECTIONS backward
|[c|            [c              1  cursor N times backwards to start of change
|[d|            [d                 show first #define found in current and
                                   included files matching the word under the
                                   cursor, start searching at beginning of
                                   current file
|[f|            [f                 same as "gf"
|[i|            [i                 show first line found in current and
                                   included files that contains the word under
                                   the cursor, start searching at beginning of
                                   current file
|[m|            [m              1  cursor N times back to start of member
                                   function
|[p|            [p              2  like "P", but adjust indent to current line
|[s|            [s              1  move to the previous misspelled word
|[z|            [z              1  move to start of open fold
|[{|            [{              1  cursor N times back to unmatched '{'
```

```
|[<MiddleMouse>| [<MiddleMouse> 2  same as "[p"

|]_CTRL-D|      ] CTRL-D           jump to first #define found in current and
                                   included files matching the word under the
                                   cursor, start searching at cursor position
|]_CTRL-I|      ] CTRL-I           jump to first line in current and included
                                   files that contains the word under the
                                   cursor, start searching at cursor position
|]#|            ]#              1  cursor to N next unmatched #endif or #else
|]'|            ]'              1  cursor to next lowercase mark, on first
                                   non-blank
|])|            ])              1  cursor N times forward to unmatched ')'
|]star|         ]*              1  same as "]/"
|]`|            ]`              1  cursor to next lowercase mark
|]/|            ]/              1  cursor to N next end of a C comment
|]D|            ]D                 list all #defines found in current and
                                   included files matching the word under the
                                   cursor, start searching at cursor position
|]I|            ]I                 list all lines found in current and
                                   included files that contain the word under
                                   the cursor, start searching at cursor
                                   position
|]P|            ]P              2  same as "[p"
|][|            ][              1  cursor N SECTIONS forward
|]]|            ]]              1  cursor N sections forward
|]c|            ]c              1  cursor N times forward to start of change
|]d|            ]d                 show first #define found in current and
                                   included files matching the word under the
                                   cursor, start searching at cursor position
|]f|            ]f                 same as "gf"
|]i|            ]i                 show first line found in current and
                                   included files that contains the word under
                                   the cursor, start searching at cursor
                                   position
|]m|            ]m              1  cursor N times forward to end of member
                                   function
|]p|            ]p              2  like "p", but adjust indent to current line
|]s|            ]s              1  move to next misspelled word
|]z|            ]z              1  move to end of open fold
|]}|            ]}              1  cursor N times forward to unmatched '}'
|]<MiddleMouse>| ]<MiddleMouse> 2  same as "]p"

==============================================================================
2.4 Commands starting with 'g'                                          *g*

tag             char          note action in Normal mode       ~
------------------------------------------------------------------------------
|g_CTRL-A|      g CTRL-A           only when compiled with MEM_PROFILE
                                   defined: dump a memory profile
|g_CTRL-G|      g CTRL-G           show information about current cursor
                                   position
|g_CTRL-H|      g CTRL-H           start Select block mode
|g_CTRL-]|      g CTRL-]           |:tjump| to the tag under the cursor
|g#|            g#              1  like "#", but without using "\<" and "\>"
|g$|            g$              1  when 'wrap' off go to rightmost character of
                                   the current line that is on the screen;
                                   when 'wrap' on go to the rightmost character
                                   of the current screen line
|g&|            g&              2  repeat last ":s" on all lines
|g'|            g'{mark}        1  like |'| but without changing the jumplist
|g`|            g`{mark}        1  like |`| but without changing the jumplist
|gstar|         g*              1  like "*", but without using "\<" and "\>"
```

```
|g+|            g+                 go to newer text state N times
|g,|            g,               1  go to N newer position in change list
|g-|            g-                 go to older text state N times
|g0|            g0               1  when 'wrap' off go to leftmost character of
                                     the current line that is on the screen;
                                     when 'wrap' on go to the leftmost character
                                     of the current screen line
|g8|            g8                 print hex value of bytes used in UTF-8
                                     character under the cursor
|g;|            g;               1  go to N older position in change list
|g<|            g<                 display previous command output
|g?|            g?               2  Rot13 encoding operator
|g?g?|          g??              2  Rot13 encode current line
|g?g?|          g?g?             2  Rot13 encode current line
|gD|            gD               1  go to definition of word under the cursor
                                     in current file
|gE|            gE               1  go backwards to the end of the previous
                                     WORD
|gH|            gH                 start Select line mode
|gI|            gI               2  like "I", but always start in column 1
|gJ|            gJ               2  join lines without inserting space
|gN|            gN             1,2  find the previous match with the last used
                                     search pattern and Visually select it
|gP|            ["x]gP           2  put the text [from register x] before the
                                     cursor N times, leave the cursor after it
|gQ|            gQ                  switch to "Ex" mode with Vim editing
|gR|            gR               2  enter Virtual Replace mode
|gT|            gT                 go to the previous tab page
|gU|            gU{motion}       2  make Nmove text uppercase
|gV|            gV                 don't reselect the previous Visual area
                                     when executing a mapping or menu in Select
                                     mode
|g]|            g]                 :tselect on the tag under the cursor
|g^|            g^               1  when 'wrap' off go to leftmost non-white
                                     character of the current line that is on
                                     the screen; when 'wrap' on go to the
                                     leftmost non-white character of the current
                                     screen line
|g_|            g_               1  cursor to the last CHAR N - 1 lines lower
|ga|            ga                 print ascii value of character under the
                                     cursor
|gd|            gd               1  go to definition of word under the cursor
                                     in current function
|ge|            ge               1  go backwards to the end of the previous
                                     word
|gf|            gf                 start editing the file whose name is under
                                     the cursor
|gF|            gF                 start editing the file whose name is under
                                     the cursor and jump to the line number
                                     following the filename.
|gg|            gg               1  cursor to line N, default first line
|gh|            gh                 start Select mode
|gi|            gi               2  like "i", but first move to the |'^| mark
|gj|            gj               1  like "j", but when 'wrap' on go N screen
                                     lines down
|gk|            gk               1  like "k", but when 'wrap' on go N screen
                                     lines up
|gn|            gn             1,2  find the next match with the last used
                                     search pattern and Visually select it
|gm|            gm               1  go to character at middle of the screenline
|go|            go               1  cursor to byte N in the buffer
|gp|            ["x]gp           2  put the text [from register x] after the
```

```
                                    cursor N times, leave the cursor after it
|gq|            gq{motion}      2  format Nmove text
|gr|            gr{char}        2  virtual replace N chars with {char}
|gs|            gs                 go to sleep for N seconds (default 1)
|gt|            gt                 go to the next tab page
|gu|            gu{motion}      2  make Nmove text lowercase
|gv|            gv                 reselect the previous Visual area
|gw|            gw{motion}      2  format Nmove text and keep cursor
|netrw-gx|      gx                 execute application for file name under the
                                    cursor (only with |netrw| plugin)
|g@|            g@{motion}         call 'operatorfunc'
|g~|            g~{motion}      2  swap case for Nmove text
|g<Down>|       g<Down>          1  same as "gj"
|g<End>|        g<End>           1  same as "g$"
|g<Home>|       g<Home>          1  same as "g0"
|g<LeftMouse>|  g<LeftMouse>        same as <C-LeftMouse>
               g<MiddleMouse>     same as <C-MiddleMouse>
|g<RightMouse>| g<RightMouse>      same as <C-RightMouse>
|g<Up>|         g<Up>            1  same as "gk"

==============================================================================
2.5 Commands starting with 'z'                                          *z*

tag             char          note action in Normal mode        ~
------------------------------------------------------------------------------
|z<CR>|         z<CR>              redraw, cursor line to top of window,
                                    cursor on first non-blank
|zN<CR>|        z{height}<CR>      redraw, make window {height} lines high
|z+|            z+                 cursor on line N (default line below
                                    window), otherwise like "z<CR>"
|z-|            z-                 redraw, cursor line at bottom of window,
                                    cursor on first non-blank
|z.|            z.                 redraw, cursor line to center of window,
                                    cursor on first non-blank
|z=|            z=                 give spelling suggestions
|zA|            zA                 open a closed fold or close an open fold
                                    recursively
|zC|            zC                 close folds recursively
|zD|            zD                 delete folds recursively
|zE|            zE                 eliminate all folds
|zF|            zF                 create a fold for N lines
|zG|            zG                 mark word as good spelled word
|zH|            zH                 when 'wrap' off scroll half a screenwidth
                                    to the right
|zL|            zL                 when 'wrap' off scroll half a screenwidth
                                    to the left
|zM|            zM                 set 'foldlevel' to zero
|zN|            zN                 set 'foldenable'
|zO|            zO                 open folds recursively
|zR|            zR                 set 'foldlevel' to the deepest fold
|zW|            zW                 mark word as wrong (bad) spelled word
|zX|            zX                 re-apply 'foldlevel'
|z^|            z^                 cursor on line N (default line above
                                    window), otherwise like "z-"
|za|            za                 open a closed fold, close an open fold
|zb|            zb                 redraw, cursor line at bottom of window
|zc|            zc                 close a fold
|zd|            zd                 delete a fold
|ze|            ze                 when 'wrap' off scroll horizontally to
                                    position the cursor at the end (right side)
                                    of the screen
|zf|            zf{motion}         create a fold for Nmove text
```

```
|zg|            zg                mark word as good spelled word
|zh|            zh                when 'wrap' off scroll screen N characters
                                   to the right
|zi|            zi                toggle 'foldenable'
|zj|            zj             1  move to the start of the next fold
|zk|            zk             1  move to the end of the previous fold
|zl|            zl                when 'wrap' off scroll screen N characters
                                   to the left
|zm|            zm                subtract one from 'foldlevel'
|zn|            zn                reset 'foldenable'
|zo|            zo                open fold
|zr|            zr                add one to 'foldlevel'
|zs|            zs                when 'wrap' off scroll horizontally to
                                   position the cursor at the start (left
                                   side) of the screen
|zt|            zt                redraw, cursor line at top of window
|zv|            zv                open enough folds to view the cursor line
|zw|            zw                mark word as wrong (bad) spelled word
|zx|            zx                re-apply 'foldlevel' and do "zv"
|zz|            zz                redraw, cursor line at center of window
|z<Left>|       z<Left>           same as "zh"
|z<Right>|      z<Right>          same as "zl"
```

==

3. Visual mode *visual-index*

Most commands in Visual mode are the same as in Normal mode. The ones listed here are those that are different.

```
tag             command       note action in Visual mode        ~
------------------------------------------------------------------------------
|v_CTRL-\_CTRL-N| CTRL-\ CTRL-N    stop Visual mode
|v_CTRL-\_CTRL-G| CTRL-\ CTRL-G    go to mode specified with 'insertmode'
|v_CTRL-A|      CTRL-A          2  add N to number in highlighted text
|v_CTRL-C|      CTRL-C             stop Visual mode
|v_CTRL-G|      CTRL-G             toggle between Visual mode and Select mode
|v_<BS>|        <BS>            2  Select mode: delete highlighted area
|v_CTRL-H|      CTRL-H          2  same as <BS>
|v_CTRL-O|      CTRL-O             switch from Select to Visual mode for one
                                   command
|v_CTRL-V|      CTRL-V             make Visual mode blockwise or stop Visual
                                   mode
|v_CTRL-X|      CTRL-X          2  subtract N from number in highlighted text
|v_<Esc>|       <Esc>              stop Visual mode
|v_CTRL-]|      CTRL-]             jump to highlighted tag
|v_!|           !{filter}       2  filter the highlighted lines through the
                                   external command {filter}
|v_:|           :                 start a command-line with the highlighted
                                   lines as a range
|v_<|           <               2  shift the highlighted lines one
                                   'shiftwidth' left
|v_=|           =               2  filter the highlighted lines through the
                                   external program given with the 'equalprg'
                                   option
|v_>|           >               2  shift the highlighted lines one
                                   'shiftwidth' right
|v_b_A|         A               2  block mode: append same text in all lines,
                                   after the highlighted area
|v_C|           C               2  delete the highlighted lines and start
                                   insert
|v_D|           D               2  delete the highlighted lines
|v_b_I|         I               2  block mode: insert same text in all lines,
```

```
                                   before the highlighted area
|v_J|           J               2  join the highlighted lines
|v_K|           K                  run 'keywordprg' on the highlighted area
|v_O|           O                  Move horizontally to other corner of area.
               Q                   does not start Ex mode
|v_R|           R               2  delete the highlighted lines and start
                                   insert
|v_S|           S               2  delete the highlighted lines and start
                                   insert
|v_U|           U               2  make highlighted area uppercase
|v_V|           V                  make Visual mode linewise or stop Visual
                                   mode
|v_X|           X               2  delete the highlighted lines
|v_Y|           Y                  yank the highlighted lines
|v_aquote|      a"                 extend highlighted area with a double
                                   quoted string
|v_a'|          a'                 extend highlighted area with a single
                                   quoted string
|v_a(|          a(                 same as ab
|v_a)|          a)                 same as ab
|v_a<|          a<                 extend highlighted area with a <> block
|v_a>|          a>                 same as a<
|v_aB|          aB                 extend highlighted area with a {} block
|v_aW|          aW                 extend highlighted area with "a WORD"
|v_a[|          a[                 extend highlighted area with a [] block
|v_a]|          a]                 same as a[
|v_a`|          a`                 extend highlighted area with a backtick
                                   quoted string
|v_ab|          ab                 extend highlighted area with a () block
|v_ap|          ap                 extend highlighted area with a paragraph
|v_as|          as                 extend highlighted area with a sentence
|v_at|          at                 extend highlighted area with a tag block
|v_aw|          aw                 extend highlighted area with "a word"
|v_a{|          a{                 same as aB
|v_a}|          a}                 same as aB
|v_c|           c               2  delete highlighted area and start insert
|v_d|           d               2  delete highlighted area
|v_g_CTRL-A|    g CTRL-A        2  add N to number in highlighted text
|v_g_CTRL-X|    g CTRL-X        2  subtract N from number in highlighted text
|v_gJ|          gJ              2  join the highlighted lines without
                                   inserting spaces
|v_gq|          gq              2  format the highlighted lines
|v_gv|          gv                 exchange current and previous highlighted
                                   area
|v_iquote|      i"                 extend highlighted area with a double
                                   quoted string (without quotes)
|v_i'|          i'                 extend highlighted area with a single
                                   quoted string (without quotes)
|v_i(|          i(                 same as ib
|v_i)|          i)                 same as ib
|v_i<|          i<                 extend highlighted area with inner <> block
|v_i>|          i>                 same as i<
|v_iB|          iB                 extend highlighted area with inner {} block
|v_iW|          iW                 extend highlighted area with "inner WORD"
|v_i[|          i[                 extend highlighted area with inner [] block
|v_i]|          i]                 same as i[
|v_i`|          i`                 extend highlighted area with a backtick
                                   quoted string (without the backticks)
|v_ib|          ib                 extend highlighted area with inner () block
|v_ip|          ip                 extend highlighted area with inner paragraph
|v_is|          is                 extend highlighted area with inner sentence
|v_it|          it                 extend highlighted area with inner tag block
```

```
|v_iw|          iw                 extend highlighted area with "inner word"
|v_i{|          i{                 same as iB
|v_i}|          i}                 same as iB
|v_o|           o                  move cursor to other corner of area
|v_r|           r              2   delete highlighted area and start insert
|v_s|           s              2   delete highlighted area and start insert
|v_u|           u              2   make highlighted area lowercase
|v_v|           v                  make Visual mode characterwise or stop
                                   Visual mode
|v_x|           x              2   delete the highlighted area
|v_y|           y                  yank the highlighted area
|v_~|           ~              2   swap case for the highlighted area

==============================================================================
4. Command-line editing                                  *ex-edit-index*

Get to the command-line with the ':', '!', '/' or '?' commands.
Normal characters are inserted at the current cursor position.
"Completion" below refers to context-sensitive completion.  It will complete
file names, tags, commands etc. as appropriate.

tag             command       action in Command-line editing mode       ~
------------------------------------------------------------------------------
                CTRL-@          not used
|c_CTRL-A|      CTRL-A          do completion on the pattern in front of the
                                cursor and insert all matches
|c_CTRL-B|      CTRL-B          cursor to begin of command-line
|c_CTRL-C|      CTRL-C          same as <Esc>
|c_CTRL-D|      CTRL-D          list completions that match the pattern in
                                front of the cursor
|c_CTRL-E|      CTRL-E          cursor to end of command-line
|'cedit'|       CTRL-F          default value for 'cedit': opens the
                                command-line window; otherwise not used
|c_CTRL-G|      CTRL-G          next match when 'incsearch' is active
|c_<BS>|        <BS>            delete the character in front of the cursor
|c_digraph|     {char1} <BS> {char2}
                                enter digraph when 'digraph' is on
|c_CTRL-H|      CTRL-H          same as <BS>
|c_<Tab>|       <Tab>           if 'wildchar' is <Tab>: Do completion on
                                the pattern in front of the cursor
|c_<S-Tab>|     <S-Tab>         same as CTRL-P
|c_wildchar|    'wildchar'      Do completion on the pattern in front of the
                                cursor (default: <Tab>)
|c_CTRL-I|      CTRL-I          same as <Tab>
|c_<NL>|        <NL>            same as <CR>
|c_CTRL-J|      CTRL-J          same as <CR>
|c_CTRL-K|      CTRL-K {char1} {char2}
                                enter digraph
|c_CTRL-L|      CTRL-L          do completion on the pattern in front of the
                                cursor and insert the longest common part
|c_<CR>|        <CR>            execute entered command
|c_CTRL-M|      CTRL-M          same as <CR>
|c_CTRL-N|      CTRL-N          after using 'wildchar' with multiple matches:
                                go to next match, otherwise: recall older
                                command-line from history.
                CTRL-O          not used
|c_CTRL-P|      CTRL-P          after using 'wildchar' with multiple matches:
                                go to previous match, otherwise: recall older
                                command-line from history.
|c_CTRL-Q|      CTRL-Q          same as CTRL-V, unless it's used for terminal
                                control flow
|c_CTRL-R|      CTRL-R {0-9a-z"%#*:= CTRL-F CTRL-P CTRL-W CTRL-A}
```

```
                                insert the contents of a register or object
                                under the cursor as if typed
|c_CTRL-R_CTRL-R| CTRL-R CTRL-R {0-9a-z"%#*:= CTRL-F CTRL-P CTRL-W CTRL-A}
                                insert the contents of a register or object
                                under the cursor literally
                CTRL-S          (used for terminal control flow)
|c_CTRL-T|      CTRL-T          previous match when 'incsearch' is active
|c_CTRL-U|      CTRL-U          remove all characters
|c_CTRL-V|      CTRL-V          insert next non-digit literally, insert three
                                digit decimal number as a single byte.
|c_CTRL-W|      CTRL-W          delete the word in front of the cursor
                CTRL-X          not used (reserved for completion)
                CTRL-Y          copy (yank) modeless selection
                CTRL-Z          not used (reserved for suspend)
|c_<Esc>|       <Esc>           abandon command-line without executing it
|c_CTRL-[|      CTRL-[          same as <Esc>
|c_CTRL-\_CTRL-N| CTRL-\ CTRL-N go to Normal mode, abandon command-line
|c_CTRL-\_CTRL-G| CTRL-\ CTRL-G go to mode specified with 'insertmode',
                                abandon command-line
                CTRL-\ a - d    reserved for extensions
|c_CTRL-\_e|    CTRL-\ e {expr} replace the command line with the result of
                                {expr}
                CTRL-\ f - z    reserved for extensions
                CTRL-\ others   not used
|c_CTRL-]|      CTRL-]          trigger abbreviation
|c_CTRL-^|      CTRL-^          toggle use of |:lmap| mappings
|c_CTRL-_|      CTRL-_          when 'allowrevins' set: change language
                                (Hebrew, Farsi)
|c_<Del>|       <Del>           delete the character under the cursor

|c_<Left>|      <Left>          cursor left
|c_<S-Left>|    <S-Left>        cursor one word left
|c_<C-Left>|    <C-Left>        cursor one word left
|c_<Right>|     <Right>         cursor right
|c_<S-Right>|   <S-Right>       cursor one word right
|c_<C-Right>|   <C-Right>       cursor one word right
|c_<Up>|        <Up>            recall previous command-line from history that
                                matches pattern in front of the cursor
|c_<S-Up>|      <S-Up>          recall previous command-line from history
|c_<Down>|      <Down>          recall next command-line from history that
                                matches pattern in front of the cursor
|c_<S-Down>|    <S-Down>        recall next command-line from history
|c_<Home>|      <Home>          cursor to start of command-line
|c_<End>|       <End>           cursor to end of command-line
|c_<PageDown>|  <PageDown>      same as <S-Down>
|c_<PageUp>|    <PageUp>        same as <S-Up>
|c_<Insert>|    <Insert>        toggle insert/overstrike mode
|c_<LeftMouse>| <LeftMouse>     cursor at mouse click

You found it, Arthur!                           *holy-grail* *:smile*

==============================================================================
5. EX commands                                  *ex-cmd-index* *:index*

This is a brief but complete listing of all the ":" commands, without
mentioning any arguments.  The optional part of the command name is inside [].
The commands are sorted on the non-optional part of their name.

tag           command         action ~
------------------------------------------------------------------------------
|:!|           :!              filter lines or execute an external command
|:!!|          :!!             repeat last ":!" command
```

```
|:#|            :#              same as ":number"
|:&|            :&              repeat last ":substitute"
|:star|         :*              execute contents of a register
|:<|            :<              shift lines one 'shiftwidth' left
|:=|            :=              print the cursor line number
|:>|            :>              shift lines one 'shiftwidth' right
|:@|            :@              execute contents of a register
|:@@|           :@@             repeat the previous ":@"
|:Next|         :N[ext]          go to previous file in the argument list
|:Print|        :P[rint]         print lines
|:X|            :X              ask for encryption key
|:append|       :a[ppend]        append text
|:abbreviate|   :ab[breviate]   enter abbreviation
|:abclear|      :abc[lear]       remove all abbreviations
|:aboveleft|    :abo[veleft]     make split window appear left or above
|:all|          :al[l]           open a window for each file in the argument
                                 list
|:amenu|        :am[enu]         enter new menu item for all modes
|:anoremenu|    :an[oremenu]     enter a new menu for all modes that will not
                                 be remapped
|:args|         :ar[gs]          print the argument list
|:argadd|       :arga[dd]        add items to the argument list
|:argdelete|    :argd[elete]     delete items from the argument list
|:argedit|      :arge[dit]       add item to the argument list and edit it
|:argdo|        :argdo           do a command on all items in the argument list
|:argglobal|    :argg[lobal]     define the global argument list
|:arglocal|     :argl[ocal]      define a local argument list
|:argument|     :argu[ment]      go to specific file in the argument list
|:ascii|        :as[cii]         print ascii value of character under the cursor
|:autocmd|      :au[tocmd]       enter or show autocommands
|:augroup|      :aug[roup]       select the autocommand group to use
|:aunmenu|      :aun[menu]       remove menu for all modes
|:buffer|       :b[uffer]        go to specific buffer in the buffer list
|:bNext|        :bN[ext]         go to previous buffer in the buffer list
|:ball|         :ba[ll]          open a window for each buffer in the buffer list
|:badd|         :bad[d]          add buffer to the buffer list
|:bdelete|      :bd[elete]       remove a buffer from the buffer list
|:behave|       :be[have]        set mouse and selection behavior
|:belowright|   :bel[owright]    make split window appear right or below
|:bfirst|       :bf[irst]        go to first buffer in the buffer list
|:blast|        :bl[ast]         go to last buffer in the buffer list
|:bmodified|    :bm[odified]     go to next buffer in the buffer list that has
                                 been modified
|:bnext|        :bn[ext]         go to next buffer in the buffer list
|:botright|     :bo[tright]      make split window appear at bottom or far right
|:bprevious|    :bp[revious]     go to previous buffer in the buffer list
|:brewind|      :br[ewind]       go to first buffer in the buffer list
|:break|        :brea[k]         break out of while loop
|:breakadd|     :breaka[dd]      add a debugger breakpoint
|:breakdel|     :breakd[el]      delete a debugger breakpoint
|:breaklist|    :breakl[ist]     list debugger breakpoints
|:browse|       :bro[wse]        use file selection dialog
|:bufdo|        :bufdo           execute command in each listed buffer
|:buffers|      :buffers         list all files in the buffer list
|:bunload|      :bun[load]       unload a specific buffer
|:bwipeout|     :bw[ipeout]      really delete a buffer
|:change|       :c[hange]        replace a line or series of lines
|:cNext|        :cN[ext]         go to previous error
|:cNfile|       :cNf[ile]        go to last error in previous file
|:cabbrev|      :ca[bbrev]       like ":abbreviate" but for Command-line mode
|:cabclear|     :cabc[lear]      clear all abbreviations for Command-line mode
|:caddbuffer|   :cad[dbuffer]    add errors from buffer
```

```
|:caddexpr|     :cadde[xpr]     add errors from expr
|:caddfile|     :caddf[ile]     add error message to current quickfix list
|:call|         :cal[l]         call a function
|:catch|        :cat[ch]        part of a :try command
|:cbottom|      :cbo[ttom]      scroll to the bottom of the quickfix window
|:cbuffer|      :cb[uffer]      parse error messages and jump to first error
|:cc|           :cc             go to specific error
|:cclose|       :ccl[ose]       close quickfix window
|:cd|           :cd             change directory
|:cdo|          :cdo            execute command in each valid error list entry
|:cfdo|         :cfd[o]         execute command in each file in error list
|:center|       :ce[nter]       format lines at the center
|:cexpr|        :cex[pr]        read errors from expr and jump to first
|:cfile|        :cf[ile]        read file with error messages and jump to first
|:cfirst|       :cfir[st]       go to the specified error, default first one
|:cgetbuffer|   :cgetb[uffer]   get errors from buffer
|:cgetexpr|     :cgete[xpr]     get errors from expr
|:cgetfile|     :cg[etfile]     read file with error messages
|:changes|      :changes         print the change list
|:chdir|        :chd[ir]         change directory
|:checkpath|    :che[ckpath]    list included files
|:checktime|    :checkt[ime]    check timestamp of loaded buffers
|:chistory|     :chi[story]     list the error lists
|:clast|        :cla[st]        go to the specified error, default last one
|:clearjumps|   :cle[arjumps]   clear the jump list
|:clist|        :cl[ist]        list all errors
|:close|        :clo[se]        close current window
|:cmap|         :cm[ap]         like ":map" but for Command-line mode
|:cmapclear|    :cmapc[lear]    clear all mappings for Command-line mode
|:cmenu|        :cme[nu]        add menu for Command-line mode
|:cnext|        :cn[ext]        go to next error
|:cnewer|       :cnew[er]       go to newer error list
|:cnfile|       :cnf[ile]       go to first error in next file
|:cnoremap|     :cno[remap]     like ":noremap" but for Command-line mode
|:cnoreabbrev|  :cnorea[bbrev]  like ":noreabbrev" but for Command-line mode
|:cnoremenu|    :cnoreme[nu]    like ":noremenu" but for Command-line mode
|:copy|         :co[py]         copy lines
|:colder|       :col[der]       go to older error list
|:colorscheme|  :colo[rscheme]  load a specific color scheme
|:command|      :com[mand]      create user-defined command
|:comclear|     :comc[lear]     clear all user-defined commands
|:compiler|     :comp[iler]     do settings for a specific compiler
|:continue|     :con[tinue]     go back to :while
|:confirm|      :conf[irm]      prompt user when confirmation required
|:copen|        :cope[n]        open quickfix window
|:cprevious|    :cp[revious]    go to previous error
|:cpfile|       :cpf[ile]       go to last error in previous file
|:cquit|        :cq[uit]        quit Vim with an error code
|:crewind|      :cr[ewind]      go to the specified error, default first one
|:cscope|       :cs[cope]       execute cscope command
|:cstag|        :cst[ag]        use cscope to jump to a tag
|:cunmap|       :cu[nmap]       like ":unmap" but for Command-line mode
|:cunabbrev|    :cuna[bbrev]    like ":unabbrev" but for Command-line mode
|:cunmenu|      :cunme[nu]      remove menu for Command-line mode
|:cwindow|      :cw[indow]      open or close quickfix window
|:delete|       :d[elete]       delete lines
|:delmarks|     :delm[arks]     delete marks
|:debug|        :deb[ug]         run a command in debugging mode
|:debuggreedy|  :debugg[reedy]  read debug mode commands from normal input
|:delcommand|   :delc[ommand]   delete user-defined command
|:delfunction|  :delf[unction]  delete a user function
|:diffupdate|   :dif[fupdate]   update 'diff' buffers
```

```
|:diffget|      :diffg[et]      remove differences in current buffer
|:diffoff|      :diffo[ff]      switch off diff mode
|:diffpatch|    :diffp[atch]    apply a patch and show differences
|:diffput|      :diffpu[t]      remove differences in other buffer
|:diffsplit|    :diffs[plit]    show differences with another file
|:diffthis|     :diffthis       make current window a diff window
|:digraphs|     :dig[raphs]     show or enter digraphs
|:display|      :di[splay]      display registers
|:djump|        :dj[ump]        jump to #define
|:dl|           :dl             short for |:delete| with the 'l' flag
|:del|          :del[ete]l      short for |:delete| with the 'l' flag
|:dlist|        :dli[st]        list #defines
|:doautocmd|    :do[autocmd]    apply autocommands to current buffer
|:doautoall|    :doautoa[ll]    apply autocommands for all loaded buffers
|:dp|           :d[elete]p      short for |:delete| with the 'p' flag
|:drop|         :dr[op]         jump to window editing file or edit file in
                                current window
|:dsearch|      :ds[earch]      list one #define
|:dsplit|       :dsp[lit]       split window and jump to #define
|:edit|         :e[dit]         edit a file
|:earlier|      :ea[rlier]      go to older change, undo
|:echo|         :ec[ho]         echoes the result of expressions
|:echoerr|      :echoe[rr]      like :echo, show like an error and use history
|:echohl|       :echoh[l]       set highlighting for echo commands
|:echomsg|      :echom[sg]      same as :echo, put message in history
|:echon|        :echon          same as :echo, but without <EOL>
|:else|         :el[se]         part of an :if command
|:elseif|       :elsei[f]       part of an :if command
|:emenu|        :em[enu]        execute a menu by name
|:endif|        :en[dif]        end previous :if
|:endfor|       :endfo[r]       end previous :for
|:endfunction|  :endf[unction]  end of a user function
|:endtry|       :endt[ry]       end previous :try
|:endwhile|     :endw[hile]     end previous :while
|:enew|         :ene[w]         edit a new, unnamed buffer
|:ex|           :ex             same as ":edit"
|:execute|      :exe[cute]      execute result of expressions
|:exit|         :exi[t]         same as ":xit"
|:exusage|      :exu[sage]      overview of Ex commands
|:file|         :f[ile]         show or set the current file name
|:files|        :files          list all files in the buffer list
|:filetype|     :filet[ype]     switch file type detection on/off
|:filter|       :filt[er]       filter output of following command
|:find|         :fin[d]         find file in 'path' and edit it
|:finally|      :fina[lly]      part of a :try command
|:finish|       :fini[sh]       quit sourcing a Vim script
|:first|        :fir[st]        go to the first file in the argument list
|:fixdel|       :fix[del]       set key code of <Del>
|:fold|         :fo[ld]         create a fold
|:foldclose|    :foldc[lose]    close folds
|:folddoopen|   :foldd[oopen]   execute command on lines not in a closed fold
|:folddoclosed| :folddoc[losed] execute command on lines in a closed fold
|:foldopen|     :foldo[pen]     open folds
|:for|          :for            for loop
|:function|     :fu[nction]     define a user function
|:global|       :g[lobal]       execute commands for matching lines
|:goto|         :go[to]         go to byte in the buffer
|:grep|         :gr[ep]         run 'grepprg' and jump to first match
|:grepadd|      :grepa[dd]      like :grep, but append to current list
|:gui|          :gu[i]          start the GUI
|:gvim|         :gv[im]         start the GUI
|:hardcopy|     :ha[rdcopy]     send text to the printer
```

```
|:help|          :h[elp]          open a help window
|:helpclose|     :helpc[lose]     close one help window
|:helpfind|      :helpf[ind]      dialog to open a help window
|:helpgrep|      :helpg[rep]      like ":grep" but searches help files
|:helptags|      :helpt[ags]      generate help tags for a directory
|:highlight|     :hi[ghlight]     specify highlighting methods
|:hide|          :hid[e]          hide current buffer for a command
|:history|       :his[tory]       print a history list
|:insert|        :i[nsert]        insert text
|:iabbrev|       :ia[bbrev]       like ":abbrev" but for Insert mode
|:iabclear|      :iabc[lear]      like ":abclear" but for Insert mode
|:if|            :if             execute commands when condition met
|:ijump|         :ij[ump]         jump to definition of identifier
|:ilist|         :il[ist]         list lines where identifier matches
|:imap|          :im[ap]          like ":map" but for Insert mode
|:imapclear|     :imapc[lear]     like ":mapclear" but for Insert mode
|:imenu|         :ime[nu]         add menu for Insert mode
|:inoremap|      :ino[remap]      like ":noremap" but for Insert mode
|:inoreabbrev|  :inorea[bbrev]   like ":noreabbrev" but for Insert mode
|:inoremenu|    :inoreme[nu]     like ":noremenu" but for Insert mode
|:intro|         :int[ro]         print the introductory message
|:isearch|       :is[earch]       list one line where identifier matches
|:isplit|        :isp[lit]        split window and jump to definition of
                                  identifier
|:iunmap|        :iu[nmap]        like ":unmap" but for Insert mode
|:iunabbrev|     :iuna[bbrev]     like ":unabbrev" but for Insert mode
|:iunmenu|       :iunme[nu]       remove menu for Insert mode
|:join|          :j[oin]          join lines
|:jumps|         :ju[mps]         print the jump list
|:k|             :k              set a mark
|:keepalt|       :keepa[lt]       following command keeps the alternate file
|:keepmarks|     :kee[pmarks]     following command keeps marks where they are
|:keepjumps|     :keepj[umps]     following command keeps jumplist and marks
|:keeppatterns|  :keepp[atterns]  following command keeps search pattern history
|:lNext|         :lN[ext]         go to previous entry in location list
|:lNfile|        :lNf[ile]        go to last entry in previous file
|:list|          :l[ist]          print lines
|:laddexpr|      :lad[dexpr]      add locations from expr
|:laddbuffer|    :laddb[uffer]    add locations from buffer
|:laddfile|      :laddf[ile]      add locations to current location list
|:last|          :la[st]          go to the last file in the argument list
|:language|      :lan[guage]      set the language (locale)
|:later|         :lat[er]         go to newer change, redo
|:lbottom|       :lbo[ttom]       scroll to the bottom of the location window
|:lbuffer|       :lb[uffer]       parse locations and jump to first location
|:lcd|           :lc[d]           change directory locally
|:lchdir|        :lch[dir]        change directory locally
|:lclose|        :lcl[ose]        close location window
|:lcscope|       :lcs[cope]       like ":cscope" but uses location list
|:ldo|           :ld[o]           execute command in valid location list entries
|:lfdo|          :lfd[o]          execute command in each file in location list
|:left|          :le[ft]          left align lines
|:leftabove|     :lefta[bove]     make split window appear left or above
|:let|           :let             assign a value to a variable or option
|:lexpr|         :lex[pr]         read locations from expr and jump to first
|:lfile|         :lf[ile]         read file with locations and jump to first
|:lfirst|        :lfir[st]        go to the specified location, default first one
|:lgetbuffer|    :lgetb[uffer]    get locations from buffer
|:lgetexpr|      :lgete[xpr]      get locations from expr
|:lgetfile|      :lg[etfile]      read file with locations
|:lgrep|         :lgr[ep]         run 'grepprg' and jump to first match
|:lgrepadd|      :lgrepa[dd]      like :grep, but append to current list
```

```
|:lhelpgrep|    :lh[elpgrep]    like ":helpgrep" but uses location list
|:lhistory|     :lhi[story]     list the location lists
|:ll|           :ll             go to specific location
|:llast|        :lla[st]        go to the specified location, default last one
|:llist|        :lli[st]        list all locations
|:lmake|        :lmak[e]        execute external command 'makeprg' and parse
                                error messages
|:lmap|         :lm[ap]          like ":map!" but includes Lang-Arg mode
|:lmapclear|    :lmapc[lear]    like ":mapclear!" but includes Lang-Arg mode
|:lnext|        :lne[xt]        go to next location
|:lnewer|       :lnew[er]       go to newer location list
|:lnfile|       :lnf[ile]       go to first location in next file
|:lnoremap|     :ln[oremap]     like ":noremap!" but includes Lang-Arg mode
|:loadkeymap|   :loadk[eymap]   load the following keymaps until EOF
|:loadview|     :lo[adview]     load view for current window from a file
|:lockmarks|    :loc[kmarks]    following command keeps marks where they are
|:lockvar|      :lockv[ar]      lock variables
|:lolder|       :lol[der]       go to older location list
|:lopen|        :lope[n]        open location window
|:lprevious|    :lp[revious]    go to previous location
|:lpfile|       :lpf[ile]       go to last location in previous file
|:lrewind|      :lr[ewind]      go to the specified location, default first one
|:ls|           :ls             list all buffers
|:ltag|         :lt[ag]         jump to tag and add matching tags to the
                                location list
|:lunmap|       :lu[nmap]       like ":unmap!" but includes Lang-Arg mode
|:lua|          :lua            execute |Lua| command
|:luado|        :luad[o]        execute Lua command for each line
|:luafile|      :luaf[ile]      execute |Lua| script file
|:lvimgrep|     :lv[imgrep]     search for pattern in files
|:lvimgrepadd|  :lvimgrepa[dd]  like :vimgrep, but append to current list
|:lwindow|      :lw[indow]      open or close location window
|:move|         :m[ove]         move lines
|:mark|         :ma[rk]         set a mark
|:make|         :mak[e]         execute external command 'makeprg' and parse
                                error messages
|:map|          :map            show or enter a mapping
|:mapclear|     :mapc[lear]     clear all mappings for Normal and Visual mode
|:marks|        :marks          list all marks
|:match|        :mat[ch]        define a match to highlight
|:menu|         :me[nu]         enter a new menu item
|:menutranslate|  :menut[ranslate] add a menu translation item
|:messages|     :mes[sages]     view previously displayed messages
|:mkexrc|       :mk[exrc]       write current mappings and settings to a file
|:mksession|    :mks[ession]    write session info to a file
|:mkspell|      :mksp[ell]      produce .spl spell file
|:mkvimrc|      :mkv[imrc]      write current mappings and settings to a file
|:mkview|       :mkvie[w]       write view of current window to a file
|:mode|         :mod[e]         show or change the screen mode
|:mzscheme|     :mz[scheme]     execute MzScheme command
|:mzfile|       :mzf[ile]       execute MzScheme script file
|:nbclose|      :nbc[lose]      close the current Netbeans session
|:nbkey|        :nb[key]        pass a key to Netbeans
|:nbstart|      :nbs[art]       start a new Netbeans session
|:next|         :n[ext]         go to next file in the argument list
|:new|          :new            create a new empty window
|:nmap|         :nm[ap]         like ":map" but for Normal mode
|:nmapclear|    :nmapc[lear]    clear all mappings for Normal mode
|:nmenu|        :nme[nu]        add menu for Normal mode
|:nnoremap|     :nn[oremap]     like ":noremap" but for Normal mode
|:nnoremenu|    :nnoreme[nu]    like ":noremenu" but for Normal mode
|:noautocmd|    :noa[utocmd]    following commands don't trigger autocommands
```

```
|:noremap|      :no[remap]      enter a mapping that will not be remapped
|:nohlsearch|   :noh[lsearch]   suspend 'hlsearch' highlighting
|:noreabbrev|   :norea[bbrev]   enter an abbreviation that will not be
                                remapped
|:noremenu|     :noreme[nu]     enter a menu that will not be remapped
|:normal|       :norm[al]       execute Normal mode commands
|:noswapfile|   :nos[wapfile]   following commands don't create a swap file
|:number|       :nu[mber]       print lines with line number
|:nunmap|       :nun[map]       like ":unmap" but for Normal mode
|:nunmenu|      :nunme[nu]      remove menu for Normal mode
|:oldfiles|     :ol[dfiles]     list files that have marks in the viminfo file
|:open|         :o[pen]         start open mode (not implemented)
|:omap|         :om[ap]         like ":map" but for Operator-pending mode
|:omapclear|    :omapc[lear]    remove all mappings for Operator-pending mode
|:omenu|        :ome[nu]        add menu for Operator-pending mode
|:only|         :on[ly]         close all windows except the current one
|:onoremap|     :ono[remap]     like ":noremap" but for Operator-pending mode
|:onoremenu|    :onoreme[nu]    like ":noremenu" but for Operator-pending mode
|:options|      :opt[ions]      open the options-window
|:ounmap|       :ou[nmap]       like ":unmap" but for Operator-pending mode
|:ounmenu|      :ounme[nu]      remove menu for Operator-pending mode
|:ownsyntax|    :ow[nsyntax]    set new local syntax highlight for this window
|:packadd|      :pa[ckadd]      add a plugin from 'packpath'
|:packloadall|  :packl[oadall]  load all packages under 'packpath'
|:pclose|       :pc[lose]       close preview window
|:pedit|        :ped[it]        edit file in the preview window
|:perl|         :pe[rl]         execute Perl command
|:print|        :p[rint]         print lines
|:profdel|      :profd[el]       stop profiling a function or script
|:profile|      :prof[ile]       profiling functions and scripts
|:promptfind|   :pro[mptfind]   open GUI dialog for searching
|:promptrepl|   :promptr[epl]   open GUI dialog for search/replace
|:perldo|       :perld[o]        execute Perl command for each line
|:pop|          :po[p]          jump to older entry in tag stack
|:popup|        :popu[p]        popup a menu by name
|:ppop|         :pp[op]         ":pop" in preview window
|:preserve|     :pre[serve]     write all text to swap file
|:previous|     :prev[ious]     go to previous file in argument list
|:psearch|      :ps[earch]      like ":ijump" but shows match in preview window
|:ptag|         :pt[ag]         show tag in preview window
|:ptNext|       :ptN[ext]       |:tNext| in preview window
|:ptfirst|      :ptf[irst]      |:trewind| in preview window
|:ptjump|       :ptj[ump]       |:tjump| and show tag in preview window
|:ptlast|       :ptl[ast]       |:tlast| in preview window
|:ptnext|       :ptn[ext]       |:tnext| in preview window
|:ptprevious|   :ptp[revious]   |:tprevious| in preview window
|:ptrewind|     :ptr[ewind]     |:trewind| in preview window
|:ptselect|     :pts[elect]     |:tselect| and show tag in preview window
|:put|          :pu[t]          insert contents of register in the text
|:pwd|          :pw[d]          print current directory
|:py3|          :py3            execute Python 3 command
|:python3|      :python3        same as :py3
|:py3do|        :py3d[o]        execute Python 3 command for each line
|:py3file|      :py3f[ile]      execute Python 3 script file
|:python|       :py[thon]       execute Python command
|:pydo|         :pyd[o]         execute Python command for each line
|:pyfile|       :pyf[ile]       execute Python script file
|:pyx|          :pyx            execute |python_x| command
|:pythonx|      :pythonx        same as :pyx
|:pyxdo|        :pyxd[o]        execute |python_x| command for each line
|:pyxfile|      :pyxf[ile]      execute |python_x| script file
|:quit|         :q[uit]         quit current window (when one window quit Vim)
```

```
|:quitall|      :quita[ll]      quit Vim
|:qall|         :qa[ll]         quit Vim
|:read|         :r[ead]         read file into the text
|:recover|      :rec[over]      recover a file from a swap file
|:redo|         :red[o]         redo one undone change
|:redir|        :redi[r]        redirect messages to a file or register
|:redraw|       :redr[aw]       force a redraw of the display
|:redrawstatus| :redraws[tatus] force a redraw of the status line(s)
|:registers|    :reg[isters]    display the contents of registers
|:resize|       :res[ize]       change current window height
|:retab|        :ret[ab]        change tab size
|:return|       :retu[rn]       return from a user function
|:rewind|       :rew[ind]       go to the first file in the argument list
|:right|        :ri[ght]        right align text
|:rightbelow|   :rightb[elow]   make split window appear right or below
|:ruby|         :rub[y]         execute Ruby command
|:rubydo|       :rubyd[o]       execute Ruby command for each line
|:rubyfile|     :rubyf[ile]     execute Ruby script file
|:rundo|        :rund[o]        read undo information from a file
|:runtime|      :ru[ntime]      source vim scripts in 'runtimepath'
|:rviminfo|     :rv[iminfo]     read from viminfo file
|:substitute|   :s[ubstitute]   find and replace text
|:sNext|        :sN[ext]        split window and go to previous file in
                                argument list
|:sandbox|      :san[dbox]      execute a command in the sandbox
|:sargument|    :sa[rgument]    split window and go to specific file in
                                argument list
|:sall|         :sal[l]         open a window for each file in argument list
|:saveas|       :sav[eas]       save file under another name.
|:sbuffer|      :sb[uffer]      split window and go to specific file in the
                                buffer list
|:sbNext|       :sbN[ext]       split window and go to previous file in the
                                buffer list
|:sball|        :sba[ll]        open a window for each file in the buffer list
|:sbfirst|      :sbf[irst]      split window and go to first file in the
                                buffer list
|:sblast|       :sbl[ast]       split window and go to last file in buffer
                                list
|:sbmodified|   :sbm[odified]   split window and go to modified file in the
                                buffer list
|:sbnext|       :sbn[ext]       split window and go to next file in the buffer
                                list
|:sbprevious|   :sbp[revious]   split window and go to previous file in the
                                buffer list
|:sbrewind|     :sbr[ewind]     split window and go to first file in the
                                buffer list
|:scriptnames|  :scr[iptnames]  list names of all sourced Vim scripts
|:scriptencoding| :scripte[ncoding]     encoding used in sourced Vim script
|:scscope|      :scs[cope]      split window and execute cscope command
|:set|          :se[t]          show or set options
|:setfiletype|  :setf[iletype]  set 'filetype', unless it was set already
|:setglobal|    :setg[lobal]    show global values of options
|:setlocal|     :setl[ocal]     show or set options locally
|:sfind|        :sf[ind]        split current window and edit file in 'path'
|:sfirst|       :sfir[st]       split window and go to first file in the
                                argument list
|:shell|        :sh[ell]        escape to a shell
|:simalt|       :sim[alt]       Win32 GUI: simulate Windows ALT key
|:sign|         :sig[n]         manipulate signs
|:silent|       :sil[ent]       run a command silently
|:sleep|        :sl[eep]        do nothing for a few seconds
|:slast|        :sla[st]        split window and go to last file in the
```

```
                                argument list
|:smagic|       :sm[agic]       :substitute with 'magic'
|:smap|         :smap           like ":map" but for Select mode
|:smapclear|    :smapc[lear]    remove all mappings for Select mode
|:smenu|        :sme[nu]        add menu for Select mode
|:smile|        :smi[le]        make the user happy
|:snext|        :sn[ext]        split window and go to next file in the
                                argument list
|:snomagic|     :sno[magic]     :substitute with 'nomagic'
|:snoremap|     :snor[emap]     like ":noremap" but for Select mode
|:snoremenu|    :snoreme[nu]    like ":noremenu" but for Select mode
|:sort|         :sor[t]         sort lines
|:source|       :so[urce]       read Vim or Ex commands from a file
|:spelldump|    :spelld[ump]    split window and fill with all correct words
|:spellgood|    :spe[llgood]    add good word for spelling
|:spellinfo|    :spelli[nfo]    show info about loaded spell files
|:spellrepall|  :spellr[epall]  replace all bad words like last |z=|
|:spellundo|    :spellu[ndo]    remove good or bad word
|:spellwrong|   :spellw[rong]   add spelling mistake
|:split|        :sp[lit]        split current window
|:sprevious|    :spr[evious]    split window and go to previous file in the
                                argument list
|:srewind|      :sre[wind]      split window and go to first file in the
                                argument list
|:stop|         :st[op]         suspend the editor or escape to a shell
|:stag|         :sta[g]         split window and jump to a tag
|:startinsert|  :star[tinsert]  start Insert mode
|:startgreplace|  :startg[replace] start Virtual Replace mode
|:startreplace| :startr[eplace] start Replace mode
|:stopinsert|   :stopi[nsert]   stop Insert mode
|:stjump|       :stj[ump]       do ":tjump" and split window
|:stselect|     :sts[elect]     do ":tselect" and split window
|:sunhide|      :sun[hide]      same as ":unhide"
|:sunmap|       :sunm[ap]       like ":unmap" but for Select mode
|:sunmenu|      :sunme[nu]      remove menu for Select mode
|:suspend|      :sus[pend]      same as ":stop"
|:sview|        :sv[iew]        split window and edit file read-only
|:swapname|     :sw[apname]     show the name of the current swap file
|:syntax|       :sy[ntax]       syntax highlighting
|:syntime|      :synti[me]      measure syntax highlighting speed
|:syncbind|     :sync[bind]     sync scroll binding
|:t|            :t              same as ":copy"
|:tNext|        :tN[ext]        jump to previous matching tag
|:tabNext|      :tabN[ext]      go to previous tab page
|:tabclose|     :tabc[lose]     close current tab page
|:tabdo|        :tabdo          execute command in each tab page
|:tabedit|      :tabe[dit]      edit a file in a new tab page
|:tabfind|      :tabf[ind]      find file in 'path', edit it in a new tab page
|:tabfirst|     :tabfir[st]     go to first tab page
|:tablast|      :tabl[ast]      go to last tab page
|:tabmove|      :tabm[ove]      move tab page to other position
|:tabnew|       :tabnew         edit a file in a new tab page
|:tabnext|      :tabn[ext]      go to next tab page
|:tabonly|      :tabo[nly]      close all tab pages except the current one
|:tabprevious|  :tabp[revious]  go to previous tab page
|:tabrewind|    :tabr[ewind]    go to first tab page
|:tabs|         :tabs           list the tab pages and what they contain
|:tab|          :tab            create new tab when opening new window
|:tag|          :ta[g]          jump to tag
|:tags|         :tags           show the contents of the tag stack
|:tcl|          :tc[l]          execute Tcl command
|:tcldo|        :tcld[o]        execute Tcl command for each line
```

```
|:tclfile|      :tclf[ile]      execute Tcl script file
|:tearoff|      :te[aroff]      tear-off a menu
|:terminal|     :ter[minal]     open a terminal window
|:tfirst|       :tf[irst]       jump to first matching tag
|:throw|        :th[row]        throw an exception
|:tjump|        :tj[ump]        like ":tselect", but jump directly when there
                                is only one match
|:tlast|        :tl[ast]        jump to last matching tag
|:tmapclear|    :tmapc[lear]    remove all mappings for Terminal-Job mode
|:tmap|         :tma[p]         like ":map" but for Terminal-Job mode
|:tmenu|        :tm[enu]        define menu tooltip
|:tnext|        :tn[ext]        jump to next matching tag
|:tnoremap|     :tno[remap]     like ":noremap" but for Terminal-Job mode
|:topleft|      :to[pleft]      make split window appear at top or far left
|:tprevious|    :tp[revious]    jump to previous matching tag
|:trewind|      :tr[ewind]      jump to first matching tag
|:try|          :try            execute commands, abort on error or exception
|:tselect|      :ts[elect]      list matching tags and select one
|:tunmap|       :tunma[p]       like ":unmap" but for Terminal-Job mode
|:tunmenu|      :tu[nmenu]      remove menu tooltip
|:undo|         :u[ndo]         undo last change(s)
|:undojoin|     :undoj[oin]     join next change with previous undo block
|:undolist|     :undol[ist]     list leafs of the undo tree
|:unabbreviate| :una[bbreviate] remove abbreviation
|:unhide|       :unh[ide]       open a window for each loaded file in the
                                buffer list
|:unlet|        :unl[et]        delete variable
|:unlockvar|    :unlo[ckvar]    unlock variables
|:unmap|        :unm[ap]        remove mapping
|:unmenu|       :unme[nu]       remove menu
|:unsilent|     :uns[ilent]     run a command not silently
|:update|       :up[date]       write buffer if modified
|:vglobal|      :v[global]      execute commands for not matching lines
|:version|      :ve[rsion]      print version number and other info
|:verbose|      :verb[ose]      execute command with 'verbose' set
|:vertical|     :vert[ical]     make following command split vertically
|:vimgrep|      :vim[grep]      search for pattern in files
|:vimgrepadd|   :vimgrepa[dd]   like :vimgrep, but append to current list
|:visual|       :vi[sual]       same as ":edit", but turns off "Ex" mode
|:viusage|      :viu[sage]      overview of Normal mode commands
|:view|         :vie[w]         edit a file read-only
|:vmap|         :vm[ap]         like ":map" but for Visual+Select mode
|:vmapclear|    :vmapc[lear]    remove all mappings for Visual+Select mode
|:vmenu|        :vme[nu]        add menu for Visual+Select mode
|:vnew|         :vne[w]         create a new empty window, vertically split
|:vnoremap|     :vn[oremap]     like ":noremap" but for Visual+Select mode
|:vnoremenu|    :vnoreme[nu]    like ":noremenu" but for Visual+Select mode
|:vsplit|       :vs[plit]       split current window vertically
|:vunmap|       :vu[nmap]       like ":unmap" but for Visual+Select mode
|:vunmenu|      :vunme[nu]      remove menu for Visual+Select mode
|:windo|        :windo          execute command in each window
|:write|        :w[rite]        write to a file
|:wNext|        :wN[ext]        write to a file and go to previous file in
                                argument list
|:wall|         :wa[ll]         write all (changed) buffers
|:while|        :wh[ile]        execute loop for as long as condition met
|:winsize|      :wi[nsize]      get or set window size (obsolete)
|:wincmd|       :winc[md]       execute a Window (CTRL-W) command
|:winpos|       :winp[os]       get or set window position
|:wnext|        :wn[ext]        write to a file and go to next file in
                                argument list
|:wprevious|    :wp[revious]    write to a file and go to previous file in
```

```
                                argument list
|:wq|           :wq               write to a file and quit window or Vim
|:wqall|        :wqa[ll]           write all changed buffers and quit Vim
|:wsverb|       :ws[verb]          pass the verb to workshop over IPC
|:wundo|        :wu[ndo]           write undo information to a file
|:wviminfo|     :wv[iminfo]        write to viminfo file
|:xit|          :x[it]             write if buffer changed and quit window or Vim
|:xall|         :xa[ll]            same as ":wqall"
|:xmapclear|    :xmapc[lear]       remove all mappings for Visual mode
|:xmap|         :xm[ap]            like ":map" but for Visual mode
|:xmenu|        :xme[nu]           add menu for Visual mode
|:xnoremap|     :xn[oremap]        like ":noremap" but for Visual mode
|:xnoremenu|    :xnoreme[nu]       like ":noremenu" but for Visual mode
|:xunmap|       :xu[nmap]          like ":unmap" but for Visual mode
|:xunmenu|      :xunme[nu]         remove menu for Visual mode
|:yank|         :y[ank]            yank lines into a register
|:z|            :z                print some lines
|:~|            :~                repeat last ":substitute"


 vim:tw=78:ts=8:ft=help:norl:
*howto.txt*     For Vim version 8.0.  Last change: 2006 Apr 02


                  VIM REFERENCE MANUAL    by Bram Moolenaar


How to ...                                *howdoi* *how-do-i* *howto* *how-to*

|tutor|                 get started
|:quit|                 exit?  I'm trapped, help me!
|initialization|        initialize Vim
|vimrc-intro|           write a Vim script file (vimrc)
|suspend|               suspend Vim
|usr_11.txt|            recover after a crash
|07.4|                  keep a backup of my file when writing over it

|usr_07.txt|            edit files
|23.4|                  edit binary files
|usr_24.txt|            insert text
|deleting|              delete text
|usr_04.txt|            change text
|04.5|                  copy and move text
|usr_25.txt|            format text
|30.6|                  format comments
|30.2|                  indent C programs
|25.3|                  automatically set indent

|usr_26.txt|            repeat commands
|02.5|                  undo and redo

|usr_03.txt|            move around
|word-motions|          word motions
|left-right-motions|    left-right motions
|up-down-motions|       up-down motions
|object-motions|        text-object motions
|various-motions|       various motions
|object-select|         text-object selection
|'whichwrap'|           move over line breaks
|'virtualedit'|         move to where there is no text
|usr_27.txt|            specify pattern for searches
|tags-and-searches|     do tags and special searches
```

```
|29.4|                  search in include'd files used to find
                        variables, functions, or macros
|K|                     look up manual for the keyword under cursor

|03.7|                  scroll
|'sidescroll'|          scroll horizontally/sideways
|'scrolloff'|           set visible context lines

|mode-switching|        change modes
|04.4|                  use Visual mode
|'insertmode'|          start Vim in Insert mode

|40.1|                  map keys
|24.7|                  create abbreviations

|ins-expandtab|         expand a tab to spaces in Insert mode
|i_CTRL-R|              insert contents of a register in Insert mode
|24.3|                  complete words in Insert mode
|25.1|                  break a line before it gets too long

|20.1|                  do command-line editing
|20.3|                  do command-line completion
|'cmdheight'|           increase the height of command-line
|10.3|                  specify command-line ranges
|40.3|                  specify commands to be executed automatically
                        before/after reading/writing entering/leaving a
                        buffer/window

|'autowrite'|           write automatically
|30.1|                  speedup edit-compile-edit cycle or compile and fix
                        errors within Vim

|options|               set options
|auto-setting|          set options automatically
|term-dependent-settings| set options depending on terminal name
|save-settings|         save settings
|:quote|                comment my .vim files
|'helpheight'|          change the default help height
|'highlight'|           set various highlighting modes
|'title'|               set the window title
|'icon'|                set window icon title
|'report'|              avoid seeing the change messages on every line
|'shortmess'|           avoid |hit-enter| prompts

|mouse-using|           use mouse with Vim
|usr_08.txt|            manage multiple windows and buffers
|gui.txt|               use the gui

|You can't! (yet)|      do dishes using Vim

|usr_06.txt|            switch on syntax highlighting
|2html.vim|             convert a colored file to HTML
|less|                  use Vim like less or more with syntax highlighting

 vim:tw=78:ts=8:ft=help:norl:
*tips.txt*      For Vim version 8.0.  Last change: 2009 Nov 07


                  VIM REFERENCE MANUAL    by Bram Moolenaar


Tips and ideas for using Vim                            *tips*
```

These are just a few that we thought would be helpful for many users.
You can find many more tips on the wiki. The URL can be found on
http://www.vim.org

Don't forget to browse the user manual, it also contains lots of useful tips
|usr_toc.txt|.

==
Editing C programs *C-editing*

There are quite a few features in Vim to help you edit C program files. Here
is an overview with tags to jump to:

|usr_29.txt| Moving through programs chapter in the user manual.
|usr_30.txt| Editing programs chapter in the user manual.
|C-indenting| Automatically set the indent of a line while typing
text.
|=| Re-indent a few lines.
|format-comments| Format comments.

|:checkpath| Show all recursively included files.
|[i| Search for identifier under cursor in current and
included files.
|[_CTRL-I| Jump to match for "[i"
|[I| List all lines in current and included files where
identifier under the cursor matches.
|[d| Search for define under cursor in current and included
files.

|CTRL-]| Jump to tag under cursor (e.g., definition of a
function).
|CTRL-T| Jump back to before a CTRL-] command.
|:tselect| Select one tag out of a list of matching tags.

|gd| Go to Declaration of local variable under cursor.
|gD| Go to Declaration of global variable under cursor.

|gf| Go to file name under the cursor.

|%| Go to matching (), {}, [], /* */, #if, #else, #endif.
|[/| Go to previous start of comment.
|]/| Go to next end of comment.
|[#| Go back to unclosed #if, #ifdef, or #else.

```
|]#|                    Go forward to unclosed #else or #endif.
|[(|                    Go back to unclosed '('
|])|                    Go forward to unclosed ')'
|[{|                    Go back to unclosed '{'
|]}|                    Go forward to unclosed '}'

|v_ab|                  Select "a block" from "[(" to "])", including braces
|v_ib|                  Select "inner block" from "[(" to "])"
|v_aB|                  Select "a block" from "[{" to "]}", including brackets
|v_iB|                  Select "inner block" from "[{" to "]}"
```

==

Finding where identifiers are used *ident-search*

You probably already know that |tags| can be used to jump to the place where a function or variable is defined. But sometimes you wish you could jump to all the places where a function or variable is being used. This is possible in two ways:

1. Using the |:grep| command. This should work on most Unix systems, but can be slow (it reads all files) and only searches in one directory.
2. Using ID utils. This is fast and works in multiple directories. It uses a database to store locations. You will need some additional programs for this to work. And you need to keep the database up to date.

Using the GNU id-tools:

What you need:

- The GNU id-tools installed (mkid is needed to create ID and lid is needed to use the macros).
- An identifier database file called "ID" in the current directory. You can create it with the shell command "mkid file1 file2 ..".

Put this in your .vimrc: >

```
        map _u :call ID_search()<Bar>execute "/\\<" . g:word . "\\>"<CR>
        map _n :n<Bar>execute "/\\<" . g:word . "\\>"<CR>

        function! ID_search()
          let g:word = expand("<cword>")
          let x = system("lid --key=none ". g:word)
          let x = substitute(x, "\n", " ", "g")
          execute "next " . x
        endfun
```

To use it, place the cursor on a word, type "_u" and vim will load the file that contains the word. Search for the next occurrence of the word in the same file with "n". Go to the next file with "_n".

This has been tested with id-utils-3.2 (which is the name of the id-tools archive file on your closest gnu-ftp-mirror).

[the idea for this comes from Andreas Kutschera]

==

Switching screens in an xterm *xterm-screens* *xterm-save-screen*

(From comp.editors, by Juergen Weigert, in reply to a question)

```
:> Another question is that after exiting vim, the screen is left as it
:> was, i.e. the contents of the file I was viewing (editing) was left on
:> the screen. The output from my previous like "ls" were lost,
:> ie. no longer in the scrolling buffer. I know that there is a way to
:> restore the screen after exiting vim or other vi like editors,
```

```
:> I just don't know how. Helps are appreciated. Thanks.
:
:I imagine someone else can answer this.  I assume though that vim and vi do
:the same thing as each other for a given xterm setup.

They not necessarily do the same thing, as this may be a termcap vs.
terminfo problem.  You should be aware that there are two databases for
describing attributes of a particular type of terminal: termcap and
terminfo.  This can cause differences when the entries differ AND when of
the programs in question one uses terminfo and the other uses termcap
(also see |+terminfo|).

In your particular problem, you are looking for the control sequences
^[[?47h and ^[[?47l.  These switch between xterms alternate and main screen
buffer.  As a quick workaround a command sequence like >
        echo -n "^[[?47h"; vim ... ; echo -n "^[[?47l"
may do what you want.  (My notation ^[ means the ESC character, further down
you'll see that the databases use \E instead).

On startup, vim echoes the value of the termcap variable ti (terminfo:
smcup) to the terminal.  When exiting, it echoes te (terminfo: rmcup).  Thus
these two variables are the correct place where the above mentioned control
sequences should go.

Compare your xterm termcap entry (found in /etc/termcap) with your xterm
terminfo entry (retrieved with "infocmp -C xterm").  Both should contain
entries similar to: >
        :te=\E[2J\E[?47l\E8:ti=\E7\E[?47h:

PS: If you find any difference, someone (your sysadmin?) should better check
    the complete termcap and terminfo database for consistency.

NOTE 1: If you recompile Vim with FEAT_XTERM_SAVE defined in feature.h, the
builtin xterm will include the mentioned "te" and "ti" entries.

NOTE 2: If you want to disable the screen switching, and you don't want to
change your termcap, you can add these lines to your .vimrc: >
        :set t_ti= t_te=

==============================================================================
Scrolling in Insert mode                                *scroll-insert*

If you are in insert mode and you want to see something that is just off the
screen, you can use CTRL-X CTRL-E and CTRL-X CTRL-Y to scroll the screen.
                                                |i_CTRL-X_CTRL-E|

To make this easier, you could use these mappings: >
        :inoremap <C-E> <C-X><C-E>
        :inoremap <C-Y> <C-X><C-Y>
(Type this literally, make sure the '<' flag is not in 'cpoptions').
You then lose the ability to copy text from the line above/below the cursor
|i_CTRL-E|.

Also consider setting 'scrolloff' to a larger value, so that you can always see
some context around the cursor.  If 'scrolloff' is bigger than half the window
height, the cursor will always be in the middle and the text is scrolled when
the cursor is moved up/down.

==============================================================================
Smooth scrolling                                        *scroll-smooth*

If you like the scrolling to go a bit smoother, you can use these mappings: >
```

```
        :map <C-U> <C-Y><C-Y><C-Y><C-Y><C-Y><C-Y><C-Y><C-Y><C-Y><C-Y><C-Y><C-
Y><C-Y><C-Y><C-Y>
        :map <C-D> <C-E><C-E><C-E><C-E><C-E><C-E><C-E><C-E><C-E><C-E><C-E><C-
E><C-E><C-E><C-E>

(Type this literally, make sure the '<' flag is not in 'cpoptions').

==============================================================================
Correcting common typing mistakes                       *type-mistakes*

When there are a few words that you keep on typing in the wrong way, make
abbreviations that correct them.  For example: >
        :ab teh the
        :ab fro for

==============================================================================
Counting words, lines, etc.                             *count-items*

To count how often any pattern occurs in the current buffer use the substitute
command and add the 'n' flag to avoid the substitution.  The reported number
of substitutions is the number of items.  Examples: >

        :%s/./&/gn              characters
        :%s/\i\+/&/gn           words
        :%s/^//n                lines
        :%s/the/&/gn            "the" anywhere
        :%s/\<the\>/&/gn        "the" as a word

You might want to reset 'hlsearch' or do ":nohlsearch".
Add the 'e' flag if you don't want an error when there are no matches.

An alternative is using |v_g_CTRL-G| in Visual mode.

If you want to find matches in multiple files use |:vimgrep|.

                                                        *count-bytes*
If you want to count bytes, you can use this:

        Visually select the characters (block is also possible)
        Use "y" to yank the characters
        Use the strlen() function: >
                :echo strlen(@")
A line break is counted for one byte.

==============================================================================
Restoring the cursor position                           *restore-position*

Sometimes you want to write a mapping that makes a change somewhere in the
file and restores the cursor position, without scrolling the text.  For
example, to change the date mark in a file: >
   :map <F2> msHmtgg/Last [cC]hange:\s*/e+1<CR>"_D"=strftime("%Y %b %d")<CR>p'tzt`s

Breaking up saving the position:
        ms      store cursor position in the 's' mark
        H       go to the first line in the window
        mt      store this position in the 't' mark

Breaking up restoring the position:
        't      go to the line previously at the top of the window
        zt      scroll to move this line to the top of the window
        `s      jump to the original position of the cursor
```

For something more advanced see |winsaveview()| and |winrestview()|.

==
Renaming files *rename-files*

Say I have a directory with the following files in them (directory picked at
random :-):

```
buffer.c
charset.c
digraph.c
...
```

and I want to rename *.c *.bla. I'd do it like this: >

```
        $ vim
        :r !ls *.c
        :%s/\(.*\).c/mv & \1.bla
        :w !sh
        :q!
```

==
Change a name in multiple files *change-name*

Example for using a script file to change a name in several files:

        Create a file "subs.vim" containing substitute commands and a :update
        command: >
```
                :%s/Jones/Smith/g
                :%s/Allen/Peter/g
                :update
```
<
        Execute Vim on all files you want to change, and source the script for
        each argument: >

```
                vim *.let
                argdo source subs.vim
```

See |:argdo|.

==
Speeding up external commands *speed-up*

In some situations, execution of an external command can be very slow. This
can also slow down wildcard expansion on Unix. Here are a few suggestions to
increase the speed.

If your .cshrc (or other file, depending on the shell used) is very long, you
should separate it into a section for interactive use and a section for
non-interactive use (often called secondary shells). When you execute a
command from Vim like ":!ls", you do not need the interactive things (for
example, setting the prompt). Put the stuff that is not needed after these
lines: >

```
        if ($?prompt == 0) then
                exit 0
        endif
```

Another way is to include the "-f" flag in the 'shell' option, e.g.: >

```
        :set shell=csh\ -f
```

```
(the backslash is needed to include the space in the option).
This will make csh completely skip the use of the .cshrc file.  This may cause
some things to stop working though.

==============================================================================
Useful mappings                                              *useful-mappings*

Here are a few mappings that some people like to use.

                                                             *map-backtick*  >
        :map ' `
Make the single quote work like a backtick.  Puts the cursor on the column of
a mark, instead of going to the first non-blank character in the line.

                                                             *emacs-keys*
For Emacs-style editing on the command-line: >
        " start of line
        :cnoremap <C-A>         <Home>
        " back one character
        :cnoremap <C-B>         <Left>
        " delete character under cursor
        :cnoremap <C-D>         <Del>
        " end of line
        :cnoremap <C-E>         <End>
        " forward one character
        :cnoremap <C-F>         <Right>
        " recall newer command-line
        :cnoremap <C-N>         <Down>
        " recall previous (older) command-line
        :cnoremap <C-P>         <Up>
        " back one word
        :cnoremap <Esc><C-B>    <S-Left>
        " forward one word
        :cnoremap <Esc><C-F>    <S-Right>

NOTE: This requires that the '<' flag is excluded from 'cpoptions'. |<>|

                                                             *format-bullet-list*
This mapping will format any bullet list.  It requires that there is an empty
line above and below each list entry.  The expression commands are used to
be able to give comments to the parts of the mapping. >

        :let m =     ":map _f  :set ai<CR>"    " need 'autoindent' set
        :let m = m . "{O<Esc>"                 " add empty line above item
        :let m = m . "}{()^W"                  " move to text after bullet
        :let m = m . "i     <CR>     <Esc>"    " add space for indent
        :let m = m . "gq}"                     " format text after the bullet
        :let m = m . "{dd"                     " remove the empty line
        :let m = m . "5lDJ"                    " put text after bullet
        :execute m                             |" define the mapping

(<> notation |<>|.  Note that this is all typed literally.  ^W is "^" "W", not
CTRL-W.  You can copy/paste this into Vim if '<' is not included in
'cpoptions'.)

Note that the last comment starts with |", because the ":execute" command
doesn't accept a comment directly.

You also need to set 'textwidth' to a non-zero value, e.g., >
        :set tw=70

A mapping that does about the same, but takes the indent for the list from the
```

```
first line (Note: this mapping is a single long line with a lot of spaces): >
        :map _f :set ai<CR>}
{a
<Esc>WWmmkD`mi<CR><Esc>kkddpJgq}'mJO<Esc>j
<
                                                              *collapse*
These two mappings reduce a sequence of empty (;b) or blank (;n) lines into a
single line >
    :map ;b   GoZ<Esc>:g/^$/.,/./-j<CR>Gdd
    :map ;n   GoZ<Esc>:g/^[ <Tab>]*$/.,/[^ <Tab>]/-j<CR>Gdd

==============================================================================
Compressing the help files                              *gzip-helpfile*

For those of you who are really short on disk space, you can compress the help
files and still be able to view them with Vim.  This makes accessing the help
files a bit slower and requires the "gzip" program.

(1) Compress all the help files: "gzip doc/*.txt".

(2) Edit "doc/tags" and change the ".txt" to ".txt.gz": >
        :%s=\(\t.*\.txt\)\t=\1.gz\t=

(3) Add this line to your vimrc: >
        set helpfile={dirname}/help.txt.gz

Where {dirname} is the directory where the help files are.  The |gzip| plugin
will take care of decompressing the files.
You must make sure that $VIMRUNTIME is set to where the other Vim files are,
when they are not in the same location as the compressed "doc" directory.  See
|$VIMRUNTIME|.

==============================================================================
Executing shell commands in a window                    *shell-window*

There have been questions for the possibility to execute a shell in a window
inside Vim.  The answer: you can't!  Including this would add a lot of code to
Vim, which is a good reason not to do this.  After all, Vim is an editor, it
is not supposed to do non-editing tasks.  However, to get something like this,
you might try splitting your terminal screen or display window with the
"splitvt" program.  You can probably find it on some ftp server.  The person
that knows more about this is Sam Lantinga <slouken@cs.ucdavis.edu>.
An alternative is the "window" command, found on BSD Unix systems, which
supports multiple overlapped windows.  Or the "screen" program, found at
www.uni-erlangen.de, which supports a stack of windows.

==============================================================================
Hex editing                                     *hex-editing* *using-xxd*

See section |23.4| of the user manual.

If one has a particular extension that one uses for binary files (such as exe,
bin, etc), you may find it helpful to automate the process with the following
bit of autocmds for your <.vimrc>.  Change that "*.bin" to whatever
comma-separated list of extension(s) you find yourself wanting to edit: >

        " vim -b : edit binary using xxd-format!
        augroup Binary
          au!
          au BufReadPre  *.bin let &bin=1
          au BufReadPost *.bin if &bin | %!xxd
          au BufReadPost *.bin set ft=xxd | endif
```

```
          au BufWritePre *.bin if &bin | %!xxd -r
          au BufWritePre *.bin endif
          au BufWritePost *.bin if &bin | %!xxd
          au BufWritePost *.bin set nomod | endif
        augroup END

==============================================================================
Using <> notation in autocommands                       *autocmd-<>*

The <> notation is not recognized in the argument of an :autocmd.  To avoid
having to use special characters, you could use a self-destroying mapping to
get the <> notation and then call the mapping from the autocmd.  Example:

                                                    *map-self-destroy*  >
 " This is for automatically adding the name of the file to the menu list.
 " It uses a self-destroying mapping!
 " 1. use a line in the buffer to convert the 'dots' in the file name to \.
 " 2. store that in register '"'
 " 3. add that name to the Buffers menu list
 " WARNING: this does have some side effects, like overwriting the
 " current register contents and removing any mapping for the "i" command.
 "
 autocmd BufNewFile,BufReadPre * nmap i :nunmap i<CR>O<C-R>%<Esc>:.g/\./s/\./\\./
g<CR>0"9y$u:menu Buffers.<C-R>9 :buffer <C-R>%<C-V><CR><CR>
 autocmd BufNewFile,BufReadPre * normal i

Another method, perhaps better, is to use the ":execute" command.  In the
string you can use the <> notation by preceding it with a backslash.  Don't
forget to double the number of existing backslashes and put a backslash before
'"'.
>
  autocmd BufNewFile,BufReadPre * exe "normal O\<C-R>%\<Esc>:.g/\\./s/\\./\\\\./
g\<CR>0\"9y$u:menu Buffers.\<C-R>9 :buffer \<C-R>%\<C-V>\<CR>\<CR>"

For a real buffer menu, user functions should be used (see |:function|), but
then the <> notation isn't used, which defeats using it as an example here.

==============================================================================
Highlighting matching parens                                    *match-parens*

This example shows the use of a few advanced tricks:
- using the |CursorMoved| autocommand event
- using |searchpairpos()| to find a matching paren
- using |synID()| to detect whether the cursor is in a string or comment
- using |:match| to highlight something
- using a |pattern| to match a specific position in the file.

This should be put in a Vim script file, since it uses script-local variables.
It skips matches in strings or comments, unless the cursor started in string
or comment.  This requires syntax highlighting.

A slightly more advanced version is used in the |matchparen| plugin.
>
        let s:paren_hl_on = 0
        function s:Highlight_Matching_Paren()
          if s:paren_hl_on
            match none
            let s:paren_hl_on = 0
          endif

          let c_lnum = line('.')
          let c_col = col('.')
```

```
          let c = getline(c_lnum)[c_col - 1]
          let plist = split(&matchpairs, ':\|,')
          let i = index(plist, c)
          if i < 0
            return
          endif
          if i % 2 == 0
            let s_flags = 'nW'
            let c2 = plist[i + 1]
          else
            let s_flags = 'nbW'
            let c2 = c
            let c = plist[i - 1]
          endif
          if c == '['
            let c = '\['
            let c2 = '\]'
          endif
          let s_skip ='synIDattr(synID(line("."), col("."), 0), "name") ' .
                \ '=~?  "string\\|comment"'
          execute 'if' s_skip '| let s_skip = 0 | endif'

          let [m_lnum, m_col] = searchpairpos(c, '', c2, s_flags, s_skip)

          if m_lnum > 0 && m_lnum >= line('w0') && m_lnum <= line('w$')
            exe 'match Search /\(\%' . c_lnum . 'l\%' . c_col .
                  \ 'c\)\|\(\%' . m_lnum . 'l\%' . m_col . 'c\)/'
            let s:paren_hl_on = 1
          endif
        endfunction

        autocmd CursorMoved,CursorMovedI * call s:Highlight_Matching_Paren()
        autocmd InsertEnter * match none
<

 vim:tw=78:ts=8:ft=help:norl:
*message.txt*   For Vim version 8.0.  Last change: 2017 Mar 25


                  VIM REFERENCE MANUAL    by Bram Moolenaar


This file contains an alphabetical list of messages and error messages that
Vim produces.  You can use this if you don't understand what the message
means.  It is not complete though.

1. Old messages         |:messages|
2. Error messages       |error-messages|
3. Messages             |messages|

==============================================================================
1. Old messages                 *:messages* *:mes* *message-history*

The ":messages" command can be used to view previously given messages.  This
is especially useful when messages have been overwritten or truncated.  This
depends on the 'shortmess' option.

        :messages               Show all messages.

        :{count}messages        Show the {count} most recent messages.
```

```
        :messages clear         Clear all messages.

        :{count}messages clear  Clear messages, keeping only the {count} most
                                recent ones.

The number of remembered messages is fixed at 20 for the tiny version and 200
for other versions.

                                                                      *g<*
The "g<" command can be used to see the last page of previous command output.
This is especially useful if you accidentally typed <Space> at the hit-enter
prompt.  You are then back at the hit-enter prompt and can then scroll further
back.
Note: If the output has been stopped with "q" at the more prompt, it will only
be displayed up to this point.
The previous command output is cleared when another command produces output.
The "g<" output is not redirected.

If you are using translated messages, the first printed line tells who
maintains the messages or the translations.  You can use this to contact the
maintainer when you spot a mistake.

If you want to find help on a specific (error) message, use the ID at the
start of the message.  For example, to get help on the message: >

        E72: Close error on swap file

or (translated): >

        E72: Errore durante chiusura swap file

Use: >

        :help E72

If you are lazy, it also works without the shift key: >

        :help e72

==============================================================================
2. Error messages                                   *error-messages* *errors*

When an error message is displayed, but it is removed before you could read
it, you can see it again with: >
  :echo errmsg
Or view a list of recent messages with: >
  :messages
See `:messages` above.


LIST OF MESSAGES
                        *E222* *E228* *E232* *E256* *E293* *E298* *E304* *E317*
                        *E318* *E356* *E438* *E439* *E440* *E316* *E320* *E322*
                        *E323* *E341* *E473* *E570* *E685*  >
  Add to read buffer
  makemap: Illegal mode
  Cannot create BalloonEval with both message and callback
  Hangul automata ERROR
  block was not locked
  Didn't get block nr {N}?
  ml_upd_block0(): Didn't get block 0??
  pointer block id wrong {N}
```

  Updated too many blocks?
  get_varp ERROR
  u_undo: line numbers wrong
  undo list corrupt
  undo line missing
  ml_get: cannot find line {N}
  cannot find line {N}
  line number out of range: {N} past the end
  line count wrong in block {N}
  Internal error
  Internal error: {function}
  fatal error in cs_manage_matches

This is an internal error. If you can reproduce it, please send in a bug
report. |bugs|

>
  ATTENTION
  Found a swap file by the name ...

See |ATTENTION|.

E92 >
  Buffer {N} not found

The buffer you requested does not exist. This can also happen when you have
wiped out a buffer which contains a mark or is referenced in another way.
|:bwipeout|

E95 >
  Buffer with this name already exists

You cannot have two buffers with the same name.

E72 >
  Close error on swap file

The |swap-file|, that is used to keep a copy of the edited text, could not be
closed properly. Mostly harmless.

E169 >
  Command too recursive

This happens when an Ex command executes an Ex command that executes an Ex
command, etc. The limit is 200 or the value of 'maxfuncdepth', whatever is
larger. When it's more there probably is an endless loop. Probably a
|:execute| or |:source| command is involved.

E254 >
  Cannot allocate color {name}

The color name {name} is unknown. See |gui-colors| for a list of colors that
are available on most systems.

E458 >
  Cannot allocate colormap entry, some colors may be incorrect

This means that there are not enough colors available for Vim. It will still
run, but some of the colors will not appear in the specified color. Try
stopping other applications that use many colors, or start them after starting
gvim.
Browsers are known to consume a lot of colors. You can avoid this with

```
netscape by telling it to use its own colormap: >
        netscape -install
Or tell it to limit to a certain number of colors (64 should work well): >
        netscape -ncols 64
This can also be done with a line in your Xdefaults file: >
        Netscape*installColormap: Yes
or >
        Netscape*maxImageColors:  64
<
                                                              *E79*  >
  Cannot expand wildcards

A filename contains a strange combination of characters, which causes Vim to
attempt expanding wildcards but this fails.  This does NOT mean that no
matching file names could be found, but that the pattern was illegal.

                                                              *E459*  >
  Cannot go back to previous directory

While expanding a file name, Vim failed to go back to the previously used
directory.  All file names being used may be invalid now!  You need to have
execute permission on the current directory.

                                                              *E190* *E212*  >
  Cannot open "{filename}" for writing
  Can't open file for writing

For some reason the file you are writing to cannot be created or overwritten.
The reason could be that you do not have permission to write in the directory
or the file name is not valid.

                                                              *E166*  >
  Can't open linked file for writing

You are trying to write to a file which can't be overwritten, and the file is
a link (either a hard link or a symbolic link).  Writing might still be
possible if the directory that contains the link or the file is writable, but
Vim now doesn't know if you want to delete the link and write the file in its
place, or if you want to delete the file itself and write the new file in its
place.  If you really want to write the file under this name, you have to
manually delete the link or the file, or change the permissions so that Vim
can overwrite.

                                                              *E46*  >
  Cannot change read-only variable "{name}"

You are trying to assign a value to an argument of a function |a:var| or a Vim
internal variable |v:var| which is read-only.

                                                              *E90*  >
  Cannot unload last buffer

Vim always requires one buffer to be loaded, otherwise there would be nothing
to display in the window.

                                                              *E40*  >
  Can't open errorfile <filename>

When using the ":make" or ":grep" commands: The file used to save the error
messages or grep output cannot be opened.  This can have several causes:
- 'shellredir' has a wrong value.
- The shell changes directory, causing the error file to be written in another
```

```
  directory.  This could be fixed by changing 'makeef', but then the make
  command is still executed in the wrong directory.
- 'makeef' has a wrong value.
- The 'grepprg' or 'makeprg' could not be executed.  This cannot always be
  detected (especially on MS-Windows).  Check your $PATH.

 >
  Can't open file C:\TEMP\VIoD243.TMP

On MS-Windows, this message appears when the output of an external command was
to be read, but the command didn't run successfully.  This can be caused by
many things.  Check the 'shell', 'shellquote', 'shellxquote', 'shellslash' and
related options.  It might also be that the external command was not found,
there is no different error message for that.

                                                              *E12*  >
  Command not allowed from exrc/vimrc in current dir or tag search

Some commands are not allowed for security reasons.  These commands mostly
come from a .exrc or .vimrc file in the current directory, or from a tags
file.  Also see 'secure'.

                                                              *E74*  >
  Command too complex

A mapping resulted in a very long command string.  Could be caused by a
mapping that indirectly calls itself.

>
  CONVERSION ERROR

When writing a file and the text "CONVERSION ERROR" appears, this means that
some bits were lost when converting text from the internally used UTF-8 to the
format of the file.  The file will not be marked unmodified.  If you care
about the loss of information, set the 'fileencoding' option to another value
that can handle the characters in the buffer and write again.  If you don't
care, you can abandon the buffer or reset the 'modified' option.

                                                              *E302*  >
  Could not rename swap file

When the file name changes, Vim tries to rename the |swap-file| as well.
This failed and the old swap file is now still used.  Mostly harmless.

                                                              *E43* *E44*  >
  Damaged match string
  Corrupted regexp program

Something inside Vim went wrong and resulted in a corrupted regexp.  If you
know how to reproduce this problem, please report it. |bugs|

                                                              *E208* *E209* *E210*  >
  Error writing to "{filename}"
  Error closing "{filename}"
  Error reading "{filename}"

This occurs when Vim is trying to rename a file, but a simple change of file
name doesn't work.  Then the file will be copied, but somehow this failed.
The result may be that both the original file and the destination file exist
and the destination file may be incomplete.

>
```

  Vim: Error reading input, exiting...

This occurs when Vim cannot read typed characters while input is required.
Vim got stuck, the only thing it can do is exit. This can happen when both
stdin and stderr are redirected and executing a script that doesn't exit Vim.

                                                              E47 >
  Error while reading errorfile

Reading the error file was not possible. This is NOT caused by an error
message that was not recognized.

                                                              E80 >
  Error while writing

Writing a file was not completed successfully. The file is probably
incomplete.

                                                              E13 *E189* >
  File exists (add ! to override)
  "{filename}" exists (add ! to override)

You are protected from accidentally overwriting a file. When you want to
write anyway, use the same command, but add a "!" just after the command.
Example: >
        :w /tmp/test
changes to: >
        :w! /tmp/test
<
                                                              E768 >
  Swap file exists: {filename} (:silent! overrides)

You are protected from overwriting a file that is being edited by Vim. This
happens when you use ":w! filename" and a swapfile is found.
- If the swapfile was left over from an old crashed edit session you may want
  to delete the swapfile. Edit {filename} to find out information about the
  swapfile.
- If you want to write anyway prepend ":silent!" to the command. For example: >
        :silent! w! /tmp/test
< The special command is needed, since you already added the ! for overwriting
  an existing file.

                                                              E139 >
  File is loaded in another buffer

You are trying to write a file under a name which is also used in another
buffer. This would result in two versions of the same file.

                                                              E142 >
  File not written: Writing is disabled by 'write' option

The 'write' option is off. This makes all commands that try to write a file
generate this message. This could be caused by a |-m| commandline argument.
You can switch the 'write' option on with ":set write".

                                                              E25 >
  GUI cannot be used: Not enabled at compile time

You are running a version of Vim that doesn't include the GUI code. Therefore
"gvim" and ":gui" don't work.

                                                              E49 >

  Invalid scroll size

This is caused by setting an invalid value for the 'scroll', 'scrolljump' or
'scrolloff' options.

                                                                E17 >
  "{filename}" is a directory

You tried to write a file with the name of a directory. This is not possible.
You probably need to append a file name.

                                                                E19 >
  Mark has invalid line number

You are using a mark that has a line number that doesn't exist. This can
happen when you have a mark in another file, and some other program has
deleted lines from it.

                                                                E219 *E220* >
  Missing {.
  Missing }.

Using a {} construct in a file name, but there is a { without a matching } or
the other way around. It should be used like this: {foo,bar}. This matches
"foo" and "bar".

                                                                E315 >
  ml_get: invalid lnum: {number}

This is an internal Vim error. Please try to find out how it can be
reproduced, and submit a bug report |bugreport.vim|.

                                                                E173 >
  {number} more files to edit

You are trying to exit, while the last item in the argument list has not been
edited. This protects you from accidentally exiting when you still have more
files to work on. See |argument-list|. If you do want to exit, just do it
again and it will work.

                                                                E23 *E194* >
  No alternate file
  No alternate file name to substitute for '#'

The alternate file is not defined yet. See |alternate-file|.

                                                                E32 >
  No file name

The current buffer has no name. To write it, use ":w fname". Or give the
buffer a name with ":file fname".

                                                                E141 >
  No file name for buffer {number}

One of the buffers that was changed does not have a file name. Therefore it
cannot be written. You need to give the buffer a file name: >
        :buffer {number}
        :file {filename}
<
                                                                E33 >
  No previous substitute regular expression

When using the '~' character in a pattern, it is replaced with the previously
used pattern in a ":substitute" command. This fails when no such command has
been used yet. See |/~|. This also happens when using ":s/pat/%/", where the
"%" stands for the previous substitute string.

E35 >
  No previous regular expression

When using an empty search pattern, the previous search pattern is used. But
that is not possible if there was no previous search.

E24 >
  No such abbreviation

You have used an ":unabbreviate" command with an argument which is not an
existing abbreviation. All variations of this command give the same message:
":cunabbrev", ":iunabbrev", etc. Check for trailing white space.

>
  /dev/dsp: No such file or directory

Only given for GTK GUI with Gnome support. Gnome tries to use the audio
device and it isn't present. You can ignore this error.

E31 >
  No such mapping

You have used an ":unmap" command with an argument which is not an existing
mapping. All variations of this command give the same message: ":cunmap",
":unmap!", etc. A few hints:
- Check for trailing white space.
- If the mapping is buffer-local you need to use ":unmap <buffer>".
  |:map-<buffer>|

E37 *E89* >
  No write since last change (add ! to override)
  No write since last change for buffer {N} (add ! to override)

You are trying to |abandon| a file that has changes. Vim protects you from
losing your work. You can either write the changed file with ":w", or, if you
are sure, |abandon| it anyway, and lose all the changes. This can be done by
adding a '!' character just after the command you used. Example: >
        :e other_file
changes to: >
        :e! other_file
<
E162 >
  No write since last change for buffer "{name}"

This appears when you try to exit Vim while some buffers are changed. You
will either have to write the changed buffer (with |:w|), or use a command to
abandon the buffer forcefully, e.g., with ":qa!". Careful, make sure you
don't throw away changes you really want to keep. You might have forgotten
about a buffer, especially when 'hidden' is set.

>
  [No write since last change]

This appears when executing a shell command while at least one buffer was
changed. To avoid the message reset the 'warn' option.

E38 >
  Null argument

Something inside Vim went wrong and resulted in a NULL pointer. If you know
how to reproduce this problem, please report it. |bugs|

E41 *E82* *E83* *E342* >
  Out of memory!
  Out of memory! (allocating {number} bytes)
  Cannot allocate any buffer, exiting...
  Cannot allocate buffer, using other one...

Oh, oh. You must have been doing something complicated, or some other program
is consuming your memory. Be careful! Vim is not completely prepared for an
out-of-memory situation. First make sure that any changes are saved. Then
try to solve the memory shortage. To stay on the safe side, exit Vim and
start again.

Buffers are only partly kept in memory, thus editing a very large file is
unlikely to cause an out-of-memory situation. Undo information is completely
in memory, you can reduce that with these options:
- 'undolevels' Set to a low value, or to -1 to disable undo completely. This
  helps for a change that affects all lines.
- 'undoreload' Set to zero to disable.

E339 >
  Pattern too long

This happens on systems with 16 bit ints: The compiled regexp pattern is
longer than about 65000 characters. Try using a shorter pattern.
It also happens when the offset of a rule doesn't fit in the space available.
Try simplifying the pattern.

E45 >
  'readonly' option is set (add ! to override)

You are trying to write a file that was marked as read-only. To write the
file anyway, either reset the 'readonly' option, or add a '!' character just
after the command you used. Example: >
        :w
changes to: >
        :w!
<
E294 *E295* *E301* >
  Read error in swap file
  Seek error in swap file read
  Oops, lost the swap file!!!

Vim tried to read text from the |swap-file|, but something went wrong. The
text in the related buffer may now be corrupted! Check carefully before you
write a buffer. You may want to write it in another file and check for
differences.

E192 >
  Recursive use of :normal too deep

You are using a ":normal" command, whose argument again uses a ":normal"
command in a recursive way. This is restricted to 'maxmapdepth' levels. This
example illustrates how to get this message: >
        :map gq :normal gq<CR>
If you type "gq", it will execute this mapping, which will call "gq" again.

E22 >
  Scripts nested too deep

Scripts can be read with the "-s" command-line argument and with the ":source"
command. The script can then again read another script. This can continue
for about 14 levels. When more nesting is done, Vim assumes that there is a
recursive loop somewhere and stops with this error message.

E319 >
  Sorry, the command is not available in this version

You have used a command that is not present in the version of Vim you are
using. When compiling Vim, many different features can be enabled or
disabled. This depends on how big Vim has chosen to be and the operating
system. See |+feature-list| for when which feature is available. The
|:version| command shows which feature Vim was compiled with.

E300 >
  Swap file already exists (symlink attack?)

This message appears when Vim is trying to open a swap file and finds it
already exists or finds a symbolic link in its place. This shouldn't happen,
because Vim already checked that the file doesn't exist. Either someone else
opened the same file at exactly the same moment (very unlikely) or someone is
attempting a symlink attack (could happen when editing a file in /tmp or when
'directory' starts with "/tmp", which is a bad choice).

E432 >
  Tags file not sorted: {file name}

Vim (and Vi) expect tags files to be sorted in ASCII order. Binary searching
can then be used, which is a lot faster than a linear search. If your tags
files are not properly sorted, reset the |'tagbsearch'| option.
This message is only given when Vim detects a problem when searching for a
tag. Sometimes this message is not given, even though the tags file is not
properly sorted.

E460 >
  The resource fork would be lost (add ! to override)

On the Macintosh (classic), when writing a file, Vim attempts to preserve all
info about a file, including its resource fork. If this is not possible you
get this error message. Append "!" to the command name to write anyway (and
lose the info).

E424 >
  Too many different highlighting attributes in use

Vim can only handle about 223 different kinds of highlighting. If you run
into this limit, you have used too many |:highlight| commands with different
arguments. A ":highlight link" is not counted.

E77 >
  Too many file names

When expanding file names, more than one match was found. Only one match is
allowed for the command that was used.

E303 >
  Unable to open swap file for "{filename}", recovery impossible

Vim was not able to create a swap file. You can still edit the file, but if

```
Vim unexpectedly exits the changes will be lost.  And Vim may consume a lot of
memory when editing a big file.  You may want to change the 'directory' option
to avoid this error.  See |swap-file|.

                                                              *E140*  >
  Use ! to write partial buffer

When using a range to write part of a buffer, it is unusual to overwrite the
original file.  It is probably a mistake (e.g., when Visual mode was active
when using ":w"), therefore Vim requires using a !  after the command, e.g.:
":3,10w!".
>

  Warning: Cannot convert string "<Key>Escape,_Key_Cancel" to type
  VirtualBinding

Messages like this appear when starting up.  This is not a Vim problem, your
X11 configuration is wrong.  You can find a hint on how to solve this here:
http://groups.yahoo.com/group/solarisonintel/message/12179.
[this URL is no longer valid]

                                                              *W10*  >
  Warning: Changing a readonly file

The file is read-only and you are making a change to it anyway.  You can use
the |FileChangedRO| autocommand event to avoid this message (the autocommand
must reset the 'readonly' option).  See 'modifiable' to completely disallow
making changes to a file.
This message is only given for the first change after 'readonly' has been set.

                                                              *W13*  >
  Warning: File "{filename}" has been created after editing started

You are editing a file in Vim when it didn't exist, but it does exist now.
You will have to decide if you want to keep the version in Vim or the newly
created file.  This message is not given when 'buftype' is not empty.

                                                              *W11*  >
  Warning: File "{filename}" has changed since editing started

The file which you have started editing has got another timestamp and the
contents changed (more precisely: When reading the file again with the current
option settings and autocommands you would end up with different text).  This
probably means that some other program changed the file.  You will have to
find out what happened, and decide which version of the file you want to keep.
Set the 'autoread' option if you want to do this automatically.
This message is not given when 'buftype' is not empty.

There is one situation where you get this message even though there is nothing
wrong: If you save a file in Windows on the day the daylight saving time
starts.  It can be fixed in one of these ways:
- Add this line in your autoexec.bat: >
            SET TZ=-1
< Adjust the "-1" for your time zone.
- Disable "automatically adjust clock for daylight saving changes".
- Just write the file again the next day.  Or set your clock to the next day,
  write the file twice and set the clock back.

                                                              *W12*  >
  Warning: File "{filename}" has changed and the buffer was changed in Vim as well

Like the above, and the buffer for the file was changed in this Vim as well.
```

You will have to decide if you want to keep the version in this Vim or the one
on disk. This message is not given when 'buftype' is not empty.

W16 >
  Warning: Mode of file "{filename}" has changed since editing started

When the timestamp for a buffer was changed and the contents are still the
same but the mode (permissions) have changed. This usually occurs when
checking out a file from a version control system, which causes the read-only
bit to be reset. It should be safe to reload the file. Set 'autoread' to
automatically reload the file.

E211 >
  File "{filename}" no longer available

The file which you have started editing has disappeared, or is no longer
accessible. Make sure you write the buffer somewhere to avoid losing
changes. This message is not given when 'buftype' is not empty.

W14 >
  Warning: List of file names overflow

You must be using an awful lot of buffers. It's now possible that two buffers
have the same number, which causes various problems. You might want to exit
Vim and restart it.

E931 >
  Buffer cannot be registered

Out of memory or a duplicate buffer number. May happen after W14. Looking up
a buffer will not always work, better restart Vim.

E296 *E297* >
  Seek error in swap file write
  Write error in swap file

This mostly happens when the disk is full. Vim could not write text into the
|swap-file|. It's not directly harmful, but when Vim unexpectedly exits some
text may be lost without recovery being possible. Vim might run out of memory
when this problem persists.

connection-refused >
  Xlib: connection to "<machine-name:0.0" refused by server

This happens when Vim tries to connect to the X server, but the X server does
not allow a connection. The connection to the X server is needed to be able
to restore the title and for the xterm clipboard support. Unfortunately this
error message cannot be avoided, except by disabling the |+xterm_clipboard|
and |+X11| features.

E10 >
  \\ should be followed by /, ? or &

A command line started with a backslash or the range of a command contained a
backslash in a wrong place. This is often caused by command-line continuation
being disabled. Remove the 'C' flag from the 'cpoptions' option to enable it.
Or use ":set nocp".

E471 >
  Argument required

This happens when an Ex command with mandatory argument(s) was executed, but

no argument has been specified.

E474 *E475* >
  Invalid argument
  Invalid argument: {arg}

An Ex command has been executed, but an invalid argument has been specified.

E488 >
  Trailing characters

An argument has been added to an Ex command that does not permit one.

E477 *E478* >
  No ! allowed
  Don't panic!

You have added a "!" after an Ex command that doesn't permit one.

E481 >
  No range allowed

A range was specified for an Ex command that doesn't permit one. See
|cmdline-ranges|.

E482 *E483* >
  Can't create file {filename}
  Can't get temp file name

Vim cannot create a temporary file.

E484 *E485* >
  Can't open file {filename}
  Can't read file {filename}

Vim cannot read a temporary file. Especially on Windows, this can be caused
by wrong escaping of special characters for cmd.exe; the approach was
changed with patch 7.3.443. Try using |shellescape()| for all shell arguments
given to |system()|, or explicitly add escaping with ^. Also see
'shellxquote' and 'shellxescape'.

E464 >
  Ambiguous use of user-defined command

There are two user-defined commands with a common name prefix, and you used
Command-line completion to execute one of them. |user-cmd-ambiguous|
Example: >
        :command MyCommand1 echo "one"
        :command MyCommand2 echo "two"
        :MyCommand
<
E492 >
  Not an editor command

You tried to execute a command that is neither an Ex command nor
a user-defined command.

E943 >
  Command table needs to be updated, run 'make cmdidxs'

This can only happen when changing the source code, when adding a command in
src/ex_cmds.h. The lookup table then needs to be updated, by running: >

```
        make cmdidxs

==============================================================================
3. Messages                                             *messages*

This is an (incomplete) overview of various messages that Vim gives:

                        *hit-enter* *press-enter* *hit-return*
                        *press-return* *hit-enter-prompt*

  Press ENTER or type command to continue

This message is given when there is something on the screen for you to read,
and the screen is about to be redrawn:
- After executing an external command (e.g., ":!ls" and "=").
- Something is displayed on the status line that is longer than the width of
  the window, or runs into the 'showcmd' or 'ruler' output.

-> Press <Enter> or <Space> to redraw the screen and continue, without that
   key being used otherwise.
-> Press ':' or any other Normal mode command character to start that command.
-> Press 'k', <Up>, 'u', 'b' or 'g' to scroll back in the messages.  This
   works the same way as at the |more-prompt|.  Only works when 'compatible'
   is off and 'more' is on.
-> Pressing 'j', 'f', 'd' or <Down> is ignored when messages scrolled off the
   top of the screen, 'compatible' is off and 'more' is on, to avoid that
   typing one 'j' or 'f' too many causes the messages to disappear.
-> Press <C-Y> to copy (yank) a modeless selection to the clipboard register.
-> Use a menu.  The characters defined for Cmdline-mode are used.
-> When 'mouse' contains the 'r' flag, clicking the left mouse button works
   like pressing <Space>.  This makes it impossible to select text though.
-> For the GUI clicking the left mouse button in the last line works like
   pressing <Space>.
{Vi: only ":" commands are interpreted}

If you accidentally hit <Enter> or <Space> and you want to see the displayed
text then use |g<|.  This only works when 'more' is set.

To reduce the number of hit-enter prompts:
- Set 'cmdheight' to 2 or higher.
- Add flags to 'shortmess'.
- Reset 'showcmd' and/or 'ruler'.

If your script causes the hit-enter prompt and you don't know why, you may
find the |v:scrollstart| variable useful.

Also see 'mouse'.  The hit-enter message is highlighted with the |hl-Question|
group.


                                                *more-prompt* *pager*  >
  -- More --
  -- More -- SPACE/d/j: screen/page/line down, b/u/k: up, q: quit

This message is given when the screen is filled with messages.  It is only
given when the 'more' option is on.  It is highlighted with the |hl-MoreMsg|
group.

Type                                    effect ~
     <CR> or <NL> or j or <Down>        one more line
     d                                  down a page (half a screen)
     <Space> or f or <PageDown>         down a screen
```

```
     G                                 down all the way, until the hit-enter
                                       prompt

     <BS> or k or <Up>                 one line back (*)
     u                                 up a page (half a screen) (*)
     b or <PageUp>                     back a screen (*)
     g                                 back to the start (*)

     q, <Esc> or CTRL-C                stop the listing
     :                                 stop the listing and enter a
                                            command-line
    <C-Y>                              yank (copy) a modeless selection to
                                       the clipboard ("* and "+ registers)
    {menu-entry}                       what the menu is defined to in
                                       Cmdline-mode.
    <LeftMouse> (**)                   next page

Any other key causes the meaning of the keys to be displayed.

(*)  backwards scrolling is {not in Vi}.  Only scrolls back to where messages
     started to scroll.
(**) Clicking the left mouse button only works:
     - For the GUI: in the last line of the screen.
     - When 'r' is included in 'mouse' (but then selecting text won't work).


Note: The typed key is directly obtained from the terminal, it is not mapped
and typeahead is ignored.

The |g<| command can be used to see the last page of previous command output.
This is especially useful if you accidentally typed <Space> at the hit-enter
prompt.

 vim:tw=78:ts=8:ft=help:norl:
*quotes.txt*    For Vim version 8.0.  Last change: 2010 Nov 03


                  VIM REFERENCE MANUAL    by Bram Moolenaar


                                                        *quotes*
Here are some nice quotes about Vim that I collected from news and mail.


vim (vim) noun - Ebullient vitality and energy.  [Latin, accusative of vis,
strength]  (Dictionary)

Vim is so much better than vi that a great many of my old vi :map's became
immediately obsolete! (Tony Nugent, Australia)

Coming with a very GUI mindset from Windows, I always thought of people using
Vi as some kind of outer space alien in human clothes.  Once I tried I really
got addicted by its power and now I found myself typing Vim keypresses in the
oddest places! That's why I would like to see Vim embedded in every
application which deals with text editing.  (Jos\E9 Fonseca)

I was a 12-year emacs user who switched to Vim about a year ago after finally
giving up on the multiple incompatible versions, flaky contributed packages,
disorganized keystrokes, etc.  And it was one of the best moves I ever made.
(Joel Burton)

Although all of the programs were used during the preparation of the new and
```

revised material, most of the editing was done with Vim versions 4.5 and 5.0 under GNU-Linux (Redhat 4.2). (Arnold Robbins, Israel, author of "Learning the Vi editor")

Out of all the open software i've ever seen and used, and i've seen a lot, Vim is the best, most useful and highest quality to work with, second only to the linux kernel itself. (Peter Jay Salzman)

It's well worth noting that the _entirety_ of SourceForge was written using Vim and its nifty PHP syntax highlighting. I think the entire SF.net tech staff uses Vim and we're all excited to have you aboard! (Tim Perdue)

Vim is one of a select bunch of tools for which I have no substitute. It is a brilliant piece of work! (Biju Chacko)

A previous girlfriend of mine switched to emacs. Needless to say, the relationship went nowhere. (Geoffrey Mann)

I rarely think about Vim, in the same way that I guess a fish rarely thinks about water. It's the environment in which everything else happens. I'm a fairly busy system administrator working on a lot of different platforms. Vim is the only thing that's consistent across all my systems, and it's just about the only thing that doesn't break from time to time. When a new system comes in the door without Vim, I install it right away. Great to have a tool that's the same everywhere, that's completely reliable, so I can ignore it and think about other things. (Pete Schaeffer)

Having recently succeeded in running Vim via telnet through a Nokia Communicator, I can now report that it works nicely on a Palm Pilot too. (Allan Kelly, Scotland)

You've done a tremendous job with 'VIM', Bram! The more I use it, the more impressed I get (I am an old 'vi' die hard who once started out with early versions of 'emacs' in the late 1970's and was relieved by finding 'vi' in the first UNIX I came across in 1983). In my opinion, it's about time 'VIM' replace 'emacs' as the standard for top editors. (Bo Thide', Sweden)

I love and use Vim heavily too. (Larry Wall)

Vi is like a Ferrari, if you're a beginner, it handles like a bitch, but once you get the hang of it, it's small, powerful and FAST! (Unknown)
Vim is like a new model Ferrari, and sounds like one too - "VIIIIIIMMM!" (Stephen Riehm, Germany)

Schon bei Nutzung eines Bruchteils der Vim-Funktionen wird der Benutzer recht schnell die Vorzuege dieses Editors kennen- und schaetzenlernen.
Translated: Even when only using a fraction of Vim-functions, the user will quickly get used to and appreciate the advantages of this editor. (Garry Glendown, conclusion of an article on Vim in iX magazine 9/1998)

I've recently acquired the O'Reilly book on Vi (it also discusses Vim in-depth), and I'm amazed at just how powerful this application is. (Jeffrey Rankin)

This guide was written using the Windows 9.x distribution of gvim, which is quite possibly the greatest thing to come along since God created the naked girl. (Michael DiBernardo)

Boy, I thought I knew almost everything about Vim, but every time I browse the online documentation, I hit upon a minor but cool aspect of a Vim feature that I didn't know before! I must say the documentation is one the finest I've ever seen in a product -- even better than most commercial products.

(Gautam Mudunuri)

Vim 4.5 is really a fantastic editor. It has sooooo many features and more importantly, the defaults are so well thought out that you really don't have to change anything!! Words cannot express my amazement and gratitude to the creators of Vim. Keep it up. (Vikas, USA)

I wonder how long it will be before people will refer to other Vi editors as Vim clones? (Darren Hiebert)

I read about [auto-positioning-in-file-based-on-the-errors-from-make] in one of those "Perfect Programmer's Editor" threads and was delighted to discover that Vim already supports it. (Brendan Macmillan, Australia)

I just discovered Vim (5.0) and I'm telling everyone I know about it!
I tell them Vim stands for Vi for the new (M)illenium. Thanks so much!
(Matt F. Valentine)

I think from now on "vi" should be called "Vim Imitation", not the other way around. (Rungun Ramanathan)

The Law of Vim:
For each member b of the possible behaviour space B of program P, there exists a finite time t before which at least one user u in the total user space U of program P will request b becomes a member of the allowed behaviour space B' (B' <= B).
In other words: Sooner or later everyone wants everything as an option.
(Negri)

Whenever I move to a new computing platform, the first thing I do is to port Vim. Lately, I am simply stunned by its ease of compilation using the configure facility. (A.M. Sabuncu, Turkey)

The options are really excellent and very powerful. (Anish Maharaj)

The Spring user-interface designs are in, and word from the boutiques is that 80x24 text-only mode is back with a *vengeance! Vi editor clone Vim burst onto March desk-tops with a dazzling show of pastel syntax highlights for its 5.0 look. Strident and customizable, Vim raises eyebrows with its interpretation of the classic Vi single-key macro collection.
http://www.ntk.net/index.cgi?back=archive98/now0327.txt&line=179#l

I just wanted to take this opportunity to let you know that Vim 5 ROCKS!
Syntax highlighting: how did I survive without it?! Thank you for creating mankind's best editor! (Mun Johl, USA)

Thanks again for Vim. I use it every day on Linux. (Eric Foster-Johnson, author of the book "UNIX Programming Tools")

The BEST EDITOR EVER (Stuart Woolford)

I have used most of Vim's fancy features at least once, many frequently, and I can honestly say that I couldn't live with anything less anymore. My productivity has easily doubled compared to what it was when I used vi.
(Sitaram Chamarty)

I luv Vim. It is incredible. I'm naming my first-born Vimberly. (Jose Unpingco, USA)

Hint: "Vim" is "vi improved" - much better! (Sven Guckes, Germany)

I use Vim every day. I spend more time in Vim than in any other program...

It's the best vi clone there is. I think it's great. (Craig Sanders, Australia)

I strongly advise using Vim--its infinite undo/redo saved me much grief. (Terry Brown)

Thanks very much for writing what in my opinion is the finest text editor on the planet. If I were to get another cat, I would name it "Vim". (Bob Sheehan, USA)

I typed :set all and the screen FILLED up with options. A whole screen of things to be set and unset. I saw some of my old friends like wrapmargin, modelines and showmode, but the screen was FILLED with new friends! I love them all! I love Vim! I'm so happy that I've found this editor! I feel like how I once felt when I started using vi after a couple of years of using ed. I never thought I'd forsake my beloved ed, but vi ... oh god, vi was great. And now, Vim. (Peter Jay Salzman, USA)

I am really happy with such a wonderful software package. Much better than almost any expensive, off the shelf program. (Jeff Walker)

Whenever I reread the Vim documentation I'm overcome with excitement at the power of the editor. (William Edward Webber, Australia)

Hurrah for Vim!! It is "at your fingertips" like vi, and has the extensions that vi sorely needs: highlighting for executing commands on blocks, an easily navigable and digestible help screen, and more. (Paul Pax)

The reason WHY I don't have this amazingly useful macro anymore, is that I now use Vim - and this is built in!! (Stephen Riehm, Germany)

I am a user of Vim and I love it. I use it to do all my programming, C, C++, HTML what ever. (Tim Allwine)

I discovered Vim after years of struggling with the original vi, and I just can't live without it anymore. (Emmanuel Mogenet, USA)

Emacs has not a bit of chance to survive so long as Vim is around. Besides, it also has the most detailed software documentation I have ever seen---much better than most commercial software! (Leiming Qian)

This version of Vim will just blow people apart when they discover just how fantastic it is! (Tony Nugent, Australia)

I took your advice & finally got Vim & I'm really impressed. Instant convert. (Patrick Killelea, USA)

Vim is by far my favorite piece of shareware and I have been particularly pleased with version 3.0. This is really a solid piece of work. (Robert Colon, USA)

Vim is a joy to use, it is so well thought and practical that I wonder why anybody would use visual development tools. Vim is powerful and elegant, it looks deceptively simple but is almost as complex as a 747 (especially when I look at my growing .vimrc), keep up that wonderful job, Vim is a centerpiece of the free software world. (Louis-David Mitterand, USA)

I cannot believe how great it is to use Vim. I think the guys at work are getting tired of hearing me bragging about it. Others eyes are lighting up. (Rick Croote)

Emacs takes way too much time to start up and run, it is too big and bulky for

effective use and the interface is more confusing than it is of any help. Vim however is short, it is fast, it is powerful, it has a good interface and it is all purpose. (Paal Ditlefsen Ekran)

From the first time I got Vim3.0, I was very enthusiastic. It has almost no problems. The swapfile handling and the backup possibilities are robust, also the protection against editing one file twice. It is very compatible to the real VI (and that is a MUST, because my brain is trained over years in using it). (Gert van Antwerpen, Holland)

Visual mode in Vim is a very powerful thing! (Tony Nugent, Australia)

I have to say that Vim is =THE= single greatest piece of source code to ever come across the net (Jim Battle, USA).

In fact, if you do want to get a new vi I'd suggest Vim-3.0. This is, by far, the best version of vi I've ever seen (Albert W. Schueller).

I should mention that Vim is a very good editor and can compete with anything (Ilya Beloozerov).

To tell the truth sometimes I used elvis, vile, xvi, calvin, etc. And this is the reason that I can state that Vim is the best! (Ferenc Deak, Hungary)

Vim is by far the best editor that I have used in a long time, and I have looked at just about every thing that is available for every platform that I use. Vim is the best on all of them. (Guy L. Oliver)

Vim is the greatest editor since the stone chisel. (Jose Unpingco, USA)

I would like to say that with Vim I am finally making the 'emacs to vi' transition - as an Editor it is so much better in many ways: keyboard layout, memory usage, text alteration to name 3. (Mark Adam)

In fact, now if I want to know what a particular setting does in vi, I fire up Vim and check out its help! (Nikhil Patel, USA)

As a vi user, Vim has made working with text a far more pleasant task than before I encountered this program. (Steinar Knutsen, Norway)

I use Vim since version 3.0. Since that time, it is the ONLY editor I use, with Solaris, Linux and OS/2 Warp. I suggest all my friends to use Vim, they try, and they continue using it. Vim is really the best software I have ever downloaded from the Internet, and the best editor I know of. (Marco Eccettuato, Italy)

In summary:

```
__      ___                 _       _   _  ___ _____  `
\ \    / (_)_ __ ___       (_)___  | | | |/ _ \_   _| `
 \ \ / /| | '_ ` _ \ _____ | / __| | |_| | | | || |  `
  \ V / | | | | | | |_____|| \__ \ |  _  | |_| || |  `
   \_/  |_|_| |_| |_|      |_|___/ |_| |_|\___/ |_|  `
                                                     `
     ____ _____ _   _ _____ _____ _ _ `
    / ___|_   _| | | |  ___|  ___| | | `
    \___ \ | | | | | | |_  | |_  | | | `
     ___) || | | |_| |  _| |  _| |_|_| `
    |____/ |_|  \___/|_|   |_|   (_|_)      (Tony Nugent, Australia) `
```

vim:tw=78:ts=8:ft=help:norl:
todo.txt For Vim version 8.0. Last change: 2017 Sep 27

VIM REFERENCE MANUAL by Bram Moolenaar

TODO list for Vim *todo*

This is a veeeery long list of known bugs, current work and desired
improvements. To make it a little bit accessible, the items are grouped by
subject. In the first column of the line a classification is used to be able
to look for "the next thing to do":

Priority classification:
9 next point release
8 next release
7 as soon as possible
6 soon
5 should be included
4 nice to have
3 consider including
2 maybe not
1 probably not
- unclassified

votes-for-changes
See |develop.txt| for development plans. You can vote for which items should
be worked on, but only if you sponsor Vim development. See |sponsor|.

Issues can also be entered online: https://github.com/vim/vim/issues
Only use this for bug reports, not for questions! Those belong on the
maillist. Updates will be forwarded to the |vim_dev| maillist. Issues
entered there will not be repeated below, unless there is extra information.

known-bugs
-------------------- Known bugs and current work ----------------------

MS-Windows build and installer improvements:
- Switch to VC2015 for building. (Ken Takata, 2017 Sep 21)
  Check resulting binary on XP.
- Patch to install 32 and 64 bit Gvimext and related dll files. (Ken Takata,
  2017 Sep 23, #2144)

:term hangs in Athena and Motif. (Kazunobu Kuriyama, 2017 Sep 17)

Universal solution to detect if t_RS is working, using cursor position.
Koichi Iwamoto, #2126

No maintainer for Vietnamese translations.
No maintainer for Simplified Chinese translations.

Terminal emulator window:
- Lots of stuff to implement, see src/terminal.c
- Improve debugger interface:
  Include all debug features of Agide.
    - Implement the right-click popup menu for the terminal. Can use the
      completion popup menu code and mouse dragging.
      Use it for "set breakpoint", "remove breakpoint", etc.
    - make showballoon() work in a terminal. Requires getting mouse-move
      events.
    - send 'balloonText' events for the cursor position (using CursorHold ?)
      in terminal mode.
  - get ideas from http://clewn.sf.net

```
  - Look into the idevim plugin/script.
- Improve testing:
  Make a screenshot of a terminal, store in a file.
  Display a stored screenshot, display diff with another one.
  Make a test that puts Vim in a specific state, make a screenshot and compare
  with the expected screenshot.  Set t_Co to 256.

+channel:
- Add a separate timeout for opening a socket.  Currently it's fixed at 50
  msec, which is too small for a remote connection. (tverniquet, #2130)
- Try out background make plugin:
  https://github.com/AndrewVos/vim-make-background
- Problem with stderr on Windows? (Vincent Rischmann, 2016 Aug 31, #1026)
- Writing raw mode to a buffer should still handle NL characters as line
  breaks. (Dmitry Zotikov, 2017 Aug 16)
- When out_cb executes :sleep, the close_cb may be invoked. (Daniel Hahler,
  2016 Dec 11, #1320)
- Implement |job-term| ?
- Channel test fails with Motif.  Sometimes kills the X11 server.
- When a message in the queue but there is no callback, drop it after a while?
  Add timestamp to queued messages and callbacks with ID, remove after a
  minute.  Option to set the droptime.
- Add an option to drop text of very long lines?  Default to 1 Mbyte.
- Add remark about undo sync, is there a way to force it?
- When starting a job, have an option to open the server socket, so we know
  the port, and pass it to the command with --socket-fd {nr}. (Olaf Dabrunz,
  Feb 9)  How to do this on MS-Windows?
- For connection to server, a "keep open" flag would be useful.  Retry
  connecting in the main loop with zero timeout.
- job_start(): run job in a newly opened terminal (not a terminal window).
    With xterm could use -S{pty}.
    Although user could use "xterm -e 'cmd arg'".

Regexp problems:
- [:space:] only matches ASCII spaces.  Add [:white:] for all space-like
  characters, esp. including 0xa0.  Use character class zero.
- Since 7.4.704 the old regex engine fails to match [[:print:]] in 0xf6.
  (Manuel Ortega, 2016 Apr 24)
  Test fails on Mac.  Avoid using isalpha(), isalnum(), etc?  Depends on
  LC_CTYPE
- The old engine does not find a match for "/\%#=1\(\)\{80}", the new engine
  matches everywhere.
- Using win_linetabsize() can still be slow. Cache the result, store col and
  vcol. Reset them when moving to another line.
- Very slow with a long line and Ruby highlighting. (John Whitley, 2014 Dec 4)
- Bug with pattern: '\vblock (\d+)\.\n.*\d+%(\1)@<!\.$'
  (Lech Lorens, 2014 Feb 3)
- Issue 164: freeze on regexp search.
- Ignorecase not handled properly for multi-byte characters. (Axel Bender,
  2013 Dec 11)
- Using \@> and \?. (Brett Stahlman, 2013 Dec 21) Remark from Marcin
  Szamotulski; Remark from Brett 2014 Jan 6 and 7.
- NFA regexp doesn't handle \%<v correctly. (Ingo Karkat, 2014 May 12)
- Does not work with NFA regexp engine:
  \%u, \%x, \%o, \%d followed by a composing character
- Search for \%d0\+ may fail with E363. (Christian Brabandt, 2016 Oct 4)
- \%'[ does not work.  '%'] does work.  (Masaaki Nakamura, 2016 Apr 4)
- Bug relating to back references. (Ingo Karkat, 2014 Jul 24)
- New RE does not give an error for empty group: "\(\)\{2}" (Dominique Pelle,
  2015 Feb 7)
- Using back reference before the capturing group sometimes works with the old
  engine, can we do this with the new engine?  E.g. with
```

```
  "/\%(<\1>\)\@<=.*\%(<\/\(\w\+\)>\)\@=" matching text inside HTML tags.
  This problem is probably the same: "\%(^\1.*$\n\)\@<=\(\d\+\).*$".
  (guotuofeng, 2015 Jun 22)
- Strange matching with "\(Hello\n\)\@<=A". (Anas Syed, 2015 Feb 12)
- Problem with \v(A)@<=b+\1c. (Issue 334)
- Diff highlighting can be very slow. (Issue 309)
- Using %> for a virtual column has a check based on 'tabsize'.  Better would
  be to cache the result of win_linetabsize(col), storing both col and vcol,
  and use them to decide whether win_linetabsize() needs to be called.  Reset
  col and vcol when moving to another line.
- this doesn't work: "syntax match ErrorMsg /.\%9l\%>20c\&\%<28c/".  Leaving
  out the \& works.  Seems any column check after \& fails.
- Difference between two engines: ".*\zs\/\@>\/" on text "///"
  (Chris Paul, 2016 Nov 13)  New engine not greedy enough?
  Another one: echom matchstr(" sdfsfsf\n sfdsdfsdf",'[^\n]*')
  (2017 May 15, #1252)

Include a few color schemes, based on popularity:
http://www.vim.org/scripts/script_search_results.php?
keywords=&script_type=color+scheme&order_by=rating&direction=descending&search=search
http://vimawesome.com/?q=tag:color-scheme
Use names that indicate their appearnce (Christian Brabandt, 2017 Aug 3)
- monokai - Xia Crusoe (2017 Aug 4)
- seoul256 - Christian Brabandt (2017 Aug 3)
- gruvbox -  Christian Brabandt (2017 Aug 3)
- janah - Marco Hinz (2017 Aug 4)
- apprentice - Romain Lafourcade (2017 Aug 6)  remarks about help file #1964
Suggested by Hiroki Kokubun:
- [Iceberg](https://github.com/cocopon/iceberg.vim) (my one)
- [hybrid](https://github.com/w0ng/vim-hybrid)
Include solarized color scheme?

When starting with --clean packages under "start" are not loaded.  Make this
work:  :packadd START {name}  similar to :runtime START name

When using :packadd files under "later" are not used, which is inconsistent
with packages under "start". (xtal8, #1994)

After 8.0.0962 pasting leaves the cursor in another position. (Ken Takata,
2017 Aug 23, #2015) Also (zdm, 2017 Aug 23)

Patch to fix popup menu drawing when changing the window size. (Ozaki Kiichi,
2017 Sep 17, #2110)

Patch to fix cursor highlighting with match. (Ozaki Kiichi, 2017 Sep 17,
#2111)

Patch for not profiling the first line of a script. (Lemonboy, 2017 Sep 17,
#2103)

Mac Terminal.app: ctermbg=15 gives light grey instead of white.
ctermbg=256 breaks clearing till end of the line.  Both work fine in xterm.

Patch to avoid `rb_load_protect` as a workaround not to crash (#2147)

Patch for drag&drop reordering of GUI tab pages reordering.
(Ken Takata, 2013 Nov 22, second one, also by Masamichi Abe)
Now on Git: https://gist.github.com/nocd5/165286495c782b815b94
Update 2016 Aug 10.

Using ":hi" causes a redraw, but a redraw may update the status line, which
may trigger a ":hi" command.
```

Last line not in profile if it is a continuation line. (LemonBoy, 2017 Sep 17,
#2112)

With foldmethod=syntax and nofoldenable comment highlighting isn't removed.
(Marcin Szewczyk, 2017 Apr 26)

Patch to make Mac features more clear and add "macdarwin". (Kazunobu Kuriyama,
2017 Sep 5)

Using 'wildignore' also applies to literally entered file name. Also with
:drop (remote commands).

ml_get error when using a Python. (Yggdroot, 2017 Jun 1, #1737)
Lemonboy can reproduce (2017 Jun 5)

ml_get errors with buggy script. (Dominique, 2017 Apr 30)

Error in emsg with buggy script. (Dominique, 2017 Apr 30)

Patch to make ":set scroll&" work properly. (Ozaki Kiichi, 2017 Sep 17, #2104)

mswin.vim should not map CTRL-F in the console (#2093)
Patch from Christian, 2017 Sep 15.
Installer patch from Ken Takata, link on #2093.

Default install on MS-Windows should source defaults.vim.
Ask whether to use Windows or Vim key behavior?

matchit hasn't been maintained for a long time. #955.

Test runtime files.
Start with filetype detection: testdir/test_filetype.vim

Window not closed when deleting buffer. (Harm te Hennepe, 2017 Aug 27, #2029)

Add options_default() / options_restore() to set several options to Vim
defaults for a plugin. Comments from Zyx, 2017 May 10.
Perhaps use a vimcontext / endvimcontext command block.

After using :noautocmd CursorMoved may still trigger. (Andy Stewart, 2017 Sep
13, #2084). Set old position after the command.

Illegal memory access, requires ASAN to see. (Dominique Pelle, 2015 Jul 28)
Still happens (2017 Jul 9)

When bracketed paste is used, pasting at the ":append" prompt does not get the
line breaks. (Ken Takata, 2017 Aug 22)

This example in the help does not work (Andy Wokula, 2017 Aug 20):
    augroup mine | au! BufRead | augroup END

Memory leaks in test_channel? (or is it because of fork())
Memory leak in test_arabic.
Using uninitialized value in test_crypt.

Patch to make gM move to middle of line. (Yasuhiro Matsumoto, Sep 8, #2070)

Include Haiku port. (Adrien Destugues, Siarzhuk Zharski, 2013 Oct 24)
It can replace the BeOS code, which is likely not used anymore.
Now on github: #1856.
Got permission to include this under the Vim license.

```
Refactored HTML indent file. (Michael Lee, #1821)

Test_writefile_fails_conversion failure on Solaris because if different iconv
behavior.  Skip when "uname" returns "SunOS"? (Pavel Heimlich, #1872)

All functions are global, which makes functions like get() and len() awkward.
For the future use the ~get() and ~len() syntax, e.g.:
    mylist~get(idx)
    mydict~get(idx)
    mystring~len()
Alternatives for ~:
    ^  list^get()    could also be used
    .  list.get()    already means concatenate
    $  list$get()    harder to read
    @  list@get()    harder to read
    ->  list->get()  two characters, used for lambda

The ++ options for the :edit command are also useful on the Vim command line.

Overlong utf-8 sequence is displayed wrong. (Harm te Hennepe, 2017 Sep 14,
#2089)  Patch with possible solution by Björn Linse.

X11: Putting more than about 262040 characters of text on the clipboard and
pasting it in another Vim doesn't work.  (Dominique Pelle, 2008 Aug 21-23)
clip_x11_request_selection_cb() is called with zero value and length.
Also: Get an error message from free() in the process that owns the selection.
Seems to happen when the selection is requested the second time, but before
clip_x11_convert_selection_cb() is invoked, thus in X library code.
Kazunobu Kuriyama is working on a proper fix. (2017 Jul 25)

Problem with three-piece comment. (Michael Lee, 2017 May 11, #1696)

Creating a partial with an autoload function is confused about the "self"
attribute of the function.  For an unknown function assume "self" and make
that optiona? (Bjorn Linse, 2017 Aug 5)

Cindent: returning a structure has more indent for the second item.
(Sam Pagenkopf, 2017 Sep 14, #2090)

Completion mixes results from the current buffer with tags and other files.
Happens when typing CTRL-N while still search for results.  E.g., type "b_" in
terminal.c and then CTRL-N twice.
Should do current file first and not split it up when more results are found.
(Also #1890)

Python: After "import vim" error messages only show the first line of the
stack trace. (Yggdroot, 2017 Jul 28, #1887)

When checking if a bufref is valid, also check the buffer number, to catch the
case of :bwipe followed by :new.

Patch to skip writing a temp file for diffing if the buffer is equal to the
existing file. (Akria Sheng, 2017 Jul 22)
Could also skip writing lines that are the same.

Patch with Files for Latvian language. (Vitolins, 2017 May 3, #1675)

MS-Windows: Opening same file in a second gvim hangs. (Sven Bruggemann, 2017
Jul 4)

Setting 'clipboard' to "unnamed" makes a global command very slow (Daniel
```

Drucker, 2017 May 8).
This was supposed to be fixed, did it break again somehow?
Christian cannot reproduce it.

Using composing char in mapping does not work properly. maparg() shows the
wrong thing. (Nikolai Pavlov, 2017 Jul 8, #1827)
Or is this not an actual problem?

Better TeX indent file. (Christian Brabandt, 2017 May 3)

Openhab syntax file (mueller, #1678)

Patch to use a separate code for BS on Windows. (Linwei, #1823)

Use gvimext.dll from the nightly build? (Issue #249)

'synmaxcol' works with bytes instead of screen cells. (Llandon, 2017 May 31,
#1736)

Problem with using :cd when remotely editing a file. (Gerd Wachsmuth, 2017 May
8, #1690)

Running test_gui and test_gui_init with Motif sometimes kills the window
manager. Problem with Motif?

Bogus characters inserted when triggering indent while changing text.
(Vitor Antunes, 2016 Nov 22, #1269)

Using "wviminfo /tmp/viminfo" does not store file marks that Vim knows about,
it only works when merging with an existing file. (Shougo, 2017 Jun 19, #1781)

Segmentation fault with complete(). (Lifepillar, 2017 Apr 29, #1668)
Check for "pat" to be NULL in search_for_exact_line()?
How did it get NULL? Comment by Christian, Apr 30.

Is it possible to keep the complete menu open when calling complete()?
(Prabir Shrestha, 2017 May 19, #1713)

Memory leak in test97? The string is actually freed. Weird.

Patch to add configure flags to skip rtl, farsi and arabic support.
(Diego Carrión, #1867)

assert_fails() can only check for the first error. Make it possible to have
it catch multiple errors and check all of them.

New value "uselast" for 'switchbuf'. (Lemonboy, 2017 Apr 23, #1652)

Add a toolbar in the terminal. Can be global, above all windows, or specific
for one window.

Make maparg() also return the raw rhs, so that it doesn't depend on 'cpo'.
(Brett Stahlman, 2017 May 23)
Even better: add a way to disable a mapping temporarily and re-enable it
later. This is for a sub-mode that is active for a short while (one buffer).
Still need maplist() to find the mappings. What can we use to identify a
mapping? Something unique would be better than the LHS.
Perhaps simpler: actually delete the mappings. Use maplist() to list matching
mappings (with a lhs prefix, like maparg()), mapdelete() to delete,
maprestore() to restore (using the output of maplist()).

Add an argument to :mkvimrc (or add aother command) to skip mappings from

plugins (source is a Vim script). No need to put these in a .vimrc, they will
be defined when the plugin is loaded.

Use tb_set(winid, [{'text': 'stop', 'cb': callback, 'hi': 'Green'}])
    tb_highlight(winid, 'ToolBar')
    tb_get(winid)

json_encode(): should convert to utf-8. (Nikolai Pavlov, 2016 Jan 23)
What if there is an invalid character?

Json string with trailing \u should be an error. (Lcd)

import can't be used in define option when include matches too.
(Romain Lafourcade, 2017 Jun 18, #1519)

When session file has name in argument list but the buffer was deleted, the
buffer is not deleted when using the session file. (#1393)
Should add the buffer in hidden state.

When an item in the quickfix list has a file name that does not exist, behave
like the item was not a match for :cnext.

Wrong diff highlighting with three files. (2016 Oct 20, #1186)
Also get E749 on exit.
Another example in #1309

When deleting a mark or register, leave a tombstone, so that it's also deleted
when writing viminfo (and the delete was the most recent action). #1339

Suggestion to improve pt-br spell checking. (Marcelo D Montu, 2016 Dec 15,
#1330)

Error in test_startup_utf8 on Solaris. (Danek Duvall, 2016 Aug 17)

Completion for :!cmd shows each match twice. #1435

Patch to change GUI behavior: instead of changing the window size change the
lines/columns when menu/toolbar/etc. is added/removed. (Ychin, 2016 Mar 20,
#703)

GTK: When adding a timer from 'balloonexpr' it won't fire, because
g_main_context_iteration() doesn't return. Need to trigger an event when the
timer expires.

Screen update bug related to matchparen. (Chris Heath, 2017 Mar 4, #1532)

Rule to use "^" for statusline does not work if a space is defined with
highlighting for both stl and stlnc. Patch by Ken Hamada (itchyny, 2016 Dec 11)

8   "stl" and "stlnc" in 'fillchars' don't work for multi-byte characters.
    Patch by Christian Wellenbrock, 2013 Jul 5.

Using CTRL-G_U in InsertCharPre causes trouble for redo. (Israel Chauca
Fuentes, 2017 Feb 12, #1470)

Add a "keytrans()" function, which turns the internal byte representation of a
key into a form that can be used for :map. E.g.
    let xx = "\<C-Home>"
    echo keytrans(xx)
    <C-Home>

Check for errors E704 and E705 only does VAR_FUNC, should also do VAR_PARTIAL.

(Nikolai Pavlov, 2017 Mar 13, #1557)
Make a function to check for function-like type?

Screen updated delayed when using CTRL-O u in Insert mode.
(Barlik, #1191) Perhaps because status message?

Implement optional arguments for functions.
```
    func Foo(start, count = 1 all = 1)
    call Foo(12)
    call Foo(12, all = 0)
    call Foo(12, 15, 0)
```

Change the Farsi code to work with UTF-8. Possibly combined with the Arabic
support, or similar.
Invalid read error in Farsi mode. (Dominique Pelle, 2009 Aug 2)

Add a command to take a range of lines, filter them and put the output
somewhere else. :{range}copy {dest} !cmd

Patch to fix that empty first tab is not in session.
(Hirohito Higashi, 2016 Nov 25, #1282)

Patch to add random number generator. (Hong Xu, 2010 Nov 8, update Nov 10)
Alternative from Christian Brabandt. (2010 Sep 19)
New one from Yasuhiro Matsumoto, #1277.

Patch to fix escaping of job arguments. (Yasuhiro Matsumoto, 2016 Oct 5)
Update Oct 14: https://gist.github.com/mattn/d47e7d3bfe5ade4be86062b565a4bfca
Update Aug 2017: #1954

Characters deleted on completion. (Adrià Farrés, 2017 Apr 20, #1645)
Remarks from Christian Brabandt (Apr 21)

The TermResponse event is not triggered when a plugin has set 'eventignore' to
"all". Netrw does this. (Gary Johnson, 2017 Jan 24)
Postpone the event until 'eventignore' is reset.

Expanding /**/ is slow. Idea by Luc Hermitte, 2017 Apr 14.

Once .exe with updated installer is available: Add remark to download page
about /S and /D options (Ken Takata, 2016 Apr 13)
Or point to nightly builds: https://github.com/vim/vim-win32-installer/releases

Problem passing non-UTF-8 strings to Python 3. (Björn Linse, 2016 Sep 11,
#1053) With patch, does it work?

Using --remote to open a file in which a # appears does not work on
MS-Windows. Perhaps in \# the \ is seen as a path separator. (Axel Bender,
2017 Feb 9) Can we expand wildcards first and send the path literally to the
receiving Vim? Or make an exception for #, it's not useful remotely.

":sbr" docs state it respects 'switchbuf', but "vsplit" does not cause a
vertical split. (Haldean Brown, 2017 Mar 1)

Use ADDR_OTHER instead of ADDR_LINES for many more commands.
Add tests for using number larger than number of lines in buffer.

Might be useful to have isreadonly(), like we have islocked().
Avoids exceptions, e.g. when using the b: namespace as a dict.

Patch to make v:shell_error writable. (Christian Brabandt, 2016 Sep 27)
Useful to restore it. Is there another solution?

"ci[" does not look for next [like ci" does look for next ".
(J.F. 2017 Jan 7)

Patch for wrong cursor position on wrapped line, involving breakindent.
(Ozaki Kiichi, 2016 Nov 25)
Does this also fix #1408 ?

Patch to add "module" to quickfix entries. (Coot, 2017 Jun 8, #1757)

'cursorline' and match interfere. (Ozaki Kiichi, 2017 Jun 23, #1792)

Patch for 'cursorlinenr' option. (Ozaki Kiichi, 2016 Nov 30)

Patch to be able to separately map CTRL-H and BS on Windows.
(Linwei, 2017 Jul 11, #1833)

When 'completeopt' has "noselect" does not insert a newline. (Lifepillar, 2017
Apr 23, #1653)

Using an external diff is inefficient. Not all systems have a good diff
program available (esp. MS-Windows). Would be nice to have in internal diff
implementation. Can then also use this for displaying changes within a line.
Olaf Dabrunz is working on this. (10 Jan 2016)
9   Instead invoking an external diff program, use builtin code. One can be
    found here: http://www.ioplex.com/~miallen/libmba/dl/src/diff.c
    It's complicated and badly documented.

Window resizing with 'winfixheight': With a vertical split the height changes
anyway. (Tommy allen, 2017 Feb 21, #1502)

When adding an item to a new quickfix list make ":cnext" jump to that item.
Make a difference being at the first item and not having used :cnext at all.
(Afanasiy Fet, 2017 Jan 3)

Invalid behavior with NULL list. (Nikolai Pavlov, #768)
E.g. deepcopy(test_null_list())

Patch to make it possible to extend a list with itself.
(Nikolai Pavlov, 2016 Sep 23)

Patch to add Zstandard compressed file support. (Nick Terrell, 2016 Oct 24)

Patch to add trim() function. (Bukn, 2016 Nov 25, #1280)

Patch to add MODIFIED_BY to MSVC build file. (Chen Lei, 2016 Nov 24, #1275)

Patch to change argument of :marks. (LemonBoy, 2017 Jan 29, #1426)

On Windows buffer completion sees backslash as escape char instead of path
separator. (Toffanim, 2016 Nov 24, #1274)

min() and max() spawn lots of error messages if sorted list/dictionary
contains invalid data (Nikolay Pavlov, 2016 Sep 4, #1039)

Should :vmap in matchit.vim be :xmap? (Tony Mechelynck)

Problem with whitespace in errorformat. (Gerd Wachsmuth, 2016 May 15, #807)

Undo problem: "g-" doesn't go back, gets stuck. (Björn Linse, 2016 Jul 18)

Add "unicode true" to NSIS installer. Doesn't work with Windows 95, which we

no longer support.

sort() is not stable when using numeric/float sort (Nikolay Pavlov, 2016 Sep
4#1038)

Patch to add "cmdline" completion to getcompletion(). (Shougo, Oct 1, #1140)

Feature request: Complete members of a dictionary. (Luc Hermitte, 2017 Jan 4,
#1350)

Undo message is not always properly displayed. Patch by Ken Takata, 2013 oct
3. Doesn't work properly according to Yukihiro Nakadaira.
Also see #1635.

Patch for systemlist(), add empty item. (thinca, Sep 30, #1135)
Add an argument to choose binary or non-binary (like readfile()), when omitted
use the current behavior.
Include the test.

Patch to add tagfunc(). Cleaned up by Christian Brabandt, 2013 Jun 22.
New update 2017 Apr 10, #1628

When 'keywordprg' starts with ":" the argument is still escaped as a shell
command argument. (Romain Lafourcade, 2016 Oct 16, #1175)

Patch to support CamelCase for spell checking: See a lower-to-upper case
change as a word boundary. (btucker-MPCData, 2016 Nov 6, #1235)

Idea from Sven: record sequence of keys. Useful to show others what they are
doing (look over the shoulder), and also to see what happened.
Probably list of keystrokes, with some annotations for mode changes.
Could store in logfile to be able to analyse it with an external command.
E.g. to see when's the last time a plugin command was used.

execute() cannot be used with command completeion. (Daniel Hahler, 2016 Oct 1,
#1141)

cmap using execute() has side effects. (Killthemule, 2016 Aug 17, #983)

:map X may print invalid data. (Nikolay Pavlov, 2017 Jul 3, #1816)

Patch to order results from taglist(). (Duncan McDougall, 2016 Oct 25)

patch for 'spellcamelcase' option: spellcheck each CamelCased word.
(Ben Tucker, 2016 Dec 2)

When using ":diffput" through a mapping, undo in the target buffer isn't
synced. (Ryan Carney, 2016 Sep 14)

Syntax highlighting for messages with RFC3339 timestamp (#946)
Did maintainer reply?

Patch to avoid problem with special characters in file name.
(Shougo, 2016 Sept 19, #1099) Not finished?

ml_get errors when reloading file. (Chris Desjardins, 2016 Apr 19)
Also with latest version.

Cannot delete a file with square brackets with delete(). (#696)

Patch to add ":syn foldlevel" to use fold level further down the line.
(Brad King, 2016 Oct 19, update 2017 Jan 30)

Completion for input() does not expand environment variables. (chdiza, 2016
Jul 25, #948)

Patch to fix wrong encoding of error message on Cygwin/MSYS terminal.
(Ken Takata, 2016 Oct 4)

Patch to add 'systemencoding', convert between 'encoding' and this for file
names, shell commands and the like. (Kikuchan, 2010 Oct 14)
Assume the system converts between the actual encoding of the filesystem to
the system encoding (usually utf-8).

Using ":tab drop file" does not trigger BufEnter or TabEnter events.
(Andy Stewart, 2017 Apr 27, #1660)
Autocommands blocked in do_arg_all(). Supposed to happen later?

'hlsearch' interferes with a Conceal match. (Rom Grk, 2016 Aug 9)

MS-Windows: use WS_HIDE instead of SW_SHOWMINNOACTIVE in os_win32.c?
Otherwise task flickers in taskbar.

Should make ":@r" handle line continuation. (Cesar Romani, 2016 Jun 26)
Also for ":@.".

Repeating 'opfunc' in a function only works once. (Tarmean, 2016 Jul 15, #925)

Have a way to get the call stack, in a function and from an exception.
#1125

Patch to add 'pythonhome' and 'pythonthreehome' options. (Kazuki Sakamoto,
2016 Nov 21, #1266)

Second problem in #966: ins_compl_add_tv() uses get_dict_string() multiple
times, overwrites the one buffer. (Nikolay Pavlov, 2016 Aug 5)

This does not work: :set cscopequickfix=a-
(Linewi, 2015 Jul 12, #914)

Possibly wrong value for seq_cur. (Florent Fayolle, 2016 May 15, #806)

Filetype plugin for awk. (Doug Kearns, 2016 Sep 5)

Patch to improve map documentation. Issue #799.

Patch for syntax folding optimization. (Shougo, 2016 Sep 6, #1045)

We can use '. to go to the last change in the current buffer, but how about
the last change in any buffer? Can we use ', (, is next to .)?

Ramel Eshed: system() is much slower than job_start(), why? (Aug 26)

When generating the Unicode tables with runtime/tools/unicode.vim the
emoji_width table has only one entry.

It's possible to add ",," to 'wildignore', an empty entry. Causes problems.
Reject the value? #710.

Patch to fix increment/decrement not working properly when 'virtualedit' is
set. (Hirohito Higashi, 2016 Aug 1, #923)

When doing "vi buf.md" a BufNew autocommand for *.md is not triggered.
Because of using the initial buffer? (Dun Peal, 2016 May 12)

Patch to add the :bvimgrep command. (Christian Brabandt, 2014 Nov 12)
Updated 2016 Jun 10, #858 Update 2017 Mar 28: use <buffer>

Add redrawtabline command. (Naruhiko Nishino, 2016 Jun 11)

Neovim patch for utfc_ptr2char_len() https://github.com/neovim/neovim/pull/4574
No test, needs some work to include.

Patch to improve indenting for C++ constructor with initializer list.
(Hirohito Higashi, 2016 Mar 31)

Zero-out krypt key information when no longer in use. (Ben Fritz, 2017 May 15)

Add stronger encryption. Could use libsodium (NaCl).
https://github.com/jedisct1/libsodium/
Possibly include the needed code so that it can be build everywhere.

Add a way to restart a timer. It's similar to timer_stop() and timer_start(),
but the reference remains valid.

Need to try out instructions in INSSTALLpc.txt about how to install all
interfaces and how to build Vim with them.
Appveyor build with self-installing executable, includes getting most
interfaces: https://github.com/k-takata/vim/tree/chrisbra-appveyor-build
result: https://ci.appveyor.com/project/k-takata/vim/history

Duplication of completion suggestions for ":!hom". Issue 539.
Patch by Christian, 2016 Jan 29
>
Problem that a previous silent ":throw" causes a following try/catch not to
work. (ZyX, 2013 Sep 28) With examples: (Malcolm Rowe, 2015 Dec 24)

Problem using ":try" inside ":execute". (ZyX, 2013 Sep 15)

Patch to make tests pass with EBCDIC. (Owen Leibman, 2016 Apr 10)

Add ":read :command", to insert the output of an Ex command?
Can already do it with ":$put =execute('command')".

When repeating the 'confirm' dialog one needs to press Enter. (ds26gte, 2016
Apr 17) #762

exists(":tearoff") does not tell you if the command is implemented. (Tony
Mechelynck) Perhaps use exists("::tearoff") to check?

Use vim.vim syntax highlighting for help file examples, but without ":" in
'iskeyword' for syntax.

Patch to make "%:h:h" return "." instead of the full path.
(Coot, 2016 Jan 24, #592)

Remove SPACE_IN_FILENAME ? What could possibly go wrong?

When command names are very long :command output is difficult to read. Use a
maximum for the column width? (#871)
Patcy by varmanishant, 2016 Jun 18, #876

Installation of .desktop files does not work everywhere.
It's now fixed, but the target directory probably isn't right.
Add configure check?
Should use /usr/local/share/applications or /usr/share/applications.

Or use $XDG_DATA_DIRS.
Also need to run update-desktop-database (Kuriyama Kazunobu, 2015 Nov 4)

Test object i{ and it do not behave the same. #1379
Do not include the linebreak at the start?

Patch to have text objects defined by arbitrary single characters. (Daniel
Thau, 2013 Nov 20, 2014 Jan 29, 2014 Jan 31)
Added tests (James McCoy, 2016 Aug 3). Still needs more work.

Feature request: add the "al" text object, to manipulate a screen line.
Especially useful when using 'linebreak'

Access to uninitialized memory in match_backref() regexp_nda.c:4882
(Dominique Pelle, 2015 Nov 6)

":cd C:\Windows\System32\drivers\etc*" does not work, even though the
directory exists. (Sergio Gallelli, 2013 Dec 29)

In debug mode one can inspect variables, but not the function parameters
(starting with a:). (Luc Hermitte, 2017 Jan 4, #1352)

7 Add a watchpoint in the debug mode: An expression that breaks execution
when evaluating to non-zero. Add the "watchadd expr" command, stop when
the value of the expression changes. ":watchdel" deletes an item,
":watchlist" lists the items. (Charles Campbell)
Patch by Christian Brabandt, 2016 Jun 10, #859

If ":bd" also closes a Tab page then the " mark is not set. (Harm te Hennepe,
2016 Apr 25, #780)

Patch to avoid redrawing tabline when the popup menu is visible.
(Christian Brabandt, 2016 Jan 28)

Patch to add {skip} argument to search(). (Christian Brabandt, 2016 Feb 24)
Update 2016 Jun 10, #861

Patch to be able to use hex numbers with :digraph. (Lcd, 2015 Sep 6)
Update Sep 7. Update by Christian Brabandt, 2015 Sep 8, 2016 Feb 1.

Patch to show search statistics. (Christian Brabandt, 2016 Jul 22)

When the CursorMovedI event triggers, and CTRL-X was typed, a script cannot
restore the mode properly. (Andrew Stewart, 2016 Apr 20)
Do not trigger the event?

Using ":windo" to set options in all windows has the side effect that it
changes the window layout and the current window. Make a variant that saves
and restores. Use in the matchparen plugin.
Perhaps we can use ":windo <restore> {cmd}"?
Patch to add <restore> to :windo, :bufdo, etc. (Christian Brabandt, 2015 Jan
6, 2nd message)
Alternative: ":keeppos" command modifier: ":keeppos windo {cmd}".

Patch to fix that executable() may fail on very long filename in MS-Windows.
(Ken Takata, 2016 Feb 1)

Patch to fix display of listchars on the cursorline. (Nayuri Aohime, 2013)
Update suggested by Yasuhiro Matsumoto, 2014 Nov 25:
https://gist.github.com/presuku/d3d6b230b9b6dcfc0477

Patch to make the behavior of "w" more straightforward, but not Vi compatible.

With a 'cpo' flag. (Christian Brabandt, 2016 Feb 8)

Patch to add optionproperties(). (Anton Lindqvist, 2016 Mar 27, update Apr 13)

Patch to add TagNotFound autocommand. (Anton Lindqvist, 2016 Feb 3)

Patch to add Error autocommand. (Anton Lindqvist, 2016 Feb 17)
Only remembers one error.

Gvim: when both Tab and CTRL-I are mapped, use CTRL-I not for Tab.

Unexpected delay when using CTRL-O u. It's not timeoutlen.
(Gary Johnson, 2015 Aug 28)

Instead of separately uploading patches to the ftp site, we can get them from
github with a URL like this:
   https://github.com/vim/vim/compare/v7.4.920%5E...v7.4.920.diff
Diff for version.c contains more context, can't skip a patch.

When t_Co is changed from termresponse, the OptionSet autocmmand event isn't
triggered. Use the code from the end of set_num_option() in
set_color_count().

Python: ":py raw_input('prompt')" doesn't work. (Manu Hack)

Comparing nested structures with "==" uses a different comparator than when
comparing individual items.
Also, "'' == 0" evaluates to true, which isn't nice.
Add "===" to have a strict comparison (type and value match).
Add "==*" (?) to have a value match, but no automatic conversion, and v:true
equals 1 and 1.0, v:false equals 0 and 0.0.?

Using uninitialized memory. (Dominique Pelle, 2015 Nov 4)

MS-Windows: When editing a file with a leading space, writing it uses the
wrong name. (Aram, 2014 Nov 7) Vim 7.4.

Can't recognize the $ProgramFiles(x86) environment variable. Recognize it
specifically? First try with the parens, then without.

Patch to add :mapgroup, put mappings in a group like augroup.
(Yasuhiro Matsumoto, 2016 Feb 19)

Value returned by virtcol() changes depending on how lines wrap. This is
inconsistent with the documentation.

Value of virtcol() for '[and '] depend on multi-byte character.
(Luchr, #277)

Can we cache the syntax attributes, so that updates for 'relativenumber' and
'cursorline'/'cursorcolumn' are a lot faster? Thus store the attributes
before combining them.

C highlighting: modern C allows: /* comment */ #ifdef
and also line continuation after #include.
I can't recommend it though.

Build with Python on Mac does not always use the right library.
(Kazunobu Kuriyama, 2015 Mar 28)

Patch to add arguments to argc() and argv(). (Yegappan Lakshmanan, 2016 Jan
24) Also need a way to get the global arg list? Update later on Jan 24

Update Mar 5. Update Apr 7. Update Jun 5.

To support Thai (and other languages) word boundaries, include the ICU
library: http://userguide.icu-project.org/boundaryanalysis

When complete() first argument is before where insert started and 'backspace'
is Vi compatible, the completion fails. (Hirohito Higashi, 2015 Feb 19)

Patch to use two highlight groups for relative numbers. (Shaun Brady, 2016 Jan
30)

MS-Windows: Crash opening very long file name starting with "\\".
(Christian Brock, 2012 Jun 29)

The OptionSet autocommand event is not always triggered. (Rick Howe, 2015 Sep
24): :diffthis, :diffoff.

":set all&" still does not handle all side effects. Centralize handling side
effects for when set by the user, on init and when reset to default.

":tag" does not jump to the right entry of a :tselect. (James Speros, 2015 Oct
9)

The argument for "-S" is not taken literally, the ":so" command expands
wildcards. Add a ":nowild" command modifier? (ZyX, 2015 March 4)

Proposal to make options.txt easier to read. (Arnaud Decara, 2015 Aug 5)
Update Aug 14.

When using --remote-tab on MS-Windows 'encoding' hasn't been initialized yet,
the file name ends up encoded wrong. (Raul Coronado, 2015 Dec 21)

Patch for problem with restoring screen on Windows. (Nobuhiro Takasaki, 2015
Sep 10)

Example in editing.txt uses $HOME with the expectation that it ends in a
slash. For me it does, but perhaps not for everybody. Add a function that
inserts a slash when needed? pathconcat(dir, path) (Thilo Six, 2015 Aug 12)

ml_updatechunk() is slow when retrying for another encoding. (John Little,
2014 Sep 11)

Patch to fix checking global option value when not using it.
(Arnaud Decara, 2015 Jul 23)

When 'showbreak' is set repeating a Visual operation counts the size of the
'showbreak' text as part of the operation. (Axel Bender, 2015 Jul 20)

Patch for multi-byte characters in langmap and applying a mapping on them.
(Christian Brabandt, 2015 Jun 12, update July 25)
Is this the right solution? Need to cleanup langmap behavior:
- in vgetorpeek() apply langmap to the typeahead buffer and put the result in
  a copy-buffer, only when langmap is appropriate for the current mode. Then
  check for mapping and let gotchars() work on the copy-buffer.
- Remove LANGMAP_ADJUST() in other parts of the code. Make sure the mode is
  covered by the above change.
So that replaying the register doesn't use keymap/langmap and still does the
same thing. Remarks on issue 543 (Roland Puntaier).
Also see #737: langmap not applied to replaying recording.

Patch to add grepfile(). (Scott Prager, 2015 May 26)
Work in progress.

Would be useful to have a treemap() or deepmap() function. Like map() but
when an item is a list or dict would recurse into it.

Patch for global-local options consistency. (Arnaud Decara, 2015 Jul 22)
Is this right?

Patch to make getregtype() return the right size for non-linux systems.
(Yasuhiro Matsumoto, 2014 Jul 8)
Breaks test_eval. Inefficient, can we only compute y_width when needed?

Patch to use different terminal mode settings for system(). (Hayaki Saito)
Does this work for everybody?

Patch for man.vim. (SungHyun Nam, 2015 May 20)
Doesn't work completely (Dominique Orban)

Patch to add a "literal" argument to bufnr(). (Olaf Dabrunz, 2015 Aug 4)

Cannot execute the shell when it's in a directory with a space.
Issue #459.

When a session file is created and there are "nofile" buffers, these are not
filled. Need to trigger BufReadCmd autocommands. Also handle deleting the
initial empty buffer better. (ZyX, 2015 March 8)

Extended file attributes lost on write (backupcopy=no). Issue 306.

Patch to add :lockjumps. (Carlo Baldassi, 2015 May 25)
OK to not block marks?

Mixup of highlighting when there is a match and SpellBad. (ZyX, 2015 Jan 1)

Patch on Issue 72: 'autochdir' causes problems for :vimgrep.

When two SIGWINCH arrive very quickly, the second one may be lost.
(Josh Triplett, 2015 Sep 17)

Make comments in the test Makefile silent. (Kartik Agaram, 2014 Sep 24)

Result of systemlist() does not show whether text ended in line break.
(Bjorn Linse, 2014 Nov 27)

When in 'comments' "n:x" follows after three-part comment directly it repeats
any one-character from the previous line. (Kartik Agaram, 2014 Sep 19)

Syntax highlighting slow (hangs) in SASS file. (Niek Bosch, 2013 Aug 21)

Adding "~" to 'cdpath' doesn't work for completion? (Davido, 2013 Aug 19)

Should be easy to highlight all matches with 'incsearch'. Idea by Itchyny,
2015 Feb 6.

Wrong scrolling when using incsearch. Patch by Christian Brabandt, 2014 Dec 4.
Is this a good solution?

Patch: Let rare word highlighting overrule good word highlighting.
(Jakson A. Aquino, 2010 Jul 30, again 2011 Jul 2)

Patch to add digits argument to round(). (Yasuhiro Matsumoto, 2015 Apr 26)

Can assign to s:type when a function s:type has been defined.

Also the other way around: define a function while a variable with that name
was already defined.
(Yasuhiro Matsumoto, 2014 Nov 3)

Patch for ordered dict. (Ozaki Kiichi, 2015 May 7)

Patch to make closed folds line up. (Charles Campbell, 2014 Sep 12)
Remark from Roland Eggner: does it cause crashes? (2014 Dec 12)
Updated patch by Roland Eggner, Dec 16
Updated patch from Charles, 2016 Jul 2

Patch to open folds for 'incsearch'. (Christian Brabandt, 2015 Jan 6)

Patch for building a 32bit Vim with 64bit MingW compiler.
(Michael Soyka, 2014 Oct 15)

Delete old code in os_msdos.c, mch_FullName().

Patch: On MS-Windows shellescape() may have to triple double quotes.
(Ingo Karkat, 2015 Jan 16)

Patch for variable tabstops. On github (Christian Brabandt, 2014 May 15)
Update 2016 Jun 10, # 857

Redo only remembers the last change. Could use "{count}g." to redo an older
change. How does the user know which change? At least have a way to list
them: ":repeats".

Patch for glob(), adding slash to normal files. (Ingo Karkat, 2014 Dec 22)

When entering and leaving the preview window autocommands are triggered, but
these may not work well. Perhaps set a flag to indicate that the preview
window is involved? (John Otter, 2015 Oct 27)

Using "." to repeat an Ex command puts that command in history. Probably
should not happen. If the command is the result of a mapping it's not put in
history either. (Jacob Niehus, 2014 Nov 2)
Patch from Jacob, Nov 2.

"hi link" does not respect groups with GUI settings only. (Mark Lodato, 2014
Jun 8)

Bug: Autocompleting ":tag/pat" replaces "/pat" with a match but does not
insert a space. (Micha Mos, 2014 Nov 7)

Patch to add argument to :cquit. (Thinca, 2014 Oct 12)

No error for missing endwhile. (ZyX, 2014 Mar 20)

Patch to make extend() fail early when it might fail at some point.
(Olaf Dabrunz, 2015 May 2) Makes extend() slower, do we still want it?
Perhaps only the checks that can be done without looping over the dict or
arguments.

Problem with transparent and matchgroup. Issue #475

Patch to add :arglocal and :arglists. (Marcin Szamotulski, 2014 Aug 6)

Spell files use a latin single quote. Unicode also has another single quote:
0x2019. (Ron Aaron, 2014 Apr 4)
New OpenOffice spell files support this with ICONV. But they are not
compatible with Vim spell files. The old files can no longer be downloaded.

xterm should be able to pass focus changes to Vim, so that Vim can check for
buffers that changed. Perhaps in misc.c, function selectwindow().
Xterm 224 supports it!
Patch to make FocusGained and FocusLost work in modern terminals. (Hayaki
Saito, 2013 Apr 24) Update 2016 Aug 12.
Also see issue #609.
We could add the enable/disable sequences to t_ti/t_te or t_ks/t_ke.

Idea: For a window in the middle (has window above and below it), use
right-mouse-drag on the status line to move a window up/down without changing
its height? It's like dragging the status bar above it at the same time.

Can we make ":unlet $VAR" use unsetenv() to delete the env var?
What for systems that don't have unsetenv()? (Issue #1116)

Patch to add a :domodeline command. (Christian Brabandt, 2014 Oct 21)

This does not give an error: (Andre Sihera, 2014 Mar 21)
    vim -u NONE 1 2 3 -c 'bufdo if 1 | echo 1'
This neither: (ZyX)
    vim -u NONE 1 2 3 -c 'bufdo while 1 | echo 1'

'viewdir' default on MS-Windows is not a good choice, it's a system directory.
Change 'viewdir' to "$HOME/vimfiles/view" and use 'viewdiralt' to also read
from?

Problem with upwards search on Windows (works OK on Linux). (Brett Stahlman,
2014 Jun 8)

Include a plugin manager with Vim? Neobundle seems to be the best currently.
Also Vundle: https://github.com/gmarik/vundle
Long message about this from ZyX, 2014 Mar 23. And following replies.
Also see http://vim-wiki.mawercer.de/wiki/topic/vim%20plugin%20managment.html
User view:
- Support multiple sources, basically any http:// URL. Or a central place that
  will work for everybody (github? redirects from vim.org?).
  Be able to look into the files before deciding to install.
- Be able to try out a plugin and remove it again with (almost) no traces.
- Each plugin needs a "manifest" file that has the version, dependencies
  (including Vim version and features), conflicts, list of files, etc.
  Updater uses that to decide what/how to update.
  Dependencies can use a URL for specific versions, or short name for scripts
  on vim.org.
- Once a plugin is installed it remembers where it came from, updater checks
  there. Can manually update when really needed.
- Must be possible to install for one user. Also system wide?
- Can edit plugin config with Vim. Can temporarily disable a plugin.
- Run the update manually, find latest version and install.
- Be able to download without special tools, must work for 95% of users.
Implementation:
- Avoid the 'runtimepath' getting long. Need some other way to keep each
  plugin separate.
- When installing or updating, first figure out what needs to be done. This
  may involve recursively fetching manifest files for dependencies. Then show
  the user what's going to change and ask for OK.
- Scripts on Vim.org must be able to consist of several files. Is zip format
  sufficient? Upload the manifest? Or refer to a site that has the manifest?
- Best is to fetch individual files or use a Vimball. Reduces dependency on
  tools that might be missing and allows inspection of the files before
  installing.
Out of scope:

- Overview of plugins, ratings, comments, etc. That's another world.
- Development work on plugins (although diff with distributed version would be
  useful).

Setting the spell file in a session only reads the local additions, not the
normal spell file. (Enno Nagel, 2014 Mar 29)

When typing the first character of a command, e.g. "f", then using a menu, the
menu item doesn't work. Clear typeahead when using a menu?

Editing an ascii file as ucs-2 or ucs-4 causes display errors.
(ZyX, 2014 Mar 30)

":Next 1 some-arg" does not complain about trailing argument. Also for
various other commands. (ZyX, 2014 Mar 30)

Patch to skip sort if no line matches the expression.
(Christian Brabandt, 2014 Jun 25)

VMS: Select() doesn't work properly, typing ESC may hang Vim. Use sys$qiow
instead. (Samuel Ferencik, 2013 Sep 28)

Patch for XDG base directory support. (Jean François Bignolles, 2014 Mar 4)
Remark on the docs. Should not be a compile time feature. But then what?

Completion of ":e" is ":earlier", should be ":edit". Complete to the matching
command instead of doing this alphabetically. (Mikel Jorgensen)

Patch to define macros for hardcoded values. (Elias Diem, 2013 Dec 14)

Several syntax file match "^\s*" which may get underlined if that's in the
highlight group. Add a "\zs" after it?

The undo file name can get too long. (Issue 346)
For the path use a hash instead of dir%dir%dir%name hash%name.

Patch to add ":undorecover", get as much text out of the undo file as
possible. (Christian Brabandt, 2014 Mar 12, update Aug 22)

Updated spec ftplugin. (Matěj Cepl, 2013 Oct 16)

Patch to right-align signs. (James Kolb (email james), 2013 Sep 23)

Patch to handle integer overflow. (Aaron Burrow, 2013 Dec 12)

Patch to add "ntab" item in 'listchars' to repeat first character. (Nathaniel
Braun, pragm, 2013 Oct 13) A better solution 2014 Mar 5.

7   Windows XP: When using "ClearType" for text smoothing, a column of yellow
    pixels remains when typing spaces in front of a "D" ('guifont' set to
    "lucida_console:h8").
Patch by Thomas Tuegel, also for GTK, 2013 Nov 24

:help gives example for z?, but it does not work. m? and t? do work.

Patch to add funcref to Lua. (Luis Carvalho, 2013 Sep 4)
With tests: Sep 5.

Discussion about canonicalization of Hebrew. (Ron Aaron, 2011 April 10)

Checking runtime scripts: Thilo Six, 2012 Jun 6.

When evaluating expression in backticks, autoload doesn't work.
(Andy Wokula, 2013 Dec 14)

Using <nr>ifoobar<esc> can slow down Vim. Patch by Christian Brabandt, 2013
Dec 13.

Patch from Christian Brabandt to preserve upper case marks when wiping out a
buffer. (2013 Dec 9)

GTK: problem with 'L' in 'guioptions' changing the window width.
(Aaron Cornelius, 2012 Feb 6)

Patch to add option that tells whether small deletes go into the numbered
registers. (Aryeh Leib Taurog, 2013 Nov 18)

Javascript file where indent gets stuck on: GalaxyMaster, 2012 May 3.

The BufUnload event is triggered when re-using the empty buffer.
(Pokey Rule, 2013 Jul 22)
Patch by Marcin Szamotulski, 2013 Jul 22.

The CompleteDone autocommand needs some info passed to it:
- The word that was selected (empty if abandoned complete)
- Type of completion: tag, omnifunc, user func.

Patch to allow more types in remote_expr(). (Lech Lorens, 2014 Jan 5)
Doesn't work for string in list. Other way to pass all types of variables
reliably?

Using ":call foo#d.f()" doesn't autoload the "foo.vim" file.
That is, calling a dictionary function on an autoloaded dict.
Works OK for echo, just not for ":call" and ":call call()". (Ted, 2011 Mar
17)
Patch by Christian Brabandt, 2013 Mar 23.
Not 100% sure this is the right solution.

Patch to add {lhs} to :mapclear: clear all maps starting with {lhs}.
(Christian Brabandt, 2013 Dec 9)

Exception caused by argument of return is not caught by try/catch.
(David Barnett, 2013 Nov 19)

Patch to fix that 'cedit' is recognized after :normal. (Christian Brabandt,
2013 Mar 19, later message)

Patch to view coverage of the tests. (Nazri Ramliy, 2013 Feb 15)

Patch to invert characters differently in GTK. (Yukihiro Nakadaira, 2013 May
5)

Patch to add "Q" and "A" responses to interactive :substitute. They are
carried over when using :global. (Christian Brabandt, 2013 Jun 19)

Bug with 'cursorline' in diff mode. Line being scrolled into view gets
highlighted as the cursor line. (Alessandro Ivaldi, 2013 Jun 4)

Two highlighting bugs. (ZyX, 2013 Aug 18)

Patch to support 'u' in interactive substitute. (Christian Brabandt, 2012 Sep
28) With tests: Oct 9.

Patch from Christian Brabandt to make the "buffer" argument for ":sign place"

optional. (2013 Jul 12)

Dialog is too big on Linux too. (David Fishburn, 2013 Sep 2)

Improve the installer for MS-Windows. There are a few alternatives:
- Add silent install option. (Shane Lee, #751)
- Installer from Cream (Steve Hall).
- Modern UI 2.0 for the Nsis installer. (Guopeng Wen)
    https://github.com/gpwen/vim-installer-mui2
- make it possible to do a silent install, see
    http://nsis.sourceforge.net/Docs/Chapter4.html#4.12
    Version from Guopeng Wen does this.
- MSI installer: https://github.com/petrkle/vim-msi/
- The one on Issue 279.
Problem: they all work slightly different (e.g. don't install vimrun.exe).
How to test that it works well for all Vim users?

Patch to make fold updates much faster. (Christian Brabandt, 2012 Dec)

-   Add regex for 'paragraphs' and 'sections': 'parare' and 'sectre'. Combine
    the two into a regex for searching. (Ned Konz)
Patch by Christian Brabandt, 2013 Apr 20, unfinished.

Bug: findfile("any", "file:///tmp;") does not work.

In the ATTENTION message about an existing swap file, mention the name of the
process that is running. It might actually be some other program, e.g. after
a reboot.

patch to add "combine" flag to syntax commands. (so8res, 2012 Dec 6)

Syntax update problem in one buffer opened in two windows, bottom window is
not correctly updated. (Paul Harris, 2012 Feb 27)

Patch to add getsid(). (Tyru, 2011 Oct 2) Do we want this? Update Oct 4.
Or use expand('<sid>')?

Patch to make confirm() display colors. (Christian Brabandt, 2012 Nov 9)

Patch to add functions for signs. (Christian Brabandt, 2013 Jan 27)

Patch to remove flicker from popup menu. (Yasuhiro Matsumoto, 2013 Aug 15)

Problem with refresh:always in completion. (Tyler Wade, 2013 Mar 17)

b:undo_ftplugin cannot call a script-local function. (Boris Danilov, 2013 Jan
7)

Win32: The Python interface only works with one version of Python, selected at
compile time. Can this be made to work with version 2.1 and 2.2 dynamically?

Python: Be able to define a Python function that can be called directly from
Vim script. Requires converting the arguments and return value, like with
vim.bindeval().

Patch for :tabcloseleft, after closing a tab go to left tab. (William Bowers,
2012 Aug 4)

Patch to improve equivalence classes in regexp patterns.
(Christian Brabandt, 2013 Jan 16, update Jan 17)

Patch to add new regexp classes :ident:, :keyword:, :fname:.

(ichizok, 2016 Jan 12, #1373)

Patch with suggestions for starting.txt. (Tony Mechelynck, 2012 Oct 24)
But use Gnome instead of GTK?

Should be possible to enable/disable matchparen per window or buffer.
Add a check for b:no_match_paren in Highlight_matching_Pair() (Marcin
Szamotulski, 2012 Nov 8)

Session file creation: 'autochdir' causes trouble. Keep it off until after
loading all files.

MS-Windows resizing problems:
- Windows window on screen positioning: Patch by Yukihiro Nakadaira, 2012 Jun
  20. Uses getWindowRect() instead of GetWindowPlacement()
- Win32: When the taskbar is at the top of the screen creating the tabbar
  causes the window to move unnecessarily. (William E. Skeith III, 2012 Jan
  12) Patch: 2012 Jan 13 Needs more work (2012 Feb 2)

'iminsert' global value set when using ":setlocal iminsert"? (Wu, 2012 Jun 23)

Patch to append regexp to tag commands to make it possible to select one out
of many matches. (Cody Cutler, 2013 Mar 28)

The input map for CTRL-O in mswin.vim causes problems after CTRL-X CTRL-O.
Suggestion for another map. (Philip Mat, 2012 Jun 18)
But use "gi" instead of "a". Or use CTRL-\ CTRL-O.

Patch to support user name completion on MS-Windows. (Yasuhiro Matsumoto, 2012
Aug 16)

When there are no command line arguments ":next" and ":argu" give E163, which
is confusing. Should say "the argument list is empty".

URXVT:
- will get stuck if byte sequence does not contain the expected semicolon.
- Use urxvt mouse support also in xterm. Explanations:
  http://www.midnight-commander.org/ticket/2662

Patch to have the fold and sign column and at the last line of the buffer.
(Marco Hinz, 2014 Sep 25)
Alternate suggestion: let all columns continue, also the number column.

Patch to add tests for if_xcmdsrv.c., Jul 8, need some more work. (Brian Burns)
New tests Jul 13. Update Jul 17. Discussion Jul 18.

When running Vim in silent ex mode, an existing swapfile causes Vim to wait
for a user action without a prompt. (Maarten Billemont, 2012 Feb 3)
Do give the prompt? Quit with an error?

Patch to list user digraphs. (Christian Brabandt, 2012 Apr 14)

Patch to add digraph() function. (Christian Brabandt, 2013 Aug 22, update Aug
24)

Patch for input method status. (Hirohito Higashi, 2012 Apr 18)

Update Vim app icon (for Gnome). (Jakub Steiner, 2013 Dec 6)

Patch to use .png icons for the toolbar on MS-Windows. (Martin Gieseking, 2013
Apr 18)

Patch for has('unnamedplus') docs. (Tony Mechelynck, 2011 Sep 27)
And one for gui_x11.txt.

":cd" doesn't work when current directory path contains "**".
finddir() has the same problem. (Yukihiro Nakadaira, 2012 Jan 10)
Requires a rewrite of the file_file_in_path code.

Should use has("browsefilter") in ftplugins. Requires patch 7.3.593.

Update for vim2html.pl. (Tyru, 2013 Feb 22)

Patch to sort functions starting with '<' after others. Omit dict functions,
they can't be called. (Yasuhiro Matsumoto, 2011 Oct 11)

Patch to pass list to or(), and() and xor(). (Yasuhiro Matsumoto, 2012 Feb 8)

Patch to improve "it" and "at" text object matching. (Christian Brabandt, 2011
Nov 20)

Patch to improve GUI find/replace dialog. (Christian Brabandt, 2012 May 26)
Update Jun 2.

`] moves to character after insert, instead of the last inserted character.
(Yukihiro Nakadaira, 2011 Dec 9)

Plugin for Modeleasy. (Massimiliano Tripoli, 2011 Nov 29)

BufWinLeave triggers too late when quitting last window in a tab page. (Lech
Lorens, 2012 Feb 21)

Patch for 'transparency' option. (Sergiu Dotenco, 2011 Sep 17)
Only for MS-Windows. No documentation. Do we want this?

Patch to support cursor shape in Cygwin console. (Ben bgold, 2011 Dec 27)

On MS-Windows a temp dir with a & init causes system() to fail. (Ben Fritz,
2012 Jun 19)

'cursorline' is displayed too short when there are concealed characters and
'list' is set. (Dennis Preiser)
Patch 7.3.116 was the wrong solution.
Christian Brabandt has another incomplete patch. (2011 Jul 13)

With concealed text mouse click doesn't put the cursor in the right position.
(Herb Sitz) Fix by Christian Brabandt, 2011 Jun 16. Doesn't work properly,
need to make the change in where RET_WIN_BUF_CHARTABSIZE() is called.

Syntax region with 'concealends' and a 'cchar' value, 'conceallevel' set to 2,
only one of the two ends gets the cchar displayed. (Brett Stahlman, 2010 Aug
21, Ben Fritz, 2010 Sep 14)

The :syntax cchar value can only be a single character. It would be useful to
support combining characters. (Charles Campbell)

'cursorline' works on a text line only. Add 'cursorscreenline' for
highlighting the screen line. (Christian Brabandt, 2012 Mar 31)

Win32: Patch to use task dialogs when available. (Sergiu Dotenco, 2011 Sep 17)
New feature, requires testing. Made some remarks.

Win32: Patch for alpha-blended icons and toolbar height. (Sergiu Dotenco, 2011
Sep 17) Asked for feedback from others.

Win32: Cannot cd into a directory that starts with a space. (Andy Wokula, 2012 Jan 19)

Need to escape $HOME on Windows for fnameescape()? (ZyX, 2011 Jul 21, discussion 2013 Jul 4) Can't simply use a backslash, \$HOME has a different meaning already. Would be possible to use $$HOME where $HOME is to be used.

"2" in 'formatoptions' not working in comments. (Christian Corneliussen, 2011 Oct 26)

Bug in repeating Visual "u". (Lawrence Kesteloot, 2010 Dec 20)

With "unamedplus" in 'clipboard' pasting in Visual mode causes error for empty register. (Michael Seiwald, 2011 Jun 28) I can't reproduce it.

Windows keys not set properly on Windows 7? (cncyber, 2010 Aug 26)

When using a Vim server, a # in the path causes an error message.
(Jeff Lanzarotta, 2011 Feb 17)

When there is a ">" in a line that "gq" wraps to the start of the next line, then the following line will pick it up as a leader. Should get the leader from the first line, not a wrapped line. (Matt Ackeret, 2012 Feb 27)

Using ":break" or something else that stops executing commands inside a ":finally" does not rethrow a previously uncaught exception. (ZyX, 2010 Oct 15)

Vim using lots of memory when joining lines. (John Little, 2010 Dec 3)

BT regexp engine: After trying a \@> match and failing, submatches are not cleared. See test64.

Changes to manpage plugin. (Elias Toivanen, 2011 Jul 25)

Patch to make "z=" work when 'spell' is off. Does this have nasty side effects? (Christian Brabandt, 2012 Aug 5, Update 2013 Aug 12)
Would also need to do this for spellbadword() and spellsuggest().

On 64 bit MS-Windows "long" is only 32 bits, but we sometimes need to store a 64 bits value. Change all number options to use nropt_T and define it to the right type.

string() can't parse back "inf" and "nan". Fix documentation or fix code?
(ZyX, 2010 Aug 23)

When doing "redir => s:foo" in a script and then "redir END" somewhere else (e.g. in a function) it can't find s:foo.
When a script contains "redir => s:foo" but doesn't end redirection, a following "redir" command gives an error for not being able to access s:foo.
(ZyX, 2011 Mar 27)

When setqflist() uses a filename that triggers a BufReadCmd autocommand Vim doesn't jump to the correct line with :cfirst. (ZyX, 2011 Sep 18)

Behavior of i" and a" text objects isn't logical. (Ben Fritz, 2013 Nov 19)

7 Make "ga" show the digraph for a character, if it exists.
Patch from Christian Brabandt, 2011 Aug 19.

maparg() does not show the <script> flag. When temporarily changing a

mapping, how to restore the script ID?

Bug in try/catch: return with invalid compare throws error that isn't caught.
(ZyX, 2011 Jan 26)

When setting a local option value from the global value, add a script ID that
indicates this, so that ":verbose set" can give a hint. Check with options in
the help file.

After patch 7.3.097 still get E15. (Yukihiro Nakadaira, 2011 Jan 18)
Also for another example (ZyX, 2011 Jan 24)

Build problem with small features on Mac OS X 10.6. (Rainer, 2011 Jan 24)

"0g@$" puts '] on last byte of multi-byte. (ZyX, 2011 Jan 22)

Patch to add TextDeletePost and TextYankPost events. (Philippe Vaucher, 2011
May 24) Update May 26.

Patch for :tabrecently. (Hirokazu Yoshida, 2012 Jan 30)

Problem with "syn sync grouphere". (Gustavo Niemeyer, 2011 Jan 27)

Loading autoload script even when usage is inside "if 0". (Christian Brabandt,
2010 Dec 18)

With a filler line in diff mode, it isn't displayed in the column with line
number, but it is in the sign column. Doesn't look right. (ZyX 2011 Jun 5)
Patch by Christian Brabandt, 2011 Jun 5. Introduces new problems.

Add jump() function. (Marcin Szamotulski, 2013 Aug 29)
Is this needed? CTRL-O and CTRL-I do the same, just more difficult to use.

8   Add a command to jump to the next character highlighted with "Error".
Patch by Christian Brabandt, uses]e [e]t and [t. 2011 Aug 9.

Add event for when the text scrolls. A bit like CursorMoved. Also a similar
one for insert mode. Use the event in matchparen to update the highlight if
the match scrolls into view.

7   Use "++--", "+++--" for different levels instead of "+---" "+----".
Patch by Christian Brabandt, 2011 Jul 27.
Update by Ben Fritz, with fix for TOhtml. (2011 Jul 30)

9   Add %F to 'errorformat': file name without spaces. Useful on Unix to
    avoid matching something up to a time 11:22:33.
Patch by Christian Brabandt, 2011 Jul 27.

Patch to add up to 99 match groups. (Christian Brabandt, 2010 Dec 22)
Also add named groups: \%{name}(re) and \%{name}g

In the sandbox it's not allowed to do many things, but it's possible to change
or set variables. Add a way to prevent variables from being changed in the
sandbox? E.g.: ":protect g:restore_settings".

GTK: drawing a double-width combining character over single-width characters
doesn't look right. (Dominique Pelle, 2010 Aug 8)

GTK: tear-off menu does not work. (Kurt Sonnenmoser, 2010 Oct 25)

Win32: tear-off menu does not work when menu language is German. (Markus
Bossler, 2011 Mar 2) Fixed by 7.3.095?

```
Wish for NetBeans commands:
    - make it possible to have 'defineAnnoType' also handle terminal colors.

Version of netbeans.c for use with MacVim. (Kazuki Sakamoto, 2010 Nov 18)

7.3.014 changed how backslash at end of line works, but still get a NUL when
there is one backslash. (Ray Frush, 2010 Nov 18)  What does the original ex
do?

Searching mixed with Visual mode doesn't redraw properly. (James Vega, 2010 Nov
22)

New esperanto spell file can't be processed. (Dominique Pelle, 2011 Jan 30)
- move compflags to separate growarray?
- instead of a regexp use a hashtable.  Expand '?', '*', '+'.  What would be
  the maximum repeat for * and +?

"L'Italie" noted as a spell error at start of the sentence. (Dominique Pelle,
2011 Feb 27)

Editing a file with a ^M with 'ff' set to "mac", opening a help file, then the
^M is displayed as ^J sometimes.  Getting 'ff' value from wrong window/buffer?

When Vim is put in the background (SIGTSTP) and then gets a SIGHUP it doesn't
exit.  It exists as soon as back in the foreground. (Stephen Liang, 2011 Jan
9)  Caused by vim_handle_signal(SIGNAL_BLOCK); in ui.c.

g` not working correctly when using :edit.  It works OK when editing a file on
the command line. (Ingo Karkat, 2011 Jan 25)

Since patch 7.2.46 Yankring plugin has become very slow, eventually make Vim
crash? (Raiwil, 2010 Nov 17)

Patch to add FoldedLineNr highlighting: different highlighting for the line
number of a closed fold. (eXerigumo Clanjor, 2013 Jul 15)

Regexp engine performance:
- Profiling:
        ./vim -u NONE -s ~/vim/test/ruby.vim
        ./vim -u NONE -s ~/vim/test/loop.vim
        ./vim -u NONE -s ~/vim/test/alsa.vim
        ./vim -s ~/vim/test/todo.vim
        ./vim -s ~/vim/test/xml.vim
    Dominique Pelle:  xmlSyncDT is particularly slow (Jun 7)
- More test files from the src/pkg/regexp/testdata directory in the Go repo.
- Performance tests:
  - Using asciidoc syntax. (Marek Schimara, 2013 Jun 6)
  - ~/vim/text/FeiqCfg.xml (file from Netjune)
  - ~/vim/text/edl.svg  (also XML)
  - glts has five tests. (May 25)
  - ~/vim/test/slowsearch
  - ~/vim/test/rgb.vim
  - search for  a.*e*exn  in the vim executable.  Go to last line to use
    'hlsearch'.
  - Slow combination of folding and PHP syntax highlighting.  Script to
    reproduce it.  Caused by "syntax sync fromstart" in combination with patch
    7.2.274.  (Christian Brabandt, 2010 May 27) Generally, folding with
    'foldmethod' set to "syntax" is slow.  Do profiling to find out why.

Problem producing tags file when hebrew.frx is present.  It has a BOM.
Results in E670. (Tony Mechelynck, 2010 May 2)
```

'beval' option should be global-local.

Ruby: ":ruby print $buffer.number" returns zero.

setpos() does not restore cursor position after :normal. (Tyru, 2010 Aug 11)

7 The 'directory' option supports changing path separators to "%" to make
file names unique, also support this for 'backupdir'. (Mikolaj Machowski)
Patch by Christian Brabandt, 2010 Oct 21.
Is this an update: related to: #179
https://github.com/chrisbra/vim-mq-patches/blob/master/backupdir
Fixed patch 2017 Jul 1.

With "tw=55 fo+=a" typing space before) doesn't work well. (Scott Mcdermott, 2010 Oct 24)

Messages in message.txt are highlighted as examples.

When using cp850 the NBSP (0xff) is not drawn correctly. (Brett Stahlman, 2010 Oct 22) 'isprint' is set to "@,161-255".

":echo "\x85" =~# '[\u0085]'" returns 1 instead of 0. (ZyX, 2010 Oct 3)

'cindent' not correct when 'list' is set. (Zdravi Korusef, 2010 Apr 15)

C-indenting: A matching { in a comment is ignored, but intermediate { are not checked to be in a comment. Implement FM_SKIPCOMM flag of findmatchlimit(). Issue 46.

Mac with X11: clipboard doesn't work properly. (Raf, 2010 Aug 16)

Using CompilerSet doesn't record where an option was set from. E.g., in the gcc compiler plugin. (Gary Johnson, 2010 Dec 13)

":helpgrep" does not put the cursor in the correct column when preceded by accented character. (Tony Mechelynck, 2010 Apr 15)

Don't call check_restricted() for histadd(), setbufvar(), settabvar(), setwinvar().

Patch for GVimExt to show an icon. (Dominik Riebeling, 2010 Nov 7)

When 'lines' is 25 and 'scrolloff' is 12, "j" scrolls zero or two lines instead of one. (Constantin Pan, 2010 Sep 10)

Gui menu edit/paste in block mode insert only inserts in one line (Bjorn Winckler, 2011 May 11)
Requires a map mode for Insert mode started from blockwise Visual mode.

Writing nested List and Dict in viminfo gives error message and can't be read back. (Yukihiro Nakadaira, 2010 Nov 13)

Problem with cursor in the wrong column. (SungHyun Nam, 2010 Mar 11)
Additional info by Dominique Pelle. (also on 2010 Apr 10)

CreateFile and CreateFileW are used without sharing, filewritable() fails when the file was already open (e.g. script is being sourced). Add FILE_SHARE_READ| FILE_SHARE_WRITE in mch_access()? (Phillippe Vaucher, 2010 Nov 2)

Is ~/bin (literally) in $PATH supposed to work? (Paul, 2010 March 29)
Looks like only bash can do it. (Yakov Lerner)

Cscope "cs add" stopped working somewhat before 7.2.438. (Gary Johnson, 2010
Jun 29) Caused by 7.2.433?

I often see pasted text (from Firefox, to Vim in xterm) appear twice.
Also, Vim in xterm sometimes loses copy/paste ability (probably after running
an external command).

Jumplist doesn't work properly in Insert mode? (Jean Johner, 2010 Mar 20)

Problem with transparent cmdline. Also: Terminal title is wrong with
non-ASCII character. (Lily White, 2010 Mar 7)

iconv() doesn't fail on an illegal character, as documented. (Yongwei Wu, 2009
Nov 15, example Nov 26) Add argument to specify whether iconv() should fail
or replace with a character and continue?

Add local time at start of --startuptime output.
Requires configure check for localtime().
Use format year-month-day hr:min:sec.

Patch to add "combine" to :syntax, combines highlight attributes. (Nate
Soares, 2012 Dec 3)

Patch to make ":hi link" also take arguments. (Nate Soares, 2012 Dec 4)

Shell not recognized properly if it ends in "csh -f". (James Vega, 2009 Nov 3)
Find tail? Might have a / in argument. Find space? Might have space in
path.

Test 51 fails when language set to German. (Marco, 2011 Jan 9)
Dominique can't reproduce it.

'ambiwidth' should be global-local.

":function f(x) keepjumps" creates a function where every command is executed
like it has ":keepjumps" before it.

Coverity: Check if there are new reported defects:
https://scan.coverity.com/projects/241

Patch to support :undo absolute jump to file save number. (Christian Brabandt,
2010 Nov 5)

Patch to use 'foldnestmax' also for "marker" foldmethod. (Arnaud Lacombe, 2011
Jan 7)

Bug with 'incsearch' going to wrong line. (Wolfram Kresse, 2009 Aug 17)
Only with "vim -u NONE".

Problem with editing file in binary mode. (Ingo Krabbe, 2009 Oct 8)

With 'wildmode' set to "longest:full,full" and pressing Tab once the first
entry in wildmenu is highlighted, that shouldn't happen. (Yuki Watanabe, 2011
Feb 12)

Display error when 'tabline' that includes a file name with double-width
characters. (2010 Aug 14, bootleq)

Problem with stop directory in findfile(). (Adam Simpkins, 2009 Aug 26)

Using ']' as the end of a range in a pattern requires double escaping:

```
     /[@-\\]]  (Andy Wokula, 2011 Jun 28)

Syntax priority problem. (Charles Campbell, 2011 Sep 15)

When completion inserts the first match, it may trigger the line to be folded.
Disable updating folds while completion is active? (Peter Odding, 2010 Jun 9)

When a:base in 'completefunc' starts with a number it's passed as a number,
not a string. (Sean Ma)  Need to add flag to call_func_retlist() to force a
string value.

For running gvim on a USB stick: avoid the OLE registration.  Use a command
line argument -noregister.

When using an expression in 'statusline' leading white space sometimes goes
missing (but not always). (ZyX, 2010 Nov 1)

When a mapping exists both for insert mode and lang-insert mode, the last one
doesn't work. (Tyru, 2010 May 6)  Or is this intended?

Still a problem with ":make" in the wrong directory.  Caused by ":bufdo".
(Ajit Thakkar, 2009 Jul 1) More information Jul 9, Jul 15.
Caused by "doautoall syntaxset BufEnter *" in syntax/nosyntax.vim ?
There also is a BufLeave/BufEnter aucmd to save/restore view.
Does the patch to save/restore globaldir work?

":bufdo normal gg" while 'hidden' is set leaves buffers without syntax
highlighting.  Don't disable Syntax autocommands then?  Or add a flag/modifier
to avoid changing 'eventignore'?

Patch for displaying 0x200c and 0x200d. (Ali Gholami Rudi, 2009 May 6)
Probably needs a bit of work.

Patch to add farsi handling to arabic.c (Ali Gholami Rudi, 2009 May 2)
Added test, updates, June 23.
Updated for 7.4: http://litcave.rudi.ir/farsi_vim.diff
With modification for Tatweel character: https://dpaste.de/VmFw
Remark from Ameretat Reith (2014 Oct 13)

List of encoding aliases. (Takao Fujiwara, 2009 Jul 18)
Are they all OK?  Update Jul 22.

Win32: Improved Makefile for MSVC. (Leonardo Valeri Manera, 2010 Aug 18)

Win32: Expanding 'path' runs into a maximum size limit. (bgold12, 2009 Nov 15)

Win32: Patch for enabling quick edit mode in console. (Craig Barkhouse, 2010
Sep 1)

Win32: Patch for using .png files for icons. (Charles Peacech, 2012 Feb 5)

Putting a Visual block while 'visualedit' is "all" does not leave the cursor
on the first character. (John Beckett, 2010 Aug 7)

Setting 'tags' to "tagsdir/*" does not find "tagsdir/tags". (Steven K. Wong,
2009 Jul 18)

Patch to add "focusonly" to 'scrollopt', so that scrollbind also applies in
window that doesn't have focus. (Jonathon Mah, 2009 Jan 12)
Needs more work.

Problem with <script> mappings (Andy Wokula, 2009 Mar 8)
```

When starting Vim with "gvim -f -u non_existent_file > foo.txt" there are a
few control characters in the output. (Dale Wiles, 2009 May 28)

'cmdwinheight' is only used in last window when 'winheight' is a large value.
(Tony Mechelynck, 2009 Apr 15)

Status line containing winnr() isn't updated when splitting the window (Clark
J. Wang, 2009 Mar 31)

When $VIMRUNTIME is set in .vimrc, need to reload lang files. Already done
for GTK, how about others? (Ron Aaron, 2010 Apr 10)

Patch for GTK buttons X1Mouse and X2Mouse. (Christian J. Robinson, 2010 Aug 9)

Motif: Build on Ubuntu can't enter any text in dialog text fields.

":tab split fname" doesn't set the alternate file in the original window,
because win_valid() always returns FALSE. Below win_new_tabpage() in
ex_docmd.c.

Space before comma in function definition not allowed: "function x(a , b)"
Give a more appropriate error message. Add a remark to the docs.

string_convert() should be able to convert between utf-8 and utf-16le. Used
for GTK clipboard. Avoid requirement for iconv.

Now that colnr_T is int instead of unsigned, more type casts can be removed.

'delcombine' does not work for the command line. (Tony Mechelynck, 2009 Jul
20)

Don't load macmap.vim on startup, turn it into a plugin. (Ron Aaron,
2009 Apr 7) Reminder Apr 14.

Add "no_hlsearch" to winsaveview().

Cursorline highlighting combines with Search ('hlsearch') but not with
SpellBad. (Jim Karsten, 2009 Mar 18)

When 'foldmethod' is "indent", adding an empty line below a fold and then
indented text, creates a new fold instead of joining it with the previous one.
(Evan Laforge, 2009 Oct 17)

Bug: When reloading a buffer changed outside of Vim, BufRead autocommands
are applied to the wrong buffer/window. (Ben Fritz, 2009 Apr 2, May 11)
Ignore window options when not in the right window?
Perhaps we need to use a hidden window for applying autocommands to a buffer
that doesn't have a window.

When using "ab foo bar" and mapping <Tab> to <Esc>, pressing <Tab> after foo
doesn't trigger the abbreviation like <Esc> would. (Ramana Kumar, 2009 Sep 6)

getbufvar() to get a window-local option value for a buffer that's not
displayed in a window should return the value that's stored for that buffer.

":he ctrl_u" can be auto-corrected to ":he ctrl-u".

There should be a way after an abbreviation has expanded to go back to what
was typed. CTRL-G h ? Would also undo last word or line break inserted
perhaps. And undo CTRL-W. CTRL-G l would redo.

Diff mode out of sync. (Gary Johnson, 2010 Aug 4)

Win32 GUI: last message from startup doesn't show up when there is an echoerr command. (Cyril Slobin, 2009 Mar 13)

Win32: completion of file name ":e c:\!test" results in ":e c:\\!test", which does not work. (Nieko Maatjes, 2009 Jan 8, Ingo Karkat, 2009 Jan 22)

opening/closing window causes other window with 'winfixheight' to change height. Also happens when there is another window in the frame, if it's not very high. (Yegappan Lakshmanan, 2010 Jul 22, Michael Peeters, 2010 Jul 22)

Directory wrong in session file, caused by ":lcd" in BufEnter autocommand. (Felix Kater, 2009 Mar 3)

Session file generates error upon loading, cause by --remote-silent-tab. (7tommm (ytommm) 2010 Nov 24)

Using ~ works OK on 'a' with composing char, but not on 0x0418 with composing char 0x0301. (Tony Mechelynck, 2009 Mar 4)

Searching for composing char works, but not when inside []. (ZyX, Benjamin R. Haskell, 2010 Aug 24)

This does not work yet: "a\(%C\)" (get composing characters into a submatch).

A function on a dictionary is not profiled. (ZyX, 2010 Dec 25)

Inconsistent: starting with $LANG set to es_ES.utf-8 gives Spanish messages, even though locale is not supported. But ":lang messages es_ES.utf-8" gives an error and doesn't switch messages. (Dominique Pelle, 2009 Jan 26)

When $HOME contains special characters, such as a comma, escape them when used in an option. (Michael Hordijk, 2009 May 5)
Turn "esc" argument of expand_env_esc() into string of chars to be escaped.

Should make 'ignorecase' global-local, so that it makes sense setting it from a modeline.

Add cscope target to Makefile. (Tony Mechelynck, 2009 Jun 18, replies by Sergey Khorev)

Consider making YankRing or something else that keeps a list of yanked text part of standard Vim. The "1 to "9 registers are not sufficient.

After doing "su" $HOME can be the old user's home, thus ~root/file is not correct. Don't use it in the swap file.

Completion for ":buf" doesn't work properly on Win32 when 'shellslash' is off. (Henrik Ohman, 2009, Jan 29)

shellescape() depends on 'shellslash' for quoting. That doesn't work when 'shellslash' is set but using cmd.exe. (Ben Fritz)
Use a different option or let it depend on whether 'shell' looks like a unix-like shell?

Bug: in Ex mode (after "Q") backslash before line break, when yanked into a register and executed, results in <Nul>: instead of line break.
(Konrad Schwarz, 2010 Apr 16)

Have a look at patch for utf-8 line breaking. (Yongwei Wu, 2008 Mar 1, Mar 23)

Now at: http://vimgadgets.sourceforge.net/liblinebreak/

Greek sigma character should be lower cased depending on the context. Can we make this work? (Dominique Pelle, 2009 Sep 24)

When changing 'encoding' convert all the swap file names, so that we can still delete them. Also convert all buffer file names?

"gqip" in Insert mode has an off-by-one error, causing it to reflow text. (Raul Coronado, 2009 Nov 2)

Update src/testdir/main.aap.

Something wrong with session that has "cd" commands and "badd", in such a way that Vim doesn't find the edited file in the buffer list, causing the ATTENTION message? (Tony Mechelynck, 2008 Dec 1)
Also: swap files are in ~/tmp/ One has relative file name ".mozilla/...".

Add v:motion_force. (Kana Natsuno, 2008 Dec 6)
Maybe call it v:motiontype.

MS-Windows: editing the first, empty buffer, 'ffs' set to "unix,dos", ":enew" doesn't set 'ff' to "unix". (Ben Fritz, 2008 Dec 5) Reusing the old buffer probably causes this.

'scrollbind' is not respected when deleting lines or undo. (Milan Vancura, 2009 Jan 16)

Document that default font in Athena can be set with resources:
   XtDefaultFont: "9x15"
   XtDefaultFontSet: "9x15"
(Richard Sherman, 2009 Apr 12)

Having "Syntax" in 'eventignore' for :bufdo may cause problems, e.g. for ":bufdo e" when buffers are open in windows. ex_listdo(eap) could set the option only for when jumping to another buffer, not when the command argument is executed.

":pedit %" with a BufReadPre autocommand causes the cursor to move to the first line. (Ingo Karkat, 2008 Jul 1) Ian Kelling is working on this.
Similar problem with ":e". (Marc Montu, 2014 Apr 22)

Wildmenu not deleted: "gvim -u NONE", ":set nocp wildmenu cmdheight=3 laststatus=2", CTRL-D CTRL-H CTRL-H CTRL-H. (A.Politz, 2008 April 1)
Works OK with Vim in an xterm.

Cursor line moves in other window when using CTRL-W J that doesn't change anything. (Dasn, 2009 Apr 7)

On Unix "glob('does not exist~')" returns the string. Without the "~" it doesn't. (John Little, 2008 Nov 9)
Shell expansion returns unexpanded string?
Don't use shell when "~" is not at the start?

When using ":e ++enc=foo file" and the file is already loaded with 'fileencoding' set to "bar", then do_ecmd() uses that buffer, even though the fileencoding differs. Reload the buffer in this situation? Need to check for the buffer to be unmodified.
Unfinished patch by Ian Kelling, 2008 Jul 11. Followup Jul 14, need to have another look at it.

c.vim: XXX in a comment is colored yellow, but not when it's after "#if 0".

(Ilya Dogolazky, 2009 Aug 7)

You can type ":w ++bad=x fname", but the ++bad argument is ignored. Give an error message? Or is this easy to implement? (Nathan Stratton Treadway, 2008 Aug 20) This is in ucs2bytes(), search for 0xBF. Using the ++bad argument is at the other match for 0xBF.

When adding "-complete=file" to a user command this also changes how the argument is processed for <f-args>. (Ivan Tishchenko, 2008 Aug 19)

Win32: associating a type with Vim doesn't take care of space after a backslash? (Robert Vibrant, 2008 Jun 5)

When 'rightleft' is set, cursorcolumn isn't highlighted after the end of a line. It's also wrong in folds. (Dominique Pelle, 2010 Aug 21)

Using an insert mode expression mapping, cursor is not in the expected position. (ZyX, 2010 Aug 29)

After using <Tab> for command line completion after ":ta blah" and getting E33 (no tags file), further editing the command to e.g., ":echo 'blah'", the command is not executed. Fix by Ian Kelling?

":help s/~" jumps to *s/\~*, while ":help s/\~" doesn't find anything. (Tim Chase) Fix by Ian Kelling, 2008 Jul 14.

When mapping : to ; and ; to :, @; doesn't work like @: and @: doesn't work either. Matt Wozniski: nv_at() calls do_execreg() which uses put_in_typebuf(). Char mapped twice?

Despite adding save_subexpr() this still doesn't work properly:
Regexp: matchlist('12a4aaa', '^\(.\{-}\)\(\%5c\@<=a\+\)\(.\+\)\?')
Returns ['12a4', 'aaa', '4aaa'], should be ['12a4', 'aaa', '']
Backreference not cleared when retrying after \@<= fails?
(Brett Stahlman, 2008 March 8)

Problem with remote_send(). (Charles Campbell, 2008 Aug 12)

ftplugin for help file should set 'isk' to help file value.

Win32: remote editing fails when the current directory name contains "[". (Ivan Tishchenko, Liu Yubao) Suggested patch by Chris Lubinski: Avoid escaping characters where the backslash is not removed later. Asked Chris for an alternate solution, also for src/ex_getln.c.
This also fails when the file or directory name contains "%". (Thoml, 2008 July 7)
Using --remote-silent while the current directory has a # in the name does not work, the # needs to be escaped. (Tramblay Bruno, 2012 Sep 15)

When using remote-silent the -R flag is not passed on. (Axel Bender, 2012 May 31)

Win32: A --remote command that has a directory name starting with a (doesn't work, the backslash is removed, assuming that it escapes the (. (Valery Kondakoff, 2009 May 13)

Win32: Using "gvim --remote-tab-silent elŝuti.txt" doesn't work, the multi-byte character isn't passed and edits elsuti.txt.
(Raúl Núñez de Arenas Coronado, 2015 Dec 18)

Problem with 'langmap' being used on the rhs of a mapping. (Nikolai Weibull, 2008 May 14)

Possibly related problem: Alexey Muranov, 2015 Apr 2

Problem with CTRL-F. (Charles Campbell, 2008 March 21)
Only happens with "gvim -geometry "160x26+4+27" -u NONE -U NONE prop.c".
'lines' is 54. (2008 March 27)

Problem with pointer wrapping around in getvcol(). (Wolfgang Kroworsch, 2008
Oct 19) Check for "col" being "MAXCOL" separately?

Unexpectedly inserting a double quote. (Anton Woellert, 2008 Mar 23)
Works OK when 'cmdheight' is 2.

8   Use a mechanism similar to omni completion to figure out the kind of tab
    for CTRL-] and jump to the appropriate matching tag (if there are
    several).
    Alternative: be able to define a function that takes the tag name and uses
    taglist() to find the right location. With indication of using CTRL-] so
    that the context can be taken into account. (Robert Webb)
Patch by Christian Brabandt, 2013 May 31.

The utf class table is missing some entries:
        0x2212, minus sign
        0x2217, star
        0x2500, bar
        0x26ab, circle

Visual line mode doesn't highlight properly when 'showbreak' is used and the
line doesn't fit. (Dasn, 2008 May 1)

GUI: In Normal mode can't yank the modeless selection. Make "gy" do this?
Works like CTRL-Y in Command line mode.

Mac: Move Carbon todo items to os_mac.txt. Note that this version is frozen,
try the Cocoa version.

Mac: After a ":vsplit" the left scrollbar doesn't appear until 'columns' is
changed or the window is resized.

GTK: when setting 'columns' in a startup script and doing ":vertical diffsplit"
the window isn't redrawn properly, see two vertical bars.

Mac: Patch for configure: remove arch from ruby link args. (Knezevic, 2008
Mar 5) Alternative: Kazuki Sakamoto, Mar 7.

Mac: trouble compiling with Motif, requires --disable-darwin. (Raf, 2008 Aug
1) Reply by Ben Schmidt.

C't: On utf-8 system, editing file with umlaut through Gnome results in URL
with %nn%nn, which is taken as two characters instead of one.
Try to reproduce at work.

Patch for default choice in file changed dialog. (Bjorn Winckler, 2008 Oct 19)
Is there a way to list all the files first?

When 'smartcase' is set and using CTRL-L to add to the search pattern it may
result in no matches. Convert chars to lower case? (Erik Wognsen, 2009 Apr
16)

Fail to edit file after failed register access. Error flag remains set?
(Lech Lorens, 2010 Aug 30)

Patch for redo register. (Ben Schmidt, 2007 Oct 19)

Await response to question to make the register writable.

Problem with 'ts' set to 9 and 'showbreak' to ">>>". (Matthew Winn, 2007 Oct
1)

In the swapfile dialog, add a H(elp) option that gives more info about what
each choice does. Similar to ":help swap-exists-choices"

try/catch not working for argument of return. (Matt Wozniski, 2008 Sep 15)

try/catch not working when inside a for loop. (ZyX, 2011 Jan 25)

":tab help" always opens a new tab, while ":help" re-uses an existing window.
Would be more consistent when an existing tab is re-used. (Tony Mechelynck)

Add ":nofold". Range will apply without expanding to closed fold.

Using Aap to build Vim: add remarks about how to set personal preferences.
Example on http://www.calmar.ws/tmp/aap.html

Syntax highlighting wrong for transparent region. (Doug Kearns, 2007 Feb 26)
Bug in using a transparent syntax region. (Hanlen in vim-dev maillist, 2007
Jul 31)

C syntax: {} inside () causes following {} to be highlighted as error.
(Michalis Giannakidis, 2006 Jun 1)

When 'diffopt' has "context:0" a single deleted line causes two folds to merge
and mess up syncing. (Austin Jennings, 2008 Jan 31)

Gnome improvements: Edward Catmur, 2007 Jan 7
    Also use Save/Discard for other GUIs

New PHP syntax file, use it? (Peter Hodge)

":echoe" in catch block stops processing, while this doesn't happen outside of
a catch block. (ZyX, 2011 Jun 2)

'foldcolumn' in modeline applied to wrong window when using a session. (Teemu
Likonen, March 19)

Test 54 uses shell commands, that doesn't work on non-Unix systems. Use some
other way to test buffer-local autocommands.

The documentation mentions the priority for ":2match" and ":3match", but it
appears the last one wins. (John Beckett, 2008 Jul 22) Caused by adding
matchadd()? Suggested patch by John, 2008 Jul 24.

When 'encoding' is utf-8 the command line is redrawn as a whole on every
character typed. (Tyler Spivey, 2008 Sep 3) Only redraw cmdline for
'arabicshape' when there is a character on the command line for which
(ARABIC_CHAR(u8c)) is TRUE.

Cheng Fang made javacomplete. (2007 Aug 11)
Asked about latest version: 0.77.1 is on www.vim.org.

More AmigaOS4 patches. (Peter Bengtsson, Nov 9)

Amiga patches with vbcc. (Adrien Destugues, 2010 Aug 30)
http://pulkomandy.ath.cx/drop/vim73_vbcc_amiga.diff

Insert mode completion: When editing the text and pressing CTRL-N again goes

back to originally completed text, edited text is gone. (Peng Yu, 2008 Jul 24)
Suggestion by Ben Schmidt, 2008 Aug 6.

Problem with compound words? (Bert, 2008 May 6)
No warning for when flags are defined after they are used in an affix.

Screen redrawing when continuously updating the buffer and resizing the terminal. (Yakov Lerner, 2006 Sept 7)

Add option settings to help ftplugin. (David Eggum, 2006 Dec 18)

Autoconf problem: when checking for iconv library we may add -L/usr/local/lib, but when compiling further tests -liconv is added without the -L argument, that may fail (e.g., sizeof(int)). (Blaine, 2007 Aug 21)

When opening quickfix window, disable spell checking?

Problem with ".add" files when using two languages and restarting Vim. (Raul Coronado, 2008 Oct 30)

Popup menu redraw: Instead of first redrawing the text and then drawing the popup menu over it, first draw the new popup menu, remember its position and size and then redraw the text, skipping the characters under the popup menu. This should avoid flicker. Other solution by A.Politz, 2007 Aug 22.

When a register contains illegal bytes, writing viminfo in utf-8 and reading it back doesn't result in utf-8. (Devin Bayer)

Command line completion: Scanning for tags doesn't check for typed key now and then? Hangs for about 5 seconds. Appears to be caused by finding include files with "foo/**" in 'path'. (Kalisiak, 2006 July 15)
Additional info: When using the |wildcards| ** globing, vim hangs indefinitely on lots of directories. The |file-searching| globing, like in ":set path=/**" does not hang as often as with globing with |wildcards|, like in ":1find /**/file". This is for files that unix "find" can find very quickly. Merging the 2 kinds of globing might make this an easier fix. (Ian Kelling, 2008 July 4)

When the file name has parenthesis, e.g., "foo (bar).txt", ":!ls '%'" has the parenthesis escaped but not the space. That's inconsistent. Either escape neither or both. No escaping might be best, because it doesn't depend on particularities of the shell. (Zvi Har'El, 2007 Nov 10) (Teemu Likonen, 2008 Jun 3)
However, for backwards compatibility escaping might be necessary. Check if the user put quotes around the expanded item?

A throw in a function causes missing an endif below the call. (Spiros Bousbouras, 2011 May 16)

Error E324 can be given when a cron script has wiped out our temp directory. Give a clear error message about this (and tell them not to wipe out /tmp).

Color for cUserLabel should differ from case label, so that a mistake in a switch list is noticed:

```
    switch (i)
    {
    case 1:
    foobar:
    }
```

Look at http://www.gtk-server.org/ . It has a Vim script implementation.

Netbeans problem. Use "nc -l 127.0.0.1 55555" for the server, then run gvim
with "gvim -nb:localhost:55555:foo". From nc do: '1:editFile!0 "foo"'. Then
go to Insert mode and add a few lines. Then backspacing every other time
moves the cursor instead of deleting. (Chris Kaiser, 2007 Sep 25)

Windows installer could add a "open in new tab of existing Vim" menu entry.
Gvimext: patch to add "Edit with single Vim &tabbed" menu entry.
Just have two choices, always using one Vim and selecting between using an
argument list or opening each file in a separate tab.
(Erik Falor, 2008 May 21, 2008 Jun 26)

Windows installer: licence text should not use indent, causes bad word wrap.
(Benjamin Fritz, 2010 Aug 16)

Dos uninstal may delete vim.bat from the wrong directory (e.g., when someone
makes his own wrapper). Add a magic string with the version number to the
.bat file and check for it in the uninstaller. E.g.
          # uninstall key: vim7.3*

Changes for Win32 makefile. (Mike Williams, 2007 Jan 22, Alexei Alexandrov,
2007 Feb 8)

Win32: Can't complete shell command names. Why is setting xp_context in
set_one_cmd_context() inside #ifndef BACKSLASH_IN_FILENAME?

Win32: Patch for cscope external command. (Mike Williams, 2007 Aug 7)

Win32: XPM support only works with path without spaces. Patch by Mathias
Michaelis, 2006 Jun 9. Another patch for more path names, 2006 May 31.
New version: http://members.tcnet.ch/michaelis/vim/patches.zip (also for other
patches by Mathias, see mail Feb 22)

Win32: compiling with normal features and OLE fails. Patch by Mathias
Michaelis, 2006 Jun 4.

Win32: after "[I" showing matches, scroll wheel messes up screen. (Tsakiridis,
2007 Feb 18)
Patch by Alex Dobrynin, 2007 Jun 3. Also fixes other scroll wheel problems.

Win32: using CTRL-S in Insert mode doesn't remove the "+" from the tab pages
label. (Tsakiridis, 2007 Feb 18) Patch from Ian Kelling, 2008 Aug 6.

Win32: using "gvim --remote-tab-silent fname" sometimes gives an empty screen
with the more prompt. Caused by setting the guitablabel? (Thomas Michael
Engelke, 2007 Dec 20 - 2008 Jan 17)

Win32: patch for fullscreen mode. (Liushaolin, 2008 April 17)

Win32: When 'shell' is bash shellescape() doesn't always do the right thing.
Depends on 'shellslash', 'shellquote' and 'shellxquote', but shellescape()
only takes 'shellslash' into account.

Menu item that does "xxd -r" doesn't work when 'fileencoding' is utf-16.
Check for this and use iconv? (Edward L. Fox, 2007 Sep 12)
Does the conversion in the other direction work when 'fileencodings' is set
properly?

Cursor displayed in the wrong position when using 'numberwidth'. (James Vega,
2007 Jun 21)

When $VAR contains a backslash expand('$VAR') removes it. (Teemu Likonen, 2008
Jun 18)

If the variable "g:x#y#z" exists completion after ":echo g:x#" doesn't work.

Feature request: Command to go to previous tab, like what CTRL-W p does for
windows. (Adam George)

F1 - F4 in an xterm produce a different escape sequence when used with a
modifier key. Need to catch three different sequences. Use K_ZF1, like
K_ZHOME? (Dickey, 2007 Dec 2)

In debug mode, using CTRL-R = to evaluate a function causes stepping through
the function. (Hari Krishna Dara, 2006 Jun 28)

C++ indenting wrong with "=". (James Kanze, 2007 Jan 26)

":lockvar" should use copyID to avoid endless loop.

When using --remote-silent and the file name matches 'wildignore' get an E479
error. without --remote-silent it works fine. (Ben Fritz, 2008 Jun 20)

Gvim: dialog for closing Vim should check if Vim is busy writing a file. Then
use a different dialog: "busy saving, really quit? yes / no".

Check other interfaces for changing curbuf in a wrong way. Patch like for
if_ruby.c.

":helpgrep" should use the directory from 'helpfile'.

The need_fileinfo flag is messy. Instead make the message right away and put
it in keep_msg?

Editing a file remotely that matches 'wildignore' results in a "no match"
error. Should only happen when there are wildcards, not when giving the file
name literally, and esp. if there is only one name.

Test 61 fails sometimes. This is a timing problem: "sleep 2" sometimes takes
longer than 2 seconds.

Using ":au CursorMoved * cmd" invokes mch_FullName(), which can be slow.
Can this be avoided? (Thomas Waba, 2008 Aug 24)
Also for ":w" without a file name.
The buffer has the full path in ffname, should pass this to the autocommand.

"vim -C" often has 'nocompatible', because it's set in some startup script.
Set 'compatible' after startup is done? Patch by James Vega, 2008 Feb 7.

VMS: while editing a file found in complex, Vim will save file into the first
directory of the path and not to the original location of the file.
(Zoltan Arpadffy)

VMS: VFC files are in some cases truncated during reading (Zoltan Arpadffy)

input() completion should not insert a backslash to escape a space in a file
name?

Ruby completion is insecure. Can this be fixed?

When 'backupskip' is set from $TEMP special characters need to be escaped.
(patch by Grembowietz, 2007 Feb 26, not quite right)
Another problem is that file_pat_to_reg_pat() doesn't recognize "\\", so "\\("
will be seen as a path separator plus "\(".

gvim d:\path\path\(FILE).xml should not remove the \ before the (.
This also fails with --remote.

When doing ":quit" the Netbeans "killed" event isn't sent. (Xavier de Gaye, 2008 Nov 10) call netbeans_file_closed() at the end of buf_freeall(), or in all places where buf_freeall() is called?

aucmd_prepbuf() should also use a window in another tab page.

When unloading a buffer in a BufHidden autocommand the hidden flag is reset? (Bob Hiestand, 2008 Aug 26, Aug 27)

Substituting an area with a line break with almost the same area does change the Visual area. Can this be fixed? (James Vega, 2006 Sept 15)

GUI: When combining fg en bg make sure they are not equal.

Spell checking: Add a way to specify punctuation characters. Add the superscript numbers by default: 0x2070, 0xb9, 0xb2, 0xb3, 0x2074 - 0x2079.

Spell checking in popup menu: If the only problem is the case of the first character, don't offer "ignore" and "add to word list".

Use different pt_br dictionary for spell checking. (Jackson A. Aquino, 2006 Jun 5)

Use different romanian dictionary for spell checking. (Andrei Popescu, Nov 2008) Use http://downloads.sourceforge.net/rospell/ro_RO.3.2.zip
Or the hunspell-ro.3.2.tar.gz file, it also has a iso-8859-2 list.

In a C file with spell checking, in "% integer" "nteger" is seen as an error, but "]s" doesn't find it. "nteger" by itself is found. (Ralf Wildenhues, 2008 Jul 22)

There should be something about spell checking in the user manual.

Spell menu: When using the Popup menu to select a replacement word, ":spellrepeat" doesn't work. SpellReplace() uses setline(). Can it use "z=" somehow? Or use a new function.

Mac: Using gvim: netrw window disappears. (Nick Lo, 2006 Jun 21)

Add an option to specify the character to use when a double-width character is moved to the next line. Default '>', set to a space to blank it out. Check that char is single width when it's set (compare with 'listchars').

The generated vim.bat can avoid the loop for NT. (Carl Zmola, 2006 Sep 3)

When showing a diff between a non-existent file and an existing one, with the cursor in the empty buffer, the other buffer only shows the last line. Change the "insert" into a change from one line to many? (Yakov Lerner, 2008 May 27)

These two abbreviations don't give the same result:
```
	let asdfasdf = "xyz\<Left>"
	cabbr XXX <C-R>=asdfasdf<CR>
	cabbr YYY xyz<Left>
```

Michael Dietrich: maximized gvim sometimes displays output of external command partly. (2006 Dec 7)

In FileChangedShell command it's no longer allowed to switch to another buffer. But the changed buffer may differ from the current buffer, how to

reload it then?

For Aap: include a config.arg.example file with hints how to use config.arg.

Command line completion when 'cmdheight' is maximum and 'wildmenu' is set,
only one buffer line displayed, causes display errors.

Completing with 'wildmenu' and using <Up> and <Down> to move through directory
tree stops unexpectedly when using ":cd " and entering a directory that
doesn't contain other directories.

Default for 'background' is wrong when using xterm with 256 colors.
Table with estimates from Matteo Cavalleri, 2014 Jan 10.

Setting 'background' resets the Normal background color:
   highlight Normal ctermbg=DarkGray
   set background=dark
This is undesired, 'background' is supposed to tell Vim what the background
color is, not reset it.

Linux distributions:
- Suggest compiling xterm with --enable-tcap-query, so that nr of colors is
  known to Vim. 88 colors instead of 16 works better. See ":help
  xfree-xterm".
- Suggest including bare "vi" and "vim" with X11, syntax, etc.

Completion menu: For a wrapping line, completing a long file name, only the
start of the path is shown in the menu. Should move the menu to the right to
show more text of the completions. Shorten the items that don't fit in the
middle?

Accessing file#var in a function should not need the g: prepended.

When exiting detects a modified buffer, instead of opening the buffer in the
current tab, use an existing tab, if possible. Like finding a window where
the buffer is displayed. (Antonios Tsakiridis)

When ":cn" moves to an error in the same line the message isn't shortened.
Only skip shortening for ":cc"?

Write "making vim work better" for the docs (mostly pointers): *nice*
    - sourcing $VIMRUNTIME/vimrc_example.vim
    - setting 'mouse' to "a"
    - getting colors in xterm
    - compiling Vim with X11, GUI, etc.

Problem with ":call" and dictionary function. Hari Krishna Dara, Charles
Campbell 2006 Jul 06.

Syntax HL error caused by "containedin". (Peter Hodge, 2006 Oct 6)

A custom completion function in a ":command" cannot be a Funcref. (Andy
Wokula, 2007 Aug 25)

Problem with using :redir in user command completion function? (Hari Krishna
Dara, 2006 June 21)

Another resizing problem when setting 'columns' and 'lines' to a very large
number. (Tony Mechelynck, 2007 Feb 6)

After starting Vim, using '0 to jump somewhere in a file, ":sp" doesn't center
the cursor line. It works OK after some other commands.

Win32: Is it possible to have both postscript and Win32 printing?

Problem with 'cdpath' on MS-Windows when a directory is equal to $HOME. (2006 Jul 26, Gary Johnson)

Using UTF-8 character with ":command" does not work properly. (Matt Wozniski, 2008 Sep 29)

In the Netbeans interface add a "vimeval" function, so that the other side can check the result of has("patch13").

Cursor line at bottom of window instead of halfway after saving view and restoring. Only with 'nowrap'. (Robert Webb, 2008 Aug 25)

Netrw has trouble executing autocommands only for a directory. Add <isdir> and <notisdir> to autocommand patterns? Also <isfile>?

Add command modifier that skips wildcard expansion, so that you don't need to put backslashes before special chars, only for white space.

Syntax HL: open two windows on the same C code, delete a ")" in one window, resulting in highlighted "{" in that window, not in the other.

In mswin.vim: Instead of mapping <C-V> for Insert mode in a complicated way, can it be done like ":imap <C-V> <MiddleMouse>" without negative side effects?

GTK: when the Tab pages bar appears or disappears while the window is maximized the window is no longer maximized. Patch that has some idea but doesn't work from Geoffrey Antos, 2008 May 5.
Also: the window may no longer fit on the screen, thus the command line is not visible.

When right after "vim file", "M" then CTRL-W v the windows are scrolled differently and unexpectedly. Caused by patch 7.2.398?

The magic clipboard format "VimClipboard2" appears in several places. Should be only one.

Win32, NTFS: When editing a specific infostream directly and 'backupcopy' is "auto" should detect this situation and work like 'backupcopy' is "yes". File name is something like "c:\path\foo.txt:bar", includes a colon. (Alex Jakushev, 2008 Feb 1)

Small problem displaying diff filler line when opening windows with a script. (David Luyer, 2007 Mar 1 ~/Mail/oldmail/mool/in.15872)

Is it allowed that 'backupext' is empty? Problems when backup is in same dir as original file? If it's OK don't compare with 'patchmode'. (Thierry Closen)

Patch for supporting count before CR in quickfix window. (AOYAMA Shotaro, 2007 Jan 1)

Patch for adding ":lscscope". (Navdeep Parhar, 2007 Apr 26; update 2008 Apr 23)

":mkview" isn't called with the right buffer argument. Happens when using tabs and the autocommand "autocmd BufWinLeave * mkview". (James Vega, 2007 Jun 18)

When completing from another file that uses a different encoding completion text has the wrong encoding. E.g., when 'encoding' is utf-8 and file is

latin1. Example from Gombault Damien, 2007 Mar 24.

Syntax HL: When using "nextgroup" and the group has an empty match, there is no search at that position for another match. (Lukas Mai, 2008 April 11)

In gvim the backspace key produces a backspace character, but on Linux the VERASE key is Delete. Set VERASE to Backspace? (patch by Stephane Chazelas, 2007 Oct 16)

TermResponse autocommand isn't always triggered when using vimdiff. (Aron Griffis, 2007 Sep 19)

Create a gvimtutor.1 file and change Makefiles to install it.

When 'encoding' is utf-8 typing text at the end of the line causes previously typed characters to be redrawn. Caused by patch 7.1.329. (Tyler Spivey, 2008 Sep 3, 11)

When Vim in an xterm owns the selection and the user does ":shell" Vim doesn't respond to selection requests. Invoking XtDisownSelection() before executing the shell doesn't help. Would require forking and doing a message loop, like what happens for the GUI.

":vimgrep" does not recognize a recursive symlink. Is it possible to detect this, at least for Unix (using device/inode)?

When switching between windows the cursor is often put in the middle. Remember the relative position and restore that, just like lnum and col are restored. (Luc St-Louis)

Patch to support horizontal scroll wheel in GTK. Untested. (Bjorn Winckler, 2010 Jun 30)

Add an option for a minimal text length before inserting a line break for 'textwidth'. Avoids very short lines when a very long word follows. (Kartik Agaram)

Better plugin support (not plugin manager, see elsewhere for that):
- Avoid use of feedkeys, add eval functions where needed:
  - manipulating the Visual selection?
- Add createmark(): add a mark like mM, but return a unique ID. Need some way to clean them up again... Use a name + the script ID.
  Add createmark(, 'c') to track inserts/deletes before the column.
- Plugins need to make a lot of effort, lots of mappings, to know what happened before pressing the key that triggers a plugin action. How about keeping the last N pressed keys, so that they do not need to be mapped?
- equivalent of netbeans_beval_cb(). With an autocommand?
- Add something to enable debugging when a remote message is received.

More patches:
-   Another patch for Javascript indenting. (Hari Kumar, 2010 Jul 11)
    Needs a few tests.
-   Add 'cscopeignorecase' option. (Liang Wenzhi, 2006 Sept 3)
-   Load intl.dll too, not only libintl.dll. (Mike Williams, 2006 May 9, docs patch May 10)
-   Extra argument to strtrans() to translate special keys to their name (Eric Arnold, 2006 May 22)
-   'threglookexp' option: only match with first word in thesaurus file. (Jakson A. Aquino, 2006 Jun 14)
-   Mac: indicate whether a buffer was modified. (Nicolas Weber, 2006 Jun 30)

```
-   Allow negative 'nrwidth' for left aligning. (Nathan Laredo, 2006 Aug 16)
-   ml_append_string(): efficiently append to an existing line. (Brad
    Beveridge, 2006 Aug 26)  Use in some situations, e.g., when pasting a
    character at a time?
-   recognize hex numbers better. (Mark Manning, 2006 Sep 13)
-   Add <AbbrExpand> key, to expand an abbreviation in a mapping. (Kana
    Natsuno, 2008 Jul 17)
-   Add 'wspara' option, also accept blank lines like empty lines for "{" and
    "}". (Mark Lundquist, 2008 Jul 18)
-   Patch to add CTRL-T to delete part of a path on cmdline. (Adek, 2008 Jul
    21)
-   Instead of creating a copy of the tutor in all the shell scripts, do it in
    vimtutor.vim. (Jan Minar, 2008 Jul 20)
-   When fsync() fails there is no hint about what went wrong.  Patch by Ben
    Schmidt, 2008 Jul 22.
-   testdir/Make_dos_sh.mak for running tests with MingW. (Bill Mccarthy, 2008
    Sep 13)
-   Replace ccomplete.vim by cppcomplete.vim from www.vim.org?  script 1520 by
    Vissale Neang.  (Martin Stubenschrott) Asked Vissale to make the scripts
    more friendly for the Vim distribution.
    New version received 2008 Jan 6.
    No maintenance in two years...
-   Patch to open dropped files in new tabs. (Michael Trim, 2010 Aug 3)

Awaiting updated patches:
9   Mac unicode patch (Da Woon Jung, Eckehard Berns):
    8   Add patch from Muraoka Taro (Mar 16) to support input method on Mac?
        New patch 2004 Jun 16
    - selecting proportional font breaks display
    - UTF-8 text causes display problems.  Font replacement causes this.
    - Command-key mappings do not work. (Alan Schmitt)
    - With 'nopaste' pasting is wrong, with 'paste' Command-V doesn't work.
      (Alan Schmitt)
    - remove 'macatsui' option when this has been fixed.
    - when 'macatsui' is off should we always convert to "macroman" and ignore
      'termencoding'?
9   HTML indenting can be slow.  Caused by using searchpair().  Can search()
    be used instead?  A.Politz is looking into a solution.
8   Win32: Add minidump generation. (George Reilly, 2006 Apr 24)
7   Completion of network shares, patch by Yasuhiro Matsumoto.
    Update 2004 Sep 6.
    How does this work?  Missing comments.
8   Add a few more command names to the menus.  Patch from Jiri Brezina
    (28 feb 2002).  Will mess the translations...
7   ATTENTION dialog choices are more logical when "Delete it" appears
    before "Quit".  Patch by Robert Webb, 2004 May 3.
-   Include flipcase patch: ~/vim/patches/wall.flipcase2 ?  Make it work
    for multi-byte characters.
-   Win32: add options to print dialog.  Patch from Vipin Aravind.
-   Patch to add highlighting for whitespace. (Tom Schumm, 2003 Jul 5)
    use the patch that keeps using HLF_8 if HLF_WS has not
    been given values.
    Add section in help files for these highlight groups?
8   "fg" and "bg" don't work in an xterm.  Get default colors from xterm
    with an ESC sequence.
    xterm can send colors for many things.  E.g. for the cursor:
        <Esc>]12;?<Bel>
    Can use this to get the background color and restore the colors on exit.
7   Add "DefaultFG" and "DefaultBG" for the colors of the menu. (Marcin
    Dalecki has a patch for Motif and Carbon)
-   Add possibility to highlight specific columns (for Fortran).  Or put a
    line in between columns (e.g., for 'textwidth').
```

```
    Patch to add 'hlcolumn' from Vit Stradal, 2004 May 20.
8   Add functions:
    gettext()           Translate a message.  (Patch from Yasuhiro Matsumoto)
                        Update 2004 Sep 10
                        Another patch from Edward L. Fox (2005 Nov 24)
                        Search in 'runtimepath'?
                        More docs needed about how to use this.
                        How to get the messages into the .po files?
    confirm()           add "flags" argument, with 'v' for vertical
                            layout and 'c' for console dialog. (Haegg)
                            Flemming Madsen has a patch for the 'c' flag
                            (2003 May 13)
    raisewin()          raise gvim window (see HierAssist patch for
                            Tcl implementation ~/vim/HierAssist/ )
    taglist()           add argument to specify maximum number of matches.
                        useful for interactive things or completion.
    col('^')            column of first non-white character.
                        Can use "len(substitute(getline('.'), '\S.*', '', ''))
                        + 1", but that's ugly.
7   Add patch from Benoit Cerrina to integrate Vim and Perl functions
    better.  Now also works for Ruby (2001 Nov 10)
-   Patch from Herculano de Lima Einloft Neto for better formatting of the
    quickfix window (2004 dec 2)
7   When 'rightleft' is set, the search pattern should be displayed right
    to left as well?  See patch of Dec 26. (Nadim Shaikli)
8   Option to lock all used memory so that it doesn't get swapped to disk
    (uncrypted).  Patch by Jason Holt, 2003 May 23.  Uses mlock.
7   Add ! register, for shell commands. (patch from Grenie)
8   In the gzip plugin, also recognize *.gz.orig, *.gz.bak, etc.  Like it's
    done for filetype detection.  Patch from Walter Briscoe, 2003 Jul 1.
7   Add a "-@ filelist" argument: read file names from a file. (David
    Kotchan has a patch for it)
7   Add ":justify" command.  Patch from Vit Stradal 2002 Nov 25.
-   findmatch() should be adjusted for Lisp.  See remark at
    get_lisp_indent().  Esp. \( and \) should be skipped. (Dorai Sitaram,
    incomplete patch Mar 18)
-   For GUI Find/Replace dialog support using a regexp.  Patch for Motif
    and GTK by degreneir (nov 10 and nov 18).
-   Patch for "paranoid mode" by Kevin Collins, March 7.  Needs much more work.


Vi incompatibility:
-   Try new POSIX tests, made after my comments. (Geoff Clare, 2005 April 7)
    Version 1.5 is in ~/src/posix/1.5. (Lynne Canal)
8   With undo/redo only marks in the changed lines should be changed.  Other
    marks should be kept.  Vi keeps each mark at the same text, even when it
    is deleted or restored. (Webb)
    Also: A mark is lost after: make change, undo, redo and undo.
    Example: "{d'" then "u" then "d'": deletes an extra line, because the ''
    position is one line down. (Veselinovic)
8   When stdin is not a tty, and Vim reads commands from it, an error should
    make Vim exit.
7   Unix Vim (not gvim): Typing CTRL-C in Ex mode should finish the line
    (currently you can continue typing, but it's truncated later anyway).
    Requires a way to make CTRL-C interrupt select() when in cooked input.
8   When loading a file in the .exrc, Vi loads the argument anyway.  Vim skips
    loading the argument if there is a file already.  When no file argument
    given, Vi starts with an empty buffer, Vim keeps the loaded file. (Bearded)
6   In Insert mode, when using <BS> or <Del>, don't wipe out the text, but
    only move back the cursor.  Behaves like '$' in 'cpoptions'.  Use a flag
    in 'cpoptions' to switch this on/off.
8   When editing a file which is a symbolic link, and then opening another
```

```
    symbolic link on the same file, Vim uses the name of the first one.
    Adjust the file name in the buffer to the last one used?  Use several file
    names in one buffer???
    Also: When first editing file "test", which is symlink to "test2", and
    then editing "test2", you end up editing buffer "test" again.  It's not
    logical that the name that was first used sticks with the buffer.
7   The ":undo" command works differently in Ex mode.  Edit a file, make some
    changes, "Q", "undo" and _all_ changes are undone, like the ":visual"
    command was one command.
    On the other hand, an ":undo" command in an Ex script only undoes the last
    change (e.g., use two :append commands, then :undo).
7   The ":map" command output overwrites the command.  Perhaps it should keep
    the ":map" when it's used without arguments?
7   CTRL-L is not the end of a section?  It is for Posix!  Make it an option.
7   Implement 'prompt' option.  Init to off when stdin is not a tty.
7   Add a way to send an email for a crashed edit session.  Create a file when
    making changes (containing name of the swap file), delete it when writing
    the file.  Supply a program that can check for crashed sessions (either
    all, for a system startup, or for one user, for in a .login file).
7   Vi doesn't do autoindenting when input is not from a tty (in Ex mode).
7   "z3<CR>" should still use the whole window, but only redisplay 3 lines.
7   ":tag xx" should move the cursor to the first non-blank.  Or should it go
    to the match with the tag?  Option?
7   Implement 'autoprint'/'ap' option.
7   Add flag in 'cpoptions' that makes <BS> after a count work like <Del>
    (Sayre).
7   Add flag in 'cpoptions' that makes operator (yank, filter) not move the
    cursor, at least when cancelled. (default Vi compatible).
7   This Vi-trick doesn't work: "Q" to go to Ex mode, then "g/pattern/visual".
    In Vi you can edit in visual mode, and when doing "Q" you jump to the next
    match.  Nvi can do it too.
7   Support '\' for line continuation in Ex mode for these commands: (Luebking)
        g/./a\              g/pattern1/ s/pattern2/rep1\\
        line 1\             line 2\\
        line 2\             line 3\\
        .                   line4/
6   ":e /tmp/$tty" doesn't work.  ":e $uid" does.  Is $tty not set because of
    the way the shell is started?
6   Vi compatibility (optional): make "ia<CR><ESC>10." do the same strange
    thing.  (only repeat insert for the first line).


GTK+ GUI known bugs:
9   Crash with X command server over ssh. (Ciaran McCreesh, 2006 Feb 6)
8   GTK 2: Combining UTF-8 characters not displayed properly in menus (Mikolaj
    Machowski)  They are displayed as separate characters.  Problem in
    creating a label?
8   GTK 2: Combining UTF-8 characters are sometimes not drawn properly.
    Depends on the font size, "monospace 13" has the problem.  Vim seems to do
    everything right, must be a GTK bug.  Is there a way to work around it?
9   Can't paste a Visual selection from GTK-gvim to vim in xterm or Motif gvim
    when it is longer than 4000 characters.  Works OK from gvim to gvim and
    vim to vim.  Pasting through xterm (using the shift key) also works.
    It starts working after GTK gvim loses the selection and gains it again.
-   Gnome2: When moving the toolbar out of the dock, so that it becomes
    floating, it can no longer be moved.  Therefore making it float has been
    blocked for now.


Win32 GUI known bugs:
-   Win32: tearoff menu window should have a scrollbar when it's taller than
    the screen.
```

8 The -P argument doesn't work very well with many MDI applications.
The last argument of CreateWindowEx() should be used, see MSDN docs.
Tutorial: http://win32assembly.online.fr/tut32.html
8 In eval.c, io.h is included when MSWIN32 is defined. Shouldn't this be
WIN32? Or can including io.h be moved to vim.h? (Dan Sharp)
6 Win32 GUI: With "-u NONE -U NONE" and doing "CTRL-W v" "CTRL-W o", the ":"
of ":only" is highlighted like the cursor. (Lipelis)
8 When 'encoding' is "utf-8", should use 'guifont' for both normal and wide
characters to make Asian languages work. Win32 fonts contain both
type of characters.
7 When font smoothing is enabled, redrawing can become very slow. The reason
appears to be drawing with a transparent background. Would it be possible
to use an opaque background in most places?
7 The cursor color indicating IME mode doesn't work properly. (Shizhu Pan,
2004 May 9)
8 Win32: When clicking on the gvim title bar, which gives it focus, produces
a file-changed dialog, after clicking on a button in that dialog the gvim
window follows the mouse. The button-up event is lost. Only with
MS-Windows 98?
Try this: ":set sw ts", get enter-prompt, then change the file in a
console, go back to Vim and click "reload" in the dialog for the changed
file: Window moves with the cursor!
Put focus event in input buffer and let generic Vim code handle it?
8 Win32 GUI: With maximized window, ":set go-=r" doesn't use the space that
comes available. (Poucet) It works OK on Win 98 but doesn't work on Win
NT 4.0. Leaves a grey area where the scrollbar was. ":set go+=r" also
doesn't work properly.
8 When Vim is minimized and when maximizing it a file-changed dialog pops
up, Vim isn't maximized. It should be done before the dialog, so that it
appears in the right position. (Webb)
9 When selecting at the more-prompt or hit-enter-prompt, the right mouse
button doesn't give popup menu.
At the hit-enter prompt CTRL-Y doesn't work to copy the modeless
selection.
On the command line, don't get a popup menu for the right mouse button.
Let the middle button paste selected text (not the clipboard but the
non-Visual selection)? Otherwise CTRL-Y has to be used to copy the text.
8 When 'grepprg' doesn't execute, the error only flashes by, the
user can hardly see what is wrong. (Moore)
Could use vimrun with an "-nowait" argument to only wait when an error
occurs, but "command.com" doesn't return an error code.
8 When the 'shell' cannot be executed, should give an appropriate error msg.
Esp. for a filter command, currently it only complains the file could not
be read.
7 Add an option to add one pixel column to the character width? Lucida
Console italic is wider than the normal font ("d" overlaps with next char).
Opposite of 'linespace': 'columnspace'.
7 At the hit-enter prompt scrolling now no longer works. Need to use the
keyboard to get around this. Pretend <CR> was hit when the user tries to
scroll?
7 Scrollbar width doesn't change when selecting other windows appearance.
Also background color of Toolbar and rectangle below vert. scrollbar.
6 Drawing text transparently doesn't seem to work (when drawing part cursor).
8 CTRL key doesn't always work in combination with ALT key. It does work
for function keys, not for alphabetic characters. Perhaps this is because
CTRL-ALT is used by Windows as AltGr?
8 CTRL-- doesn't work for AZERTY, because it's CTRL-[for QWERTY. How do we
know which keyboard is being used?
7 When scrolling, and a background color is dithered, the dither pattern
doesn't always join correctly between the scrolled area and the new drawn
area (Koloseike).
8 When gui_init_font() is called with "*", p_guifont is freed while it might

```
    still be used somewhere.  This is too tricky, do the font selection first,
    then set the new font by name (requires putting all logfont parameters in
    the font name).


Athena and Motif:
6   New Motif toolbar button from Marcin Dalecki:
    - When the mouse pointer is over an Agide button the red becomes black.
      Something with the way colors are specified in the .xpm file.
    - The pixmap is two pixels smaller than it should be.  The gap is filled
      with grey instead of the current toolbar background color.
9   Can configure be changed to disable netbeans if the Xpm library is
    required and it's missing?
8   When using the resource "Vim*borderwidth 2" the widgets are positioned
    wrong.
9   XIM is disabled by default for SGI/IRIX.  Fix XIM so that 'imdisable' can
    be off by default.
9   XIM doesn't work properly for Athena/Motif. (Yasuhiro Matsumoto)  For now,
    keep XIM active at all times when the input method has the preediting
    flag.
8   X11: A menu that contains an umlaut is truncated at that character.
    Happens when the locale is "C", which uses ASCII instead of IS0-8859-1.
    Is there a way to use latin1 by default?  Gnome_init() seems to do this.
8   Perhaps use fontsets for everything?
6   When starting in English and switching the language to Japanese, setting
    the locale with ":lang", 'guifontset' and "hi menu font=", deleting all
    menus and setting them again, the menus don't use the new font.  Most of
    the tooltips work though...
7   Motif: when using a file selection dialog, the specified file name is not
    always used (when specifying a filter or another directory).
8   When 'encoding' is different from the current locale (e.g., utf-8) the
    menu strings don't work.  Requires conversion from 'encoding' to the
    current locale.  Workaround: set 'langmenu'.


Athena GUI:
9   The first event for any button in the menu or toolbar appears to get lost.
    The second click on a menu does work.
9   When dragging the scrollbar thumb very fast, focus is only obtained in
    the scrollbar itself.  And the thumb is no longer updated when moving
    through files.
7   The file selector is not resizable.  With a big font it is difficult to
    read long file names. (Schroeder)
4   Re-write the widget attachments and code so that we will not have to go
    through and calculate the absolute position of every widget every time the
    window is refreshed/changes size.  This will help the "flashing-widgets"
    problem during a refresh.
5   When starting gvim with all the default colors and then typing
    ":hi Menu guibg=cyan", the menus change color but the background of the
    pullright pixmap doesn't change colors.
    If you type ":hi Menu guibg=cyan font=anyfont", then the pixmap changes
    colors as it should.
    Allocating a new pixmap and setting the resource doesn't change the
    pullright pixmap's colors.  Why?  Possible Athena bug?


Motif GUI:
-   gui_mch_browsedir() is missing, browsedir() doesn't work nicely.
7   Use XmStringCreateLocalized() instead of XmStringCreateSimple()?
    David Harrison says it's OK (it exists in Motif 1.2).
8   Lesstif: When deleting a menu that's torn off, the torn off menu becomes
    very small instead of disappearing.  When closing it, Vim crashes.
```

(Phillipps)

GUI:
9 On Solaris, creating the popup menu causes the right mouse button no longer to work for extending the selection. (Halevy)
9 When running an external program, it can't always be killed with CTRL-C. e.g., on Solaris 5.5, when using "K" (Keech). Other 'guipty' problems on Solaris 2.6. (Marley)
9 On Solaris: Using a "-geometry" argument, bigger than the window where Vim is started from, causes empty lines below the cmdline. (raf)
8 X11 GUI: When menu is disabled by excluding 'm' from 'guioptions', ALT key should not be used to trigger a menu (like the Win32 version).
8 When setting 'langmenu', it should be effective immediately. Store both the English and the translated text in the menu structure. Re-generate the translation when 'langmenu' has changed.
8 Basic flaw in the GUI code: NextScreen is updated before calling gui_write(), but the GUI code relies on NextScreen to represent the state of where it is processing the output.
Need better separation of Vim core and GUI code.
8 When fontset support is enabled, setting 'guifont' to a single font doesn't work.
8 Menu priority for sub-menus for: Amiga.
8 When translating menus ignore the part after the Tab, the shortcut. So that the same menu item with a different shortcut (e.g., for the Mac) are still translated.
8 Add menu separators for Amiga.
8 Add way to specify the file filter for the browse dialog. At least for browse().
8 Add dialog for search/replace to other GUIs? Tk has something for this, use that code? Or use console dialog.
8 When selecting a font with the font dialog and the font is invalid, the error message disappears too quick.
7 More features in the find/replace dialog:
- regexp on/off
- search in selection/buffer/all buffers/directory
when all buffers/directory is used:
- filter for file name
when directory is used:
- subdirectory on/off
- top directory browser
8 gui_check_colors() is not called at the right moment. Do it much later, to avoid problems.
8 gui_update_cursor() is called for a cursor shape change, even when there are mappings to be processed. Only do something when going to wait for input. Or maybe every 100 ms?
8 X11: When the window size is reduced to fit on screen, there are blank lines below the text and bottom scrollbar. "gvim -geometry 80x78+0+0". When the "+0+0" is omitted it works.
8 When starting an external command, and 'guipty' set, BS and DEL are mixed up. Set erase character somehow?
8 A dead circumflex followed by a space should give the '^' character (Rommel). Look how xterm does this.
Also: Bednar has some code for dead key handling.
Also: Nedit 5.0.2 with USE_XMIM does it right. (Gaya)
8 The compose key doesn't work properly (Cepas). Both for Win32 and X11.
7 The cursor in an inactive window should be hollow. Currently it's not visible.
7 GUI on Solaris 2.5.1, using /usr/dt/..: When gvim starts, cursor is hollow, after window lowered/raised it's OK. (Godfrey)
7 When starting GUI with ":gui", and window is made smaller because it doesn't fit on the screen, there is an extra redraw.

```
8   When setting font with .Xdefaults, there is an extra empty line at the
    bottom, which disappears when using ":set guifont=<Tab>". (Chadzelek)
8   When font shape changes, but not the size, doing ":set font=" does not
    redraw the screen with the new font.  Also for Win32.
    When the size changes, on Solaris 2.5 there isn't a redraw for the
    remaining part of the window (Phillipps).
-   Flashes really badly in certain cases when running remotely from a Sun.
4   Re-write the code so that the highlighting isn't changed multiple times
    when doing a ":hi clear".  The color changes happen three or more times
    currently.  This is very obvious on a 66Mhz 486.


Win32 console:
8   Should $USERPROFILE be preferred above $HOMEDRIVE/$HOMEPATH?  No, but it's
    a good fallback, thus use:
            $HOME
            $HOMEDRIVE$HOMEPATH
            SHGetSpecialFolderPath(NULL, lpzsPath, CSIDL_APPDATA, FALSE);
            $USERPROFILE
            SHGetSpecialFolderPath(NULL, lpzsPath, CSIDL_COMMON_APPDATA, FALSE);
            $ALLUSERSPROFILE
            $SYSTEMDRIVE\
            C:\
8   Win32 console: <M-Up> and <M-Down> don't work. (Geddes)  We don't have
    special keys for these.  Should use modifier + key.
8   Win32 console: caps-lock makes non-alpha keys work like with shift.
    Should work like in the GUI version.
8   Environment variables in DOS are not case sensitive.  Make a define for
    STRCMP_ENV(), and use it when comparing environment var names.
8   Setting 'shellslash' has no immediate effect.  Change all file names when
    it is set/reset?  Or only use it when actually executing a shell command?
8   When editing a file on a Samba server, case might matter.  ":e file"
    followed by ":e FILE" will edit "file" again, even though "FILE" might be
    another one.  Set last used name in buflist_new()?  Fix do_ecmd(), etc.
8   When a buffer is editing a file like "ftp://mach/file", which is not going
    to be used like a normal file name, don't change the slashes to
    backslashes. (Ronald Hoellwarth)


Win32 console:
9   When editing a file by its short file name, it should be expanded into its
    long file name, to avoid problems like these: (Mccollister)
     1) Create a file called ".bashrc" using some other editor.
     2) Drag that file onto a shortcut or the actual executable.
     3) Note that the file name is something like BASHRC~1
     4) Go to File->Save As menu item and type ".bashrc" as the file name.
     5) Press "Yes" to indicate that I want to overwrite the file.
     6) Note that the message "File exists (add ! to override)" is displayed
        and the file is not saved.
    Use FindFirstFile() to expand a file name and directory in the path to its
    long name.
8   Also implement 'conskey' option for the Win32 console version?  Look at
    how Xvi does console I/O under Windows NT.
7   Re-install the use of $TERM and support the use of different terminals,
    besides the console.
8   Use of <altgr> modifier doesn't work?  5.3 was OK. (Garcia-Suarez/Guckes)
9   Mapping <C-S-Tab> doesn't work correctly.  How to see the difference with
    <C-S-i>?
9   tmpnam() uses file in root of file system: "\asdf".  That doesn't work on
    a Netware network drive.  Use same function as for Win32 GUI?
8   In os_win32.h, HAVE_STRICMP and HAVE_STRNICMP are defined only if __GNUC__
    is not defined.  Shouldn't that be the other way around?
```

```
Amiga:
8   In mch_inchar() should use convert_input_safe() to handle incomplete byte
    sequences.
9   In mch_expandpath() a "*" is to be expanded, but "\*" isn't.  Remove
    backslashes in result.
8   Executing a shell, only one option for 'shell' is separated.  Should do
    all options, using white space separation.


Macintosh:
-   GUI: gui_mch_browsedir() is missing.
7   Loading the Perl library only works on OS/X 10.2 or 10.3, never on both.
    Load the Perl library dynamically see Python sources file dynload_mac
    (Jack)
    dynamic linking: http://developer.apple.com/technotes/tn2002/tn2064.html
8   inputdialog() doesn't resize when giving more text lines. (David Fishburn,
    2006 Sept 28)
8   Define vim_mkdir() for Macintosh.
8   Define mch_writable() for Macintosh.
9   When DiskLock is running, using a swap file causes a crash.  Appears to be
    a problem with writing a file that starts with a dot. (Giacalone)
9   In mac_expandpath() check that handling of backslashes is done properly.


"Small" problems:
-   Can't disable terminal flow control, to enable the use of CTRL-S and
    CTRL-Q.  Add an option for it?
-   When using e_secure in do_one_cmd() mention the command being executed,
    otherwise it's not clear where it comes from.
-   When the quickfix window is open and executing ":echo 'hello'" using the
    Command-line window, the text is immediately removed by the redrawing.
    (Michael Henry, 2008 Nov 1)
    Generic solution: When redrawing while there is a message on the
    cmdline, don't erase the display but draw over the existing text.
    Other solution, redraw after closing the cmdline window, before executing
    the command.
9   For Turkish vim_tolower() and vim_toupper() also need to use utf_
    functions for characters below 0x80. (Sertacyildiz)
9   When the last edited file is a help file, using '0 in a new Vim doesn't
    edit the file as a help file.  'filetype' is OK, but 'iskeyword' isn't,
    file isn't readonly, etc.
8   When an ":edit" is inside a try command and the ATTENTION prompt is used,
    the :catch commands are always executed, also when the file is edited
    normally.  Should reset did_emsg and undo side effects.  Also make sure
    the ATTENTION message shows up.  Servatius Brandt works on this.
7   Vimtutor leaves escape sequence in terminal. This is the xterm response to
    requesting the version number.  (Yasuhiro Matsumoto)
8   When redirecting and using ":silent" the current column for displaying and
    redirection can be different.  Use a separate variable to hold the column
    for redirection.
7   The messages for "vim --help" and "vim --version" don't use
    'termencoding'.
-   Could the hit-enter prompt be avoided when a message only overlaps the
    'showcmd' area?  Clear that area when the next cmd is typed.
8   When 'scrollbind' is set, a window won't scroll horizontally if the cursor
    line is too short.  Add a word in 'scrollopt' to allow moving the cursor
    to longer line that is visible.  A similar thing is done for the GUI when
    using the horizontal scrollbar.
7   VisVim can only open one file.  Hard to solve: each opened file is passed
    with a separate invocation, would need to use timestamps to know the
```

```
    invocations belong together.
8   When giving a ":bwipeout" command a file-changed dialog may popup for this
    buffer, which is pointless.  (Mike Williams)
8   On MS-Windows ":make" doesn't show output while it is working.  Use the
    tee.exe from  http://unxutils.sourceforge.net/ ?  About 16 Kbyte in the
    UnxUtils.zip archive.
    Is it better than what we have in src/tee?
8   When doing Insert mode completion a mapping cannot recursively call
    edit(), because the completion information is global.  Put everything in
    an allocated structure?
8   Command line completion: buffers "foo.txt" and "../b/foo.txt", completing
    ":buf foo<Tab>" doesn't find the second one. (George V. Reilly)
7   mb_off2cells() doesn't work correctly on the tail byte of a double-byte
    character. (Yasuhiro Matsumoto)  It should return 1 when used on a tail
    byte, like for utf-8.  Store second byte of double-byte in ScreenLines2[]
    (like for DBCS_JPNU) and put a zero in the second byte (like for UTF-8).
7   Inside a function with "perl <<EOF" a line with "$i++" is recognized as an
    ":insert" command, causing the following "endfunction" not to be found.
    Add skipping this perl construction inside function definitions.
7   When 'ttimeoutlen' is 10 and 'timeoutlen' is 1000, there is a keycode
    "<Esc>a" and a mapping <Esc>x", when typing "<Esc>a" with half a second
    delay should not be interpreted as a keycode. (Hans Ginzel)
7   ":botright 1 new" twice causes all window heights to be changed.  Make the
    bottom window only bigger as much as needed.
7   The Cygwin and MingW makefiles define "PC", but it's not used anywhere.
    Remove? (Dan Sharp)
9   User commands use the context of the script they were defined in.  This
    causes a "s:var" argument to unexpectedly use a variable in the defining
    script, not the calling script.  Add an argument to ":command":
    "-keepcontext".  Do replace <SID>, so that a function in the defining
    script can be called.
8   The Japanese message translations for MS-Windows are called ja.sjis.po,
    but they use encoding cp932.  Rename the file and check that it still
    works.
8   A very long message in confirm() can't be quit.  Make this possible with
    CTRL-C.
8   "gf" always excludes trailing punctuation characters.  file_name_in_line()
    is currently fixed to use ".,:;!".  Add an option to make this
    configurable?
8   'hkmap' should probably be global-local.
8   Using ":s" in a function changes the previous replacement string.  Save
    "old_sub" in save_search_patterns()?
8   Should allow multi-byte characters for the delimiter: ":s+a+b+" where "+"
    is a multi-byte character.
8   When appending to a file and 'patchmode' isn't empty, a backup file is
    always written, even when the original file already exists.
9   When getting focus while writing a large file, could warn for this file
    being changed outside of Vim.  Avoid checking this while the file is being
    written.
7   The message in bt_dontwrite_msg() could be clearer.
8   The script ID that is stored with an option and displayed with ":verbose
    set" isn't reset when the option is set internally.  For example when
    'foldlevel' is set from 'foldlevelstart'.
8   Also store the line number with the script ID and use it for ":verbose",
    so that "set nocompatible" is found when it changes other option values.
    When an option is set indirectly mention the command?  E.g. when
    ":diffsplit" sets 'foldmethod'.
8   In the fileformat dialog, "Cancel" isn't translated.  Add a global
    variable for this. (Eduardo Fernandez)
9   When editing a file with 'readonly' set, there is no check for an existing
    swap file.  Then using ":write" (without making any changes) doesn't give
    a warning either.  Should check for an existing swap file without creating
```

```
    one. Unfinished patch by Ian Kelling, 2008 July 14.
7   When 'showbreak' is set, the amount of space a Tab occupies changes.
    Should work like 'showbreak' is inserted without changing the Tabs.
7   When 'mousefocus' is set and switching to another window with a typed
    command, the mouse pointer may be moved to a part of the window that's
    covered by another window and we lose focus.  Only move in the y
    direction, not horizontally?
8   ":hardcopy":
    - Using the cterm_color[] table is wrong when t_colors is > 16.
    - Need to handle unprintable characters.
    - Win32: On a B&W printer syntax highlighting isn't visible.  Perform
      dithering to make grey text?
    - Add a flag in 'printoptions' to add an empty page to make the total
      number even.  "addempty"? (Mike Williams)
    - Respect 'linebreak'.  Perhaps also 'showbreak'?
    - Should interpret CTRL-L as a page break.
    - Grey line numbers are not always readable.  Add field in 'printoptions'.
      Default to black when no syntax highlighting.
    - Be able to print a window in diff mode.
    - Be able to specify a colorscheme to use for printing.  And a separate
      one for B&W printing (if that can be detected).
8   When 'virtualedit' is "block,insert" and encoding is "utf-8", selecting a
    block of one double-wide character, then "d" deletes only half of it.
8   When 'virtualedit' is set, should "I" in blockwise visual mode also insert
    in lines that don't extend into the block?
8   With 'virtualedit' set, in Insert mode just after the end of line, CTRL-O
    yh does not yank the last character of the line. (Pavel Papushev)
    Doing "hl" first appears to make it work.
8   With 'virtualedit' set it's possible to move into the blank area from
    'linebreak'.
8   With 'virtualedit' set and 'selection' "exclusive", a Visual selection
    that ends in or after a tab, "d" doesn't delete (part of) the tab.
    (Helmut Stiegler)
9   When jumping to a tag, the search pattern is put in the history.  When
    'magic' is on, the pattern may not work.  Translate the pattern depending
    on p_magic when putting it in the history?  Alternative: Store value of
    'magic' in history.  (Margo)
9   optwin.vim: Restoring a mapping for <Space> or <CR> is not correct for
    ":noremap".  Add "mapcmd({string}, {mode})?  Use code from ":mkexrc".
9   incsearch is incorrect for "/that/<Return>/this/;//" (last search pattern
    isn't updated).
9   term_console is used before it is set (msdos, Amiga).
9   Get out-of-memory for ":g/^/,$s//@/" on 1000 lines, this is not handled
    correctly.  Get many error messages while redrawing the screen, which
    cause another redraw, etc.
8   [<C-I> doesn't work when '*' is in 'iskeyword'.  find_pattern_in_path()
    must escape special characters in the pattern.
8   Vim can overwrite a read-only file with ":w!".  ":w" can't overwrite an
    existing file, "w!" can, but perhaps not a read-only file?  Then use
    ":w!!" for that.
    Or ask for permission to overwrite it (if file can be made writable) and
    restore file to readonly afterwards.
    Overwriting a file for which a swap file exists is similar issue.
7   When compiled with "xterm_clipboard", startup can be slower and might get
    error message for invalid $DISPLAY.  Try connecting to the X server in the
    background (forked), so that Vim starts up quicker?  Connect as soon as
    the clipboard is to be used (Visual select mode starts, paste from
    clipboard)
7   X11: Some people prefer to use CLIPBOARD instead of PRIMARY for the normal
    selection.  Add an "xclipboard" argument to the 'clipboard' option? (Mark
    Waggoner)
8   For xterm need to open a connection to the X server to get the window
```

```
    title, which can be slow.  Can also get the title with "<Esc>[21t", no
    need to use X11 calls.  This returns "<Esc>]l{title}<Esc>\".
6   When the xterm reports the number of colors, a redraw occurs.  This is
    annoying on a slow connection.  Wait for the xterm to report the number of
    colors before drawing the screen.  With a timeout.
8   When the builtin xterm termcap contains codes that are not wanted, need a
    way to avoid using the builtin termcap.
8   Xterm sends ^[[H for <Home> and ^[[F for <End> in some mode.  Also
    recognize these keys?  Mostly useful for xterm simulators, like gnometerm.
    See http://dickey.his.com/xterm/xterm.faq.html#xterm_pc_style.
8   For xterm also recognize keypad up/down/left/right and insert.
8   '[ and '] should be set to start/end of line when using a linewise operator
    (e.g., ":w").
8   CTRL-A can't handle big "long" numbers, they become negative.  Check for
    "-" character, if not present, use unsigned long.
8   Add suspending with CTRL-Z at the "more" prompt, and when executing a long
    script in do_cmdline().
8   When using 'hidden', many swap files will be open.  When Vim runs into the
    maximum number of open files, error messages will appear.  Detect that
    this problem is present, and close any hidden files that don't have
    changes.
8   With 'viminfo' set such that the ".viminfo" file is written on a FAT
    filesystem, an illegal file name may be created: ".vim".
8   For each buffer that is opened, the viminfo file is opened and read to
    check for file marks.  This can be slow.
7   In xterm, recognize both vt100 and vt220 cursor keys.  Change
    add_termcode() to not remove an existing entry for a name, when it's
    needed.
    Need a generic solution to recognize different codes for the same key.
8   Core dump within signal function: gdb doesn't show stack backtrace!  Option
    to skip catch_signals()?
9   Repeating a "cw" with "." doesn't work if the text was pasted from the
    clipboard. (Thomas Jones)  It's because the menu/toolbar item exits Insert
    mode and uses "gP".  How to fix this without breaking inserting a block of
    text?
8   In Replace mode pasting from the clipboard (using menu or toolbar) inserts
    all the text.  Add ":rmenu"?
8   Pasting with the mouse in Replace mode inserts the text, instead of
    overwriting, when it is more than one line.  Same for using <C-R>.
9   CTRL-E and CTRL-Y don't work in small window when 'so' is 4 and lines are
    wrapping (Acevedo/in.226).  E.g., when using CTRL-E, window height 7,
    window might actually scroll down when last line of buffer is displayed.
    --> Remember if the previous command was "cursor follows screen" or
    "screen follow cursor" and use this in cursupdate().
7   tilde_replace() can only handle "~/", should also do "~user/".
    Get the list of home directories (from /etc/passwd?  Use getpwent()) and
    use some clever algorithm to match a path with that.  Find common strings
    in the list?
8   When dragging status line with mouse, sometimes a jump when first clicking
    on the status line (caused by 'winheight').  Select window on button up,
    instead of on button down.
8   Dragging the status line doesn't scroll but redraw.
9   Evaluating 'statusline' in build_stl_str_hl() does not properly check for
    reaching the end of the available buffer.
    Patch to dynamically allocate the buffer for % items. (Eric Arnold, 2006
    May 14)
8   When performing incremental search, should abort searching as soon as a
    character is typed.
8   When the value of $MAKE contains a path, configure can't handle this.
    It's an autoconf bug.  Remove the path from $MAKE to work around it.
8   How to set VIMRC_FILE to \"something\" for configure?  Why does this not
    work: CFLAGS='-DVIMRC_FILE=\"/mydir/myfile\"' ./configure
```

8 The temporary file is sometimes not writable. Check for this, and use an alternate name when it isn't. Or add the 'temptemplate' option: template for the temp file name ":set temptemplate=/usr/tmp/?????.tmp". Also: Win32 version uses Windows temp directory, which might not work for cygwin bash.
7 Get error "*, \+ or \(operand could be empty" for pattern "\(.\)\1\{3}". Remember flags for backreferences.
7 When switching to Daylight Saving Time, Vim complains that a file has been changed since last read. Can we use a function that uses GMT?
7 When completing an environment variable after a '$', check for file names that contain a '$' after all have been found.
8 When "cm" termcap entry is missing, starting gvim shouldn't complain about it. (Lohner) Try out with "vt100" entry, cm replaced with cX.
7 When an include file starts with "../", the check for already visiting this file doesn't work. Need to simplify the file name.
7 The names and comments for the arguments of do_browse() are confusing. "dflt" isn't the default file name when "initdir" is not NULL and "initdir" is the default path to be used.
7 When 'scrolloff' is exactly half the window height, "j" causes a scroll of two lines at a time. "k" doesn't do this. (Cory T. Echols)
8 When write_viminfo() is used while there are many orphaned viminfo tempfiles writing the viminfo file fails. Give a clear error message so that the user knows he has to delete the files.
7 It's possible to redefine a script-local function with ":func <SNR>123_Test()". (Krishna) Disallow this.

I can't reproduce these (if you can, let me know how!):
9 NT 4.0 on NTFS file system: Editing ".bashrc" (drag and drop), file disappears. Editing ".xyz" is OK. Also, drag&drop only works for three files. (McCollister)

Problems that will (probably) not be solved:
- GTK: when using the popup menu with spelling suggestions and releasing the right mouse button before the menu appears selecting an item with the right mouse button has no effect. GTK does not produce an event for this.
- GTK 2: Cannot use the file selector. When using it many things become slow. This is caused by some code in GTK that writes ~/.recently-used.xbel every time an event is handled. It assumes the main loop is never quit, which is a wrong assumption. Also, it overwrites the file with different file permissions, which is a privacy issue. This needs to be fixed in GTK. A solution in Vim would be really complicated. (2008 Jul 31) This appears to be fixed in Vim 7.3.
- xterm title: The following scenario may occur (esp. when running the Vim test script): Vim 1 sets the title to "file1", then restores the title to "xterm" with an ESC sequence when exiting. Vim 2 obtains the old title with an X library call, this may result in "file1", because the window manager hasn't processed the "xterm" title yet. Can apparently only be worked around with a delay.
- In a terminal with 'mouse' set such that the mouse is active when entering a command line, after executing a shell command that scrolls up the display and then pressing ":": Selecting text with the mouse works like the display wasn't scrolled. Vim doesn't know how much the external command scrolled up the display. Use Shift to select text.
- X windows: When $DISPLAY points to a X server where there is no access permission, trying to connect to the X server causes an error message. XtOpenDisplay() prints this directly, there is no way to avoid it.
- X windows: Setting 'guifontset' to an illegal value sometimes crashes Vim. This is caused by a fault in a X library function, can't be solved in Vim.
- Win32 tcl: has("tcl") hangs when the tcl84.dll is from cygwin.
- Motif: When adding a menu item "Find this &Symbol", the "s" in "this" will

```
    be underlined, instead of in "Symbol".  Motif doesn't let us specify which
    character gets the highlighting.
-   Moving the cursor removes color in color-xterm.  This is a color-xterm
    problem!  color-xterm ver. 6.1 beta 3 and later work properly.
-   In zsh, "gvim&" changes the terminal settings.  This is a zsh problem.
    (Jennings)
-   Problem with HPterm under X: old contents of window is lost (Cosentino).
-   Amiga: When using quickfix with the Manx compiler we only get the first 25
    errors.  How do we get the rest?
-   Amiga: The ":cq" command does not always abort the Manx compiler.  Why?
-   Linux: A file with protection r--rw-rw- is seen readonly for others.  The
    access() function in GNU libc is probably wrong.
-   When doing a CTRL-Z and typing a command for the shell, while Vim is busy
    (e.g. writing a file), the command for the shell is sometimes eaten by Vim,
    because the terminal mode is changed from RAW to CBREAK.
-   An old version of GNU tgoto can't handle the terminfo code for "AF".  The
    "%p1" is interpreted as "%p" and "1", causing color not to be working.
    Fix: Change the "%p1" in the "AF" and "AB" terminfo entries to "%p".
    (Benzinger).
-   When running an external command from the GUI, typeahead is going to that
    program, not to Vim.  It looks like the shell eats the characters, Vim
    can't get back what the external command didn't use.
-   Win32 GUI: Error code from external command not returned in shell_error.
    It appears that cmd.exe and command.com don't return an error code.
-   Win32 GUI: The Toolbar is a bit too high when the flat style is being
    used.  We don't have control over the height of the Toolbar.
-   Win32: All files created on the day of switching from winter to summer
    time cause "changed since editing started" messages.  It goes away when
    the file is written again the next day, or the timezone is adjusted.
    DJGPP version is OK. (Zaimi)  Looks like a problem with the Win32 library.
    Rebooting doesn't help.  Time stamps look OK in directory. (Penn)
    Is this on FAT (stores wall clock time) or NTFS (stores UTS)?
-   Win32, MS-Windows XP: $HOME uses the wrong drive when the user profiles
    are not on the boot disk.  This is caused by a wrong value of $HOMEDRIVE.
    This is a bug in XP, see MSKB article 818134.
-   Win32, MS-Windows: expanding plugin/**/*.vim also picks up
    dir/ctags.vim,v.  This is because the short file name is something like
    "ctags~1.vim" and that matches the pattern.
-   SunOS 5.5.1 with Motif: The file open dialog does not have a horizontal
    scroll bar for the "files" selection.  This is a problem in the Motif
    libraries, get a patch from Sun.
-   Solaris 2.6 with GTK and Perl: gvim crashes when started.  Problem with X
    input method called from GDK code.  Without Perl it doesn't crash.
-   VMS: Vimdiff doesn't work with the VMS diff, because the output looks
    different.  This makes test 47 fail.  Install a Unix-compatible diff.
-   Win32 GUI: mouse wheel always scrolls rightmost window.  The events arrive
    in Vim as if the rightmost scrollbar was used.
-   GTK with Gnome: Produces an error message when starting up:
        Gdk-WARNING **: locale not supported by C library
    This is caused by the gnome library gnome_init() setting $LC_CTYPE to
    "en_US".  Not all systems support this locale name, thus causing the
    error.  Hopefully a newer version of GTK/Gnome fixes this problem.
-   GTK 2: With this mapping the hit-enter prompt is _sometimes_ below the
    screen, at other times there is a grey area below the command line:
        :nmap <F11> :if &guioptions=~'m' \| set guioptions-=m \| else \| set
guioptions+=m \| endif<cr>
-   GTK: When pasting a selection from Vim to xclipboard gvim crashes with a
    ABRT signal.  Probably an error in the file gdkselection.c, the assert
    always fails when XmbTextListToTextProperty() fails. (Tom Allard)
-   GTK 2: gives an assertion error for every non-builtin icon in the toolbar.
    This is a GTK 2.4.x bug, fixed in GTK 2.4.2. (Thomas de Grenier de Latour)
-   When using an xterm that supports the termresponse feature, and the 't_Co'
```

```
    termcap option was wrong when Vim started, it will be corrected when the
    termresponse is received.  Since the number of colors changes, the
    highlighting needs to be initialized again.  This may cause colors defined
    in the vimrc file to be lost.
-   On Windows NT 4.0 the number of files passed to Vim with drag&drop and
    "Edit with Vim" is limited.  The maximum command line length is 255 chars.

---------------------  extensions and improvements ----------------------
                                                      *extensions-improvements*

Most interesting new features to be added when all bugs have been fixed:
-   Using ":exe edit fname" has escaping problems.  Use ":edit ++(fname)".
    Thus use "++=" to give arguments as expressions, comma separated as if
    calling a function.
    With options: ":edit ++(['!', '++enc=abc'], ['+/pat'], fname)".
    Alternative: Make a function for Ex commands: cmd_edit().
-   Add COLUMN NUMBERS to ":" commands ":line1,line2[col1,col2]cmd".  Block
    can be selected with CTRL-V.  Allow '$' (end of line) for col2.
-   ECLIPSE plugin.  Problem is: the interface is very complicated.  Need to
    implement part in Java and then connect to Vim.  Some hints from Alexandru
    Roman, 2004 Dec 15.  Should then also work with Oracle Jdeveloper, see JSR
    198 standard http://www.jcp.org/en/jsr/detail?id=198.
    Eclim does it: http://eclim.sourceforge.net/  (Eric Van Dewoestine)
    Plugin that uses a terminal emulator: http://vimplugin.sf.net
    And another one: http://www.satokar.com/viplugin/index.php
-   STICKY CURSOR: Add a way of scrolling that leaves the cursor where it is.
    Especially when using the scrollbar.  Typing a cursor-movement command
    scrolls back to where the cursor is.
-   Scroll commands by screen line. g CTRL-E and g CTRL-Y ?  Requires the
    first line to be able to start halfway.
8   Add a command to jump to a certain kind of tag.  Allow the user to specify
    values for the optional fields.  E.g., ":tag size type=m".
    Also allow specifying the file and command, so that the result of
    taglist() can be used.
-   X11: Make it possible to run Vim inside a window of another program.
    This can be done with XReparentWindow().  But how exactly?


Documentation:
8   List of Vim runtime directories.  dotvim.txt from Charles Campbell, 2007
    Feb 20.
8   The GUI help should explain the Find and Find/Replace dialogs.  Add a link
    to it from ":promptrepl" and ":promptfind".
8   List of options should mention whether environment variables are expanded
    or not.
8   Extend usr_27.txt a bit. (Adam Seyfarth)
9   Make the Reference Manual more precise.  For each command mention:
    - change to cursor position and curswant
    - if it can be undone (u/CTRL-R) and redone (.)
    - how it works for folded lines
    - how it works with multi-byte characters
9   In change.txt, remark about Javadoc isn't right.  Right alignment would
    work too.
8   Spread the windows commands over the other files.  For example, ":stag"
    should be with ":tag".  Cross-link with tags to avoid too much double
    text.
8   Add tags for all features, e.g. "gui_running".
7   MS-Windows: When a wrong command is typed with an ALT key, give a hint to
    look at the help for 'winaltkeys'.
7   Add a help.vim plugin that maps <Tab> to jump to the next tag in || and
    <C-Tab> (and <S-Tab>) to the previous tag.
    Patch by Balazs Kezes, 2007 Dec 30.  Remark from A. Politz.
```

```
-   Check text editor compendium for vi and Vim remarks.


Help:
-   First try using the ":help" argument literally, before using it as a
    pattern.  And then match it as part of a tag.
-   When a help item has multiple matches make it possible to use ":tn" to go
    to the other matches.
-   Support a way to view (and edit) .info files.
-   Implement a "sticky" help window, some help text lines that are always
    displayed in a window with fixed height. (Guckes)  Use "~/.vimhelp" file,
    user can edit it to insert his favorite commands, new account can contain a
    default contents.
-   Make 'winminheight' a local option, so that the user can set a minimal
    height for the help window (and other windows).
-   ":help :s^I" should expand to ":help :substitute".
-   Make the help key (<F1>) context sensitive?
-   Learn mode: show short help while typing commands.


User Friendlier:
8   Windows install with install.exe: Use .exe instead of .bat files for
    links, so that command line arguments are passed on unmodified? (Walter
    Briscoe)
8   Windows install: Be able to associate Vim with a selection of file types?
8   Windows uninstall: Have uninstal.c delete the vimfiles directories that
    dosinst.c creates.  List the contents of the directory (recursively) if
    the user asks for it.  Requires an implementation of "rm -rf".
8   Remember the name of the vimrc file that was used (~/.vimrc, $VIM/_vimrc,
    $HOME/_vimrc, etc.) and add "edit vimrc" to the File menu.
-   Add a way to save local settings and mappings into a new plugin file.
    ":mkplugin <file>"?
-   Add mappings local to a window: ":map <window> ..."?
9   Add buffer-local menu.  Should offer a choice between removing the menu or
    disabling it.  Be careful that tear-offs don't disappear (keep one empty
    item?).
    Alternative: use BufEnter and BufLeave autocommands.
8   make a vimtutor script for Amiga and other systems.
7   When Vim detects a file is being edited elsewhere and it's a gvim session
    of the same user it should offer a "Raise" button, so that the other gvim
    window can be displayed. (Eduard)
8   Support saving and restoring session for X windows?  It should work to do
    ":mksession" and use "-S fname" for the restart command.  The
    gui_x11_wm_protocol_handler() already takes care of the rest.
    global_event_filter() for GTK.


Tab pages:
9   GUI implementation for the tab pages line for other systems.
7   GUI: Control over the appearance of the text in the labels (bold, color,
    font, etc.)
8   Make GUI menu in tab pages line configurable.  Like the popup menu.
8   balloons for the tab page labels that are shortened to show the full path.
7   :tabdup      duplicate the tab with all its windows.
7   Option to put tab line at the left or right?  Need an option to specify
    its width.  It's like a separate window with ":tabs" output.
8   Add local options for each tab page?  E.g., 'diffopt' could differ between
    tab pages.
7   Add local highlighting for each tab page?
7   Add local directory for tab pages?  How would this interfere with
    window-local directories?
```

```
Spell checking:
-   Support more regions? Caolan McNamara argues it's needed for es_XX.
    https://bugzilla.redhat.com/bugzilla/show_bug.cgi?id=219777
-   Unicode defines another quote character: 0x2019.  Use it as an equivalent
    of a single quote, thus use it as a word character like a quote and match
    with words, replacing the curly quote with a single quote.
-   Could filter &eacute; things for HTML before doing spell checking.
    Similarly for TeX.
-   The Hungarian spell file uses four extra characters in the FOL/UPP/LOW
    items than other spell files with the ISO-8859-2 encoding, that causes
    problem when changing 'spelllang'.  There is no obvious way to fix this.
-   Considering Hunspell 1.1.4:
    What does MAXNGRAMSUGS do?
    Is COMPLEXPREFIXES necessary when we have flags for affixes?
-   Support spelling words in CamelCase as if they were two separate words.
    Requires some option to enable it. (Timothy Knox)
-   There is no Finnish spell checking file.  For openoffice Voikko is now
    used, which is based on Malaga: http://home.arcor.de/bjoern-beutel/malaga/
    (Teemu Likonen)
8   ":mkspell" still takes much too long in Hungarian dictionary from
    hunspell.  Only solution appears to be to postpone secondary suffixes.
8   Handle postponed prefix with COMPOUNDPERMITFLAG or COMPOUNDFORBIDFLAG.
    WFP_COMPPERMIT and WFP_COMPFORBID
8   implement use of <compoptions> in .spl file:
    implement CHECKCOMPOUNDREP: when a compound word seems to be OK apply REP
    items and check if the result is a valid word.
    implement CHECKCOMPOUNDDUP
    implement CHECKCOMPOUNDTRIPLE
    Add CHECKCOMPOUNDCASE: when compounding make leading capital lower case.
    How is it supposed to work?
-   Add a command the repeats ]s and z=, showing the misspelled word in its
    context.  Thus to spell-check a whole file.
-   suggestion for "KG" to "kg" when it's keepcase.
-   For flags on affixes: Use a "AFFCOMPSET" flag; means the compound flags of
    the word are not used.
-   Support breakpoint character ? 0xb7 and ignore it?  Makes it possible to
    use same wordlist for hyphenation.
-   Compound word is accepted if nr of words is <= COMPOUNDWORDMAX OR nr of
    syllables <= COMPOUNDSYLMAX.  Specify using AND in the affix file?
-   NEEDCOMPOUND also used for affix?  Or is this called ONLYINCOMPOUND now?
    Or is ONLYINCOMPOUND only for inside a compound, not at start or end?
-   Do we need a flag for the rule that when compounding is done the following
    word doesn't have a capital after a word character, even for Onecap words?
-   New hunspell home page: http://hunspell.sourceforge.net/
    - Version 1.1.0 is out now, look into that.
    - Lots of code depends on LANG, that isn't right.  Enable each mechanism
      in the affix file separately.
    - Example with compounding dash is bad, gets in the way of setting
      COMPOUNDMIN and COMPOUNDWORDMAX to a reasonable value.
    - PSEUDOROOT == NEEDAFFIX
    - COMPOUNDROOT -> COMPOUNDED?  For a word that already is a compound word
            Or use COMPOUNDED2, COMPOUNDED3, etc.
-   CIRCUMFIX: when a word uses a prefix marked with the CIRCUMFIX flag, then
    the word must also have a suffix marked with the CIRCUMFIX flag.  It's a
    bit primitive, since only one flag is used, which doesn't allow matching
    specific prefixes with suffixes.
    Alternative:
        PSFX {flag} {pchop} {padd} {pcond} {schop} {sadd}[/flags] {scond}
    We might not need this at all, you can use the NEEDAFFIX flag and the
    affix which is required.
-   When a suffix has more than one syllable, it may count as a word for
```

```
    COMPOUNDWORDMAX.
-   Add flags to count extra syllables in a word.  SYLLABLEADD1 SYLLABLEADD2,
    etc.?  Or make it possible to specify the syllable count of a word
    directly, e.g., after another slash: /abc/3
-   MORPHO item in affix file: ignore TAB and morphological field after
    word/flags and affix.
-   Implement multiple flags for compound words and CMP item?
    Await comments from other spell checking authors.
-   Also see tklspell: http://tkltrans.sourceforge.net/
8   Charles Campbell asks for method to add "contained" groups to existing
    syntax items (to add @Spell).
    Add ":syntax contains {pattern} add=@Spell" command?  A bit like ":syn
    cluster" but change the contains list directly for matching syntax items.
-   References: MySpell library (in OpenOffice.org).
        http://spellchecker.mozdev.org/source.html
        http://whiteboard.openoffice.org/source/browse/whiteboard/lingucomponent/
source/spellcheck/myspell/
      author: Kevin Hendricks <kevin.hendricks@sympatico.ca>
8   It is currently not possible to mark "can not" as rare, because "can" and
    "not" are good words.  Find a way to let "rare" overrule "good"?
8   Make "en-rare" spell file?  Ask Charles Campbell.
8   The English dictionaries for different regions are not consistent in their
    use of words with a dash.
7   Insert mode completion mechanism that uses the spell word lists.
8   Add hl groups to 'spelllang'?
        :set spelllang=en_us,en-rare/SpellRare,en-math/SpellMath
    More complicated: Regions with different languages?  E.g., comments
    in English, strings in German (po file).


Diff mode:
9   When making small changes, e.g. deleting a character, update the diff.
    Possibly without running diff.
8   Also show difference with the file when editing started?  Should show what
    can be undone. (Tom Popovich)

Folding:
    (commands still available: zI zJ zK zp zP zq zQ zV zy zY;
    secondary: zB zS zT zZ, z=)
8   Vertical folds: looks like vertically split windows, but the cursor moves
    through the vertical separator, separator moves when scrolling.
8   Add "z/" and "z?" for searching in not folded text only.
8   When a closed fold is displayed open because of 'foldminlines', the
    behavior of commands is still like the fold is closed.  How to make the
    user aware of this?
8   Add an option 'foldskip' with values like 'foldopen' that specifies which
    commands skip over a closed fold.
8   "H" and "L" count buffer lines instead of window lines. (Servatius Brandt)
8   Add a way to add fold-plugins.  Johannes Zellner has one for VB.
7   When using manual folding, the undo command should also restore folds.
-   Allow completely hiding a closed fold.  E.g., by setting 'foldtext' to an
    empty string.  Require showing a character in 'foldcolumn' to avoid the
    missing line goes unnoticed.
    How to implement this?
-   When pressing the down arrow of a scrollbar, a closed fold doesn't scroll
    until after a long time.  How to make scrolling with closed folds
    smoother?
-   When creating a session, also store folds for buffers in the buffer list,
    using the wininfo in wi_folds.
-   When currently editing the first file in the argument list the session
    file can contain:
        args version.c main.c
```

```
        edit version.c
    Can editing version.c twice be avoided?
-   'foldmethod' "textobject": fold on sections and paragraph text objects.
-   "zuf": undo change in manual fold. "zUf" redo change in manual fold.  How
    to implement this?
-   "zJ" command: add the line or fold below the fold in the fold under the
    cursor.
-   'foldmethod' "syntax": "fold=3" argument: set fold level for a region or
    match.
-   Apply a new foldlevel to a range of lines. (Steve Litt)

Multi-byte characters:
-   When editing a file with both utf-8 and latin1 text Vim always falls back
    to latin1.  Add a command to convert the latin1 characters to utf-8?
        :unmix utf-8,latin1 filename
    Would only work when 'encoding' is utf-8.
9   When the tail byte of a double-byte character is illegal (e.g., a CR), the
    display is messed up (Yasuhiro Matsumoto).  Should check for illegal
    double-byte characters and display them differently (display each single
    byte).
9   'fenc' in modeline problem: add option to reload the file when 'fenc' is
    set to a different value in a modeline?  Option can be default on.  Could
    it be done with an autocommand?
8   Add an item in 'fileencodings' to check the first lines of a file for
    the encoding.  See Python PEP: http://www.python.org/peps/pep-0263.html.
    To avoid getting a wrong encoding only accept something Emacs-like:
    "-*- coding: enc-na_me.foo -*-" and "-*- coding= enc-na_me.foo -*-"
    Match with "-\*-\s*coding[:=]\s*\([::word::-_.]\+\)\s*-\*-" and use first
    item.
8   Add an item in 'fileencodings' to check the first line of an XML file for
    the encoding.  <?xml version="1.0" encoding="UTF-8"?>  Or "charset=UTF-8"?
    For HTML look for "charset=utf-8".
8   The quickfix file is read without conversion, thus in 'encoding'.  Add an
    option to specify the encoding of the errorfile and convert it.  Also for
    ":grep" and ":helpgrep".
    More generic solution: support a filter (e.g., by calling a function).
8   When a file was converted from 'fileencoding' to 'encoding', a tag search
    should also do this on the search pattern. (Andrzej M. Ostruszka)
8   When filtering changes the encoding 'fileencoding' may not work.  E.g.,
    when using xxd and 'fileencoding' is "utf-16".  Add an option to set a
    different fileencoding for filter output?
7   When converting a file fails, mention which byte could not be converted,
    so that the user can fix the problem.
8   Add configure option to be able to disable using the iconv library. (Udo
    Schweigert)
9   'aleph' should be set to 1488 for Unicode. (Zvi Har'El)
8   Should add test for using various commands with multi-byte characters.
8   'infercase' doesn't work with multi-byte characters.
8   toupper() function doesn't handle byte count changes.
7   Searching and composing characters:
    When searching, should order of composing characters be ignored?
    Add a special item to match with a composing character, so that composing
    characters can be manipulated.
8   Should implement 'delcombine' for command line editing.
8   Detect overlong UTF-8 sequences and handle them like illegal bytes.
8   ":s/x/\u\1/" doesn't work, making uppercase isn't done for multi-byte
    characters.
8   UTF-8: "r" in Visual mode doesn't take composing characters.
8   UTF-8: When there is a precomposed character in the font, use it instead
    of a character and a composing character.  See xterm for an example.
7   When a character can't be displayed, display its digraph instead.
    'display' option to specify this.
```

```
7   Use ideas for nl_langinfo() from Markus Kuhn in enc_default():
    (www.cl.cam.ac.uk/~mgk25/ucs/langinfo.c)
-   GTK and Win32: Allow selecting fonts for 'guifontset' with the
    fontselector somehow.
-   GTK and Win32: make it possible to set the font for the menu to make it
    possible to have 'encoding' different from the current locale.
-   dbcs_class() only works for Japanese and Korean.  Implement this for
    other encodings.  The "euc-jp" and "euc-kr" choices might be wrong.
-   Find some way to automatically select the right GUI font or fontset,
    depending on the default value of 'encoding'.
    Irrelevant in the GTK+ 2 GUI so long as UTF-8 is used.
    For Windows, the charset_pairs[] table could be used.  But how do we know
    if a font exists?
-   Do keyboard conversion from 'termencoding' to 'encoding' with
    convert_input() for Mac GUI.
-   Add mnemonics from RFC1345 longer than two characters.
    Support CTRL-K _{mnemonic}_
-   Make 'breakat' accept multi-byte characters.  Problem: can't use a lookup
    table anymore (breakat_flags[]).
    Simplistic solution: when 'formatoptions' contains "m" also break a line
    at a multi-byte character >= 0x100.
-   Add the possibility to enter mappings which are used whenever normal text
    could be entered.  E.g., for "f" command.  But not in Normal mode.  Sort
    of opposite of 'langmap'.  Use ":amap" command?
-   When breaking a line, take properties of multi-byte characters into
    account.  The "linebreak" program from Bruno Haible can do it:
    ftp://ftp.ilog.fr/pub/Users/haible/gnu/linebreak-0.1.tar.gz
    But it's very complicated...


Printing:
7   Implement "undercurl" for printing.
-   Add "page width" to wrap long lines.
-   Win32: use a font dialog for setting 'printfont'.  Can reuse the code for
    the 'guifont' dialog, put the common code in a separate function.
-   Add the file timestamp to the page header (with an option). (George
    Reilly)
-   Win32: when 'printfont' is empty use 'guifont'.
-   Unix: Use some dialog box to do the obvious settings (paper size, printer
    name, portrait/landscape, etc).
-   PostScript: Only works for text that can be converted to an 8-bit
    character set.  How to support Unicode fully?
-   Allow specifying the paper size, instead of using a standard size.  Same
    units as for the margins.
-   Support right-to-left text?
8   Make the foreground color darkening function preserve the hue of the
    color.


Syntax highlighting:
8   Make ":syn off" use 'runtimepath' instead of $VIMRUNTIME. (Gary Johnson)
    Should do the same for ":syn on" and ":syn manual".
8   Support "containedin" argument for ":syn include", so that the defined
    cluster can be added to existing syntax items.
8   C syntax: Don't highlight {} as errors inside () when used like this:
    "({ something })", often used in GCC code.
7   Add a "startgroup" to a region.  Used like "nextgroup" inside the region,
    preferred item at the start of the region. (Charles Campbell)
8   When editing a new file without a name and giving it a name (by writing
    it) and 'filetype' is not set, detect the filetype.  Avoid doing it for
    ":wq file".
7   For "nextgroup" we have skipwhite, skipnl and skipempty.  It would be
```

```
    really nice to be able to skip with a pattern.  Or skip with a syntax
    group. (Nikolai Weibull, 2007 Feb 27)
8   Make conversion to HTML faster (Write it in C or pre-compile the script).
9   There is still a redraw bug somewhere.  Probably because a cached state is
    used in a wrong way.  I can't reproduce it...
7   Be able to change only the background highlighting.  Useful for Diff* and
    Search highlighting.
7   When 'number' is set highlight the number of the current line.
    Must be enabled with an option, because it slows down display updating.
8   Allow the user to add items to the Syntax menu sorted, without having to
    change this for each release.
8   Add a "matchcontains" for regions: items contained in the start or end
    pattern, but not in the body.
8   Add a "keepend-contained" argument: Don't change the end of an item this
    one is contained in.  Like "keepend" but specified on the contained item,
    instead of the containing item.
8   cpp.vim: In C++ it's allowed to use {} inside ().
8   Some syntax files set 'iskeyword', they should use "syn iskeyword".
    Also need a separate 'iskeyword' for the command line, e.g., in a help
    window ":e /asdf/asdf/" CTRL-W works different.
8   Add specific syntax item to match with parens/braces that don't have a
    "%" match.  :syntax nomatch cMatchError (,{,[,),},] [contained]
8   Highlight the text between two matching parens (e.g., with a grey
    background) when on one of the parens or in between them.
    Option for the matchparen plugin?
8   When using a cterm, and no ctermfg or ctermbg are defined, use start/stop
    sequences.  Add remark in docs that :if 'term' == "term-name" should be
    used.
8   Add @spell cluster to String and Comment groups for many languages.  Will
    allow spell checking. (Fleiner)
8   When listing syntax items, try to sort the keywords alphabetically.  And
    re-insert the [] if possible.
8   Make it possible to use color of text for Visual highlight group (like for
    the Cursor).
8   It would be useful to make the highlight group name an expression.  Then
    when there is a match, the expression would be evaluated to find out what
    highlight group to use.  Could be used to check if the shell used in a
    password file appears in /etc/shells. (Nikolai Weibull)
        syn match =s:checkShell(v:match) contained 'pattern'
8   Make it possible to only highlight a sub-expression of a match.  Like
    using "\1" in a ":s" command.
8   Support for deleting syntax items:
    :syn keyword cTodo remove this
    :syn match cTodo remove "pattern"
    :syn region cString remove start="this" end="that"
8   Add possibility to sync on something else, when the syncing in one way
    doesn't find match.  For HTML: When no {script} is found, try looking for
    a '<'.  (Fleiner)
7   Replace the synchronizing method with a state machine specification?
    Should be able to start at any line in the file, search forwards or
    backwards, and use the result of matching a pattern.
7   Use parsing like awk, so that e.g., a ( without a matching ) can be
    detected.
8   Make it possible to use "inverted" highlighting, invert the original
    character.  For Visual mode.  (xterm-selection already does this).
8   Highlight non-printable characters with "SpecialChar", linked to
    "Special".  Display them with the digraph characters, if possible.
8   Highlight the clipboard-selection with a highlight group.
8   Be able to reset highlighting to its original (default) values.
7   Be able to write current highlighting to a file as commands, similar to
    ":mkvimrc".
8   Improve c.vim:
```

```
    - Add check for unterminated strings, with a variable to switch it on:
      "c_strict_ansi".
    - Detect unbalanced "#endif".  Requires looking back a long way...
8   Add an option to restrict the updating of syntax highlighting to the
    current line while in Insert mode.
8   When guessing value of 'background', the syntax file has already been
    loaded (from the .gvimrc).  After changing 'background', load it again?
8   Add ":syn resync" command, to re-parse the whole file until the current
    display position.
8   Should support "me" offset for a region start pattern.  To be used to
    allow searching for the end pattern inside the match of the end pattern.
    Example: syn region pikeXX start="([^{]" end=")" should work on "()".
8   When using a regexp for "contains=", should delay matching with it until
    redrawing happens.  Set a flag when a group is added, check this flag when
    highlighting starts.
7   It's possible for an item to be transparent, so that the colors of an item
    lower on the stack is used.  Also do this with highlighting, so that the
    user can set transparent highlighting?  E.g. a number in a C comment would
    get the color of a comment, a number in an assignment Normal. (Nikolai
    Weibull)
7   Add "semitrans": Add highlighting.  E.g., make the text bold, but keep the
    colors.  And add colors, so that Green+Red becomes Yellow.
    E.g. for this html:
        <B> bold text <I> italic+bold text </B> italic text </I>
7   CTRL-] checks the highlight group for finding out what the tag is.
7   Add an explanation how a list of words can be used to highlight misspelled
    words.
8   Add more command line completion for :syntax.
8   Add more command line completion for :highlight.
7   Should find a better way to parse the :syntax and :highlight commands.
    Use tables or lists that can be shared by parsing for execution and
    completion?
8   Add ColorSchemePost autocommand event, so that scripts can set up their
    highlighting. (Salman Halim)
7   Add a few sets of colors (e.g. Borland Turbo C one).  With a menu to
    select one of the sets.
8   Add offsets to sub-matches: "\(a*\) *"he=e1-1
    'e' is end of match 'e1' is end of sub-match 1, 's2' is start of submatch
    2, etc.
8   In Insert mode, when there are typeahead characters, postpone the
    highlighting (for "." command).
8   Syncing on comments isn't 100% correct when / / lines mix with / * and * /.
    For example: What about a line that starts with / / and contains * /?
8   Ignore / * and  * / inside strings, when syncing.
7   Build a few more syntax files from the file "/usr/share/misc/vgrindefs":
    ISP, LDL, Icon, ratfor.  And check "nedit/source/highlight.c".
6   Add possibility to have background color continue until the right edge of
    the window.  Useful for comment blocks and function headings. (Rogall)
-   Make it possible to add "contains" items for all items in a group.  Useful
    when extending an already existing syntax file.
-   Add line-continuation pattern for non-syncing items too?
-   Add possibility to highlight the whole line, including the right margin
    (for comment blocks).
-   Add 'hlmatch' option: List of flags:
    'c': highlight match for character under the cursor.
    'b': highlight the previous (, and its match.
    'a': highlight all text from the previous ( until its match.
         Also for {}, <>, etc.?
    'e': highlight all braces without a match (slow?)
    OR: add an argument "cursor" to the syntax command, which means that the
    region/match/keyword is only highlighted when the cursor is on it.
    (Campbell)
```

```
    Or do it like Elvis: define text objects and how to highlight them around
    the cursor. (Iain Truskett)
7   Make it possible to use all words in the tags files as Keyword.
    Can also be done with a script (but it's slow).
7   Make it possible to call a ":" command when a match is found.  Should
    allow for adding keywords from the text (e.g. variables that are set).
    And allows for sections with different highlighting.
7   Add highlight group for commandline: "Commandline".  Make sure it
    highlights the command line while typing a command, and any output from
    messages.  And external commands?
8   Make a version that works like less, but with highlighting: read stdin for
    text, exit at end of file, don't allow editing, etc.  moreim?  lessim?
7   SpecialKey highlighting overrules syntax highlighting.  Can't give an
    unprintable char another color.  Would be useful for ^M at end of line.


Vim script language:
8   Make the filename and line number available to script functions, so that
    they can give useful debugging info.  The whole call stack would be ideal.
    At least use this for error messages.
7   Execute a function with standard option values.  No need to save and
    restore option values.  Especially useful for new options.  Problem: how
    to avoid a performance penalty (esp. for string options)?
8   Add referring to key options with "&t_xx".  Both for "echo &t_xx" and
    ":let &t_xx =".  Useful for making portable mappings.
-   Add ":let var ?= value", conditional assignment.  Patch by Dave Eggum,
    2006 Dec 11.
-   range for ":exec", pass it on to the executed command.  (Webb)
8   ":{range}source": source the lines from the current file.
        You can already yank lines and use :@" to execute them.
        Most of do_source() would not be used, need a new function.
        It's easy when not doing breakpoints or profiling.
    Requires copying the lines into a list and then creating a function to
    execute lines from the list.  Similar to getnextac().
7   ":include" command: just like ":source" but doesn't start a new scriptID?
    Will be tricky for the list of script names.
8   Have a look at VSEL.  Would it be useful to include? (Bigham)
8   Have a prefix for a function to make it unique.  When using packages it
    can be the plugin name.
    Perhaps also have a way to remove everything that the package added?
    including autocommands.
7   Pre-parse or compile Vim scripts into a bytecode.
    1. Put the bytecode with the original script, with an ":if
       has('bytecode')" around it, so that it's only used with a Vim that
       supports it.  Update the code with a command, can be used in an
       autocommand.
    2. Use a ".vic" file (like Python use .pyc).  Create it when writing a
       .vim file.  Problem: distribution.
    3. Use a cache directory for each user.  How to recognize which cached
       file belongs to a sourced script?
7   Add argument to winwidth() to subtract the space taken by 'foldcolumn',
    signs and/or 'number'.
6   Add ++ and -- operators?  They only work on variables (lvals), how to
    implement this?
8   Add functions:
        has(":command")         Check if ":command" works.  compare function
                                with "ex_ni".  E.g. for ":simalt".
        escape()                Add argument to specify what to escape with.
        modestack()             Instead of just the current mode return the
                                stack of Insert / CTRL-O / :normal things.
        realname()              Get user name (first, last, full)
                                user_fullname() patch by Nikolai Weibull, Nov
```

```
                              3 2002
                              Only add this when also implemented for
                              non-Unix systems, otherwise a shell cmd could
                              be used.
                              get_user_name() gets login name.
        menuprop({name}, {idx}, {what})
                              Get menu property of menu {name} item {idx}.
                              menuprop("", 1, "name") returns "File".
                              menuprop("File", 1, "n") returns "nmenu
                              File.Open..." argument.
                              Patch by Ilya Sher, 2004 Apr 22
                              Return a list of menus and/or a dictionary
                              with properties instead.
        mapname({idx}, mode)  return the name of the idx'th mapping.
                              Patch by Ilya Sher, 2004 Mar 4.
                              Return a list instead.
        char2hex()            convert char string to hex string.
        crypt()               encrypt string
        decrypt()             decrypt string
        base64enc()           base 64 encoding
        base64dec()           base 64 decoding
        attributes()          return file protection flags "drwxrwxrwx"
        filecopy(from, to)    Copy a file
        shorten(fname)        shorten a file name, like home_replace()
        perl(cmd)             call Perl and return string
        inputrl()             like input() but right-to-left
        typed()               return the characters typed and consumed (to
                              find out what happened)
        virtualmode()         add argument to obtain whether "$" was used in
                              Visual block mode.
        getacp()              Win32: get codepage (Glenn Maynard)
        deletebufline()       delete line in any buffer
        appendbufline()       append line in any buffer
        libcall()             Allow more than one argument.
        libcallext()          Like libcall(), but using a callback function
                              to allow the library to execute a command or
                              evaluate an expression.
7   Make bufname("'0") return the buffer name from mark '0.  How to get the
    column and line number?  col("'0") currently returns zero.
8   argc() returns 0 when using "vim -t tag".  How to detect that no file was
    specified in any way?  To be able to jump to the last edited file.
8   Pass the command line arguments to Vim scripts in some way.  As v:args
    List?  Or extra parameter to argv()?
8   Add command arguments with three dashes, passed on to Vim scripts.
9   Add optional arguments to user functions:
        :func myFunc(arg1, arg2, arg3 = "blah", arg4 = 17)
6   User functions: Functions local to buffer "b:func()"?
8   For Strings add ":let var[{expr}] = {expr}".  When past the end of "var"
    just ignore.
8   The "= register should be writable, if followed by the name of a variable,
    option or environment variable.
8   ":let &option" should list the value of the option.
8   ":let Func().foo = value" should work, also when "foo" doesn't exist.
    Also: ":let Func()[foo] = value" should work.  Same for a List.
7   Add synIDlist(), making the whole list of syntax items on the syntax stack
    available as a List.
8   Add autocommand-event for when a variable is changed:
        :au VarChanged {varname} {commands}
8   Add "has("gui_capable")", to check if the GUI can be started.
8   Add possibility to use variables like registers: characterwise (default),
    linewise (when ending in '\n'), blockwise (when ending in '\001').  reg0,
    rega, reg%, etc.  Add functions linewise({expr}), blockwise({expr}) and
```

```
    charwise({expr}).
7   Make it possible to do any command on a string variable (make a buffer
    with one line, containing the string).  Maybe add an (invisible) scratch
    buffer for this?
        result = scratch(string, command)
        result = apply(string, command)
        result = execute(string, command)
    "command" would use <> notation.
    Does scratch buffer have a number?  Or re-use same number?
7   Add function to generate unique number (date in milliseconds).


Robustness:
6   Add file locking.  Lock a file when starting to edit it with flock() or
    fcntl().  This patch has advisory file locking while reading/writing
    the file for Vim 5.4: ~/vim/patches/kahn_file_locking .
    The patch is incomplete (needs support for more systems, autoconf).
    Andy doesn't have time to work on it.
    Disadvantage: Need to find ways to gracefully handle failure to obtain a
    lock.  When to release a lock: When buffer is unloaded?


Performance:
7   For string variables up to 3 bytes don't allocate memory, use v_list
    itself as a character array.  Use VAR_SSTRING (short string).
7   Add 'lazysize' option: Above this size Vim doesn't load everything before
    starting to edit a file.  Things like 'fileencodings' only work up to this
    size, modelines only work at the top.  Useful for large log files where
    you only want to look at the first few pages.  Use zero to disable it.
8   move_lines() copies every line into allocated memory, making reloading a
    buffer a lot slower than re-editing the file.  Can the memline be locked
    so that we don't need to make a copy?  Or avoid invoking ml_updatechunk(),
    that is taking a lot of time.  (Ralf Wildenhues, 2008 Jul 7)
    With a patch, but does it work?
8   Turn b_syn_ic and b_syn_containedin into b_syn_flags.
9   Loading menu.vim still takes quite a bit of time.  How to make it faster?
8   in_id_list() takes much time for syntax highlighting.  Cache the result?
7   setpcmark() shifts the jumplist, this takes quite a bit of time when
    jumping around.  Instead use an index for the start?
8   When displaying a space with only foreground highlighting, it's the same
    as a space without attributes.  Avoid displaying spaces for the "~" lines
    when starting up in a color terminal.
8   Avoid alloc() for scratch buffer use, esp. in syntax.c.  It's very slow on
    Win16.
8   Profiling shows that in_id_list() is used very often for C code.  Can this
    function be improved?
8   For an existing file, the page size of the swap file is always the
    default, instead of using the block size of the device, because the swap
    file is created only after setting the block size in mf_open().  How can
    this be improved?
8   Set default for 'ttyscroll' to half a screen height?  Should speed up
    MS-DOS version. (Negri)
7   C syntax highlighting gets a lot slower after ":set foldmethod=syntax".
    (Charles Campbell)  Inserting a "{" is very slow. (dman)
7   HTML syntax highlighting is slow for long lines.  Try displaying
    http://www.theregister.co.uk/content/4/22908.html. (Andre Pang)
7   Check how performance of loading the wordlist can be improved (adding a
    lot of abbreviations).
7   Compile Ex commands to byte codes.  Store byte codes in a vim script file
    at the end, after "compiled:.  Make it look like a single comment line
    for old Vim versions.  Insert first line "Vim script compiled <timestamp>.
    Only used compiled code when timestamp matches the file stat.
```

```
    Add command to compile a vim script and add it to the file in-place.
    Split Ex command executing into a parsing and executing phase.
    Use compiled code for functions, while loops, etc.
8   When defining autocommands (e.g., from $VIMRUNTIME/filetype.vim), need to
    compare each pattern with all existing patterns.  Use a hash code to avoid
    using strcmp() too often?
7   Include turbo_loader patches, speeding up reading a file?
    Speed up reading a file by reading it into a fixed-size buffer, creating
    the list of indexes in another buffer, and then copying the result into a
    memfile block with two copies.  Then read the next block into another
    fixed-size buffer, create the second list of indexes and copy text from
    the two blocks to the memfile block.
7   do_cmdline(): Avoid that the command line is copied to allocated memory
    and freed again later all the time.  For while loops, and for when called
    with an argument that can be messed with.
    Generic solution: Make a struct that contains a pointer and a flag that
    indicates if the pointer should be freed when replaced.
7   Check that the file size is not more than "sizeof(long)".
-   Further improve finding mappings in maphash[] in vgetorpeek()
8   Syntax highlighting is slow when deleting lines.  Try in
    $VIMRUNTIME/filetype.vim.
-   "out of memory" after deleting (1,$d) and changing (:%s/^/> /) a lot of
    lines (27000) a few times.  Memory fragmentation?
-   Have a look at how pdksh does memory allocation (alloc.c). (Dalecki)
-   Do profiling on:
    - :g/pat/normal cmd
    - 1000ii<Esc>
    - deleting 10Mbyte worth of lines (netscape binary)
    - "[i" and "[d" (Yegappan Lakshmanan)
    - ":g/^/m0" on a 450Kbyte file.  And the "u".
    - highlighting "~/vim/test/longline.tex", "~/vim/test/scwoop.tcl" and
      "~/vim/test/lockup.pl".
    - loading a syntax file to highlight all words not from a dictionary.
    - editing a Vim script with syntax highlighting on (loading vim.vim).
7   Screen updating can be further improved by only redrawing lines that were
    changed (and lines after them, when syntax highlighting was used, and it
    changed).
    - On each change, remember start and end of the change.
    - When inserting/deleting lines, remember begin, end, and line count.
-   Use macros/duarte/capicua for profiling.  Nvi 1.71 is the fastest!
-   When using a file with one long line (1Mbyte), then do "$hhhh", is still
    very slow.  Avoid calling getvcol() for each "h"?
-   Executing a register, e.g. "10000@@" is slow, because ins_typebuf has to
    move the previous commands forward each time.  Pass count from
    normal_cmd() down to do_execreg().
-   Repeating insert "1000i-<Esc>" displays --INSERT-- all the time, because of
    the <Esc> at the end.  Make this work faster (disable redrawing).
-   Avoid calls to plines() for cursor line, use w_cline_height.
-   After ":set nowrap" remove superfluous redraw with wrong hor. offset if
    cursor is right of the screen.
8   Make CTRL-C on Unix generate a signal, avoid using select() to check for a
    CTRL-C (it's slow).


Code size:
8   GUI: When NO_CONSOLE is defined, more code can be excluded.
-   Put getline() and cookie in a struct, so only one argument has to be
    passed to do_cmdline() and other functions.
8   Make a GUI-only version for Unix?
8   In buf_write _() isn't needed when setting errmsg, do it once when using
    it.
7   When compiling with a GUI-only version, the code for cterm colors can be
```

```
    left out.
8   When compiled with a GUI-only version, the termcap entries for terminals
    can be removed.
8   Can the check for libelf in configure.ac be removed?


Messages:
8   When using ":q" in a changed file, the error says to "add !".  Add the
    command so that beginners understand it: "use :q!".
8   For 'verbose' level 12 prints commands from source'ed files.  How to skip
    lines that aren't executed?  Perhaps move the echoing to do_cmdline()?
8   Use 'report' for ":bdel"?  (Krishna)  To avoid these messages when using a
    script.
-   Delete message after new command has been entered and have waited for key.
    Perhaps after ten seconds?
-   Make message history available in "msg" variables: msg1, msg2, .. msg9.
8   When reading from stdin allow suppressing the "reading from stdin"
    message.
9   Check handling of overwriting of messages and delays:
    Very wrong: errors while redrawing cause endless loop.
    When switching to another file and screen scrolls because of the long
    message and return must be typed, don't scroll the screen back before
    redrawing.
8   When address range is wrong you only get "Invalid range".  Be a bit more
    specific: Negative, beyond last line, reverse range?  Include the text.
8   Make it possible to ignore errors for a moment ('errorignore'?).  Another
    option to switch off giving error messages ('errorquiet'?).  Also an option
    not to give any messages ('quiet')?  Or ":quiet on", ":quiet off".
    Careful: For a severe error (out of memory), and when the user starts
    typing, error messages must be switched back on.
    Also a flag to ignore error messages for shell commands (for mappings).
-   Option to set time for emsg() sleep.  Interrupt sleep when key is typed?
    Sleep before second message?
8   In Ex silent mode or when reading commands from a file, what exactly is
    not printed and what is?  Check ":print", ":set all", ":args", ":vers",
    etc.  At least there should be no prompt. (Smulders)  And don't clear the
    screen when reading commands from stdin. (Kendall)
    --> Make a difference between informative messages, prompts, etc. and
        error messages, printing text, etc.
8   Window should be redrawn when resizing at the hit-enter prompt.
    Also at the ":tselect" prompt.  Find a generic solution for redrawing when
    a prompt is present (with a callback function?).


Screen updating:
7   Add a string to the 'display' option to make CTRL-E and CTRL-Y scroll one
    screen line, also if this means the first line doesn't start with the
    first character (like what happens with a single line that doesn't fit).
-   screen_line():
    - insert/delete character stuff.
    - improve delete rest of line (spaces at end of line).
-   When moving or resizing window, try to avoid a complete redraw (esp. when
    dragging the status line with the mouse).
-   When 'lazyredraw' set, don't echo :ex commands?  Need a flag to redraw when
    waiting for a character.
8   Add a ":refresh [winnr]" command, to force updating a window.  Useful from
    an event handler where ":normal" can't be used.  Also useful when
    'lazyredraw' is set in a mapping.

Scrolling:
8   Add "zy" command: scroll horizontally to put the cursor in the middle.
6   Add option to set the overlap for CTRL-F and CTRL-B. (Garhi)
```

```
-   extend 'scrollbind' option: 'scrollopt' words "search", "relative", etc..
    Also 'e'xecute some commands (search, vertical movements) in all bound
    windows.
7   Add 'scrollbind' feature to make the offset of one window with the next
    one equal to the window height.  When editing one file in both windows it
    looks like each window displays a page of the buffer.
-   Allow scrolling by dragging with the mouse (grab a character and move it
    up/down).  Like the "hand" in Acrobat reader.  Use Alt-LeftMouse for this?
    (Goldfarb)
-   Add command to execute some commands (search, vertical movements) in all
    bound windows.
-   Add 'search' option to 'scrollopt' to allow 'scrollbind' windows to
    be bound by regexp searches
-   Add "z>" and "z<": scroll sideways one screenful. (Campbell)
-   Add option to set the number of lines when not to scroll, instead of the
    fixed number used now (for terminals that scroll slow with a large number
    of lines but not with a single line).


Autoconf:
8   Should use acconfig.h to define prototypes that are used by autoheader.
8   Some compilers don't give an error for "-OPT:Olimit" but a warning. (Webb)
    Add a check for the warning, so that "Olimit" can be added automatically?
-   Autoconf: Use @datadir@ for the system independent files.  Make sure the
    system dependent and system independent files are separated. (Leitner).
-   Add autoconf check for waitpid()/wait4().
-   Remove fcntl() from autoconf, all systems have it?
-   Set default for 'dictionary', add search for dictionary to autoconf.


Perl interface:
8   Rename typemap file to something else?
7   Make buffers accessed as Perl arrays. (Clark)
7   Make it possible to compile with non-ANSI C?
6   Tcl/Tk has the "load" command: load a shared library (.so or .dll).


Shared libraries:
6   Add support for loading shared libraries, and calling functions in it.
        :libload internal-name libname
        :libunload internal-name
        :liblist
        :libcall internal-name function(arg1, arg2, ...)
        :libcall function(arg1, arg2, ...)
    libcall() can have only one integer or String argument at the moment.
6   Have a look on how Perl handles loading dynamic libraries.


Tags:
9   With ":set tags=./tags,../tags" and a tag appears in both tags files it is
    added twice.  Requires figuring out the actual file name for each found
    match.  Remove tag_fname from the match and combine it with the fname in
    the match (without expanding or other things that take time).  When
    'tagrelative' is off tag_fname isn't needed at all.
8   For 'tags' wildcard in the file name is not supported, only in the path.
    This is due to it using |file-searching|. Suboptimal solution would be to
    make the filename or the whole option use |wildcards| globing, better
    would be to merge the 2 kinds of globing. originally (Erik Falor, 2008
    April 18), updated (Ian Kelling, 2008 July 4)
7   Can CTRL-] (jump to tag) include a following "." and "->" to restrict the
    number of possible matches? Check tags file for an item that has members.
    (Flemming Madsen)
```

```
8   Scope arguments for ":tag", e.g.: ":tag class:cPage open", like Elvis.
8   When output of ":tselect" is long, getting the more-prompt, should be able
    to type the tag number directly.
7   Add the possibility to use the "-t {tag}" argument multiple times.  Open a
    window for each tag.
7   Make output of ":tselect" a bit nicer.  Use highlighting?
7   Highlight the "tag 1 of >2" message.  New highlight group, or same as "hit
    bottom" search message.
7   When using ":tag" at the top of the tag stack, should add another entry,
    so CTRL-T can bring you back to where you are now AND to where you were
    before the previous ":tag" command. (Webb)
-   When doing "[^I" or "[^D" add position to tag stack.
-   Add command to put current position to tag stack: ":tpush".
-   Add functions to save and restore the tag stack?  Or a command to switch
    to another tag stack?  So that you can do something else and come back to
    what you were working on.
7   When using CTRL-] on someClass::someMethod, separate class from method and
    use ":ta class:someClass someMethod".
    Include C++ tags changes (Bertin).  Change "class::func" tag into "func"
    with "class=class"?  Docs in oldmail/bertin/in.xxx.
7   Add ":tagargs", to set values for fields:
        :tagargs class:someclass file:version.c
        :tagargs clear
    These are then the default values (changes the order of priority in tag
    matching).
7   Support for "gtags" and "global"?  With ":rtag" command?
    There is an example for how to do this in Nvi.
    Or do it like Elvis: 'tagprg' and 'tagprgonce' options. (Yamaguchi)
    The Elvis method is far more flexible, do it that way.
7   Support "col:99" extra field, to position the cursor in that column.  With
    a flag in 'cpoptions' to switch it off again.
7   Better support for jumping to where a function or variable is used.  Use
    the id-utils, with a connection to "gid" (Emacs can do it too).  Add
    ":idselect", which uses an "ID" database (made by "mkid") like "tselect".


Win32 GUI:
8   Make debug mode work while starting up (vim -D).  Open console window for
    the message and input?
7   GvimExt: when there are several existing Vims, move the list to a submenu.
    (Mike McCollister)
8   When using "Edit with Vim" for one file it changes directory, when several
    files are selected and using "Edit with single Vim" the directory isn't
    changed.  At least change directory when the path is the same for all
    files.  Perhaps just use the path of the first file or use the longest
    common part of the path.
8   Add font argument to set the lfCharSet. (Bobcik)
8   Somehow automatically detect the system language and set $LANG, so that
    gettext and menus work.
8   Could keep console open to run multiple commands, to avoid the need to hit
    return in every console.
    Also: Look at how Emacs does run external commands:
        http://www.cs.washington.edu/homes/voelker/ntemacs.html.
8   Need a separate PopUp menu for modeless selection.  Need two new commands:
    Copy selection to clipboard, Paste selection (as typed text).
8   Support copy/paste for other file formats.  At least HTML, perhaps RTF.
    Add "copy special" and "paste special" commands?
7   Use different default colors, to match the current Windows color scheme.
    Sys_WindowText, Sys_Window, etc. (Lionel Schaffhauser)
7   Use <C-Tab> to cycle through open windows (e.g., the find dialog).
7   <Esc> should close a dialog.
7   Keep the console for external commands open.  Don't wait for a key to be
```

```
    hit.  Re-open it when the user has closed it anyway.  Or use a prepended
    command: ":nowait {cmd}", or ":quiet", which executes {cmd} without any
    prompts.
7   Should be able to set an option so that when you double click a file that
    is associated with Vim, you can either get a new instance of Vim, or have
    the file added into an already running Vim.
7   The "-P" argument only works for the current codepage.  Use wide
    functions to find the window title.


GUI:
8   Make inputdialog() work for Photon, Amiga.
-   <C--> cannot be mapped.  Should be possible to recognize this as a
    normal "-" with the Ctrl modifier.
7   Implement ":popup" for other systems than Windows.
8   Implement ":tearoff" for other systems than Win32 GUI.
6   Implement ":untearoff": hide a torn-off menu.
8   When using the scrollbar to scroll, don't move the cursor position.  When
    moving the cursor: scroll to the cursor position.
9   Make <S-Insert> paste from the clipboard by default. (Kunze)
7   Menu local to a buffer, like mappings.  Or local to a filetype?
8   In Buffers menu, add a choice whether selecting a buffer opens it in the
    current window, splits the window or uses ":hide".
8   Dragging the mouse pointer outside of a Vim Window should make the text
    scroll.  Return a value from gui_send_mouse_event() to the machine
    specific code to indicate the time in which the event should be repeated.
8   Make it possible to ignore a mouse click when it's used to give Vim (gvim)
    window focus.  Also when a mouse click is used to bring a window to front.
8   Make the split into system independent code and system specific code more
    explicit.  There are too many #ifdefs in gui.c.
    If possible, separate the Vim code completely from the GUI code, to allow
    running them in separate processes.
7   X11: Support cursorColor resource and "-cr" argument.
8   X11 (and others): CTRL-; is not different from ';'.  Set the modifier mask
    to include CTRL for keys where CTRL produces the same ASCII code.
7   Add some code to handle proportional fonts on more systems?  Need to draw
    each character separately (like xterm).  Also for when a double-width font
    is not exactly double-width. (Maeda)
8   Should take font from xterm where gvim was started (if no other default).
8   Selecting font names in X11 is difficult, make a script or something to
    select one.
8   Visual highlighting should keep the same font (bold, italic, etc.).
8   Add flag to 'guioptions' to not put anything in the clipboard at all?
8   Should support a way to use keys that we don't recognize yet.  Add a
    command that adds entries to special_keys somehow.  How do we make this
    portable (X11, Win32, ..)?
7   Add a flag to 'guioptions' that tells not to remove inactive menu items.
    For systems where greying-out or removing menu items is very slow.  The
    menu items would remain visibly normally, but not do anything.
7   Add ":minimize" and ":maximize", which iconize the window and back.
    Useful when using gvim to run a script (e.g. 2html.vim).
7   X11: Is it possible to free allocated colors, so that other programs can
    use them again?  Otherwise, allow disabling allocating the default colors.
    Or allocate an own colormap (check UAE).  With an option to use it.  For
    the commandline, "-install" is mostly used for X11 programs.
7   Should support multi-column menus.
-   Should add option for where to put the "Help" menu: like Motif at the far
    right, or with the other menus (but still at the right).
-   Add menu item to "Keep Insert mode".
8   ":mkgvimrc" command, that includes menus.
6   Big change: Move GUI to separate program "vimgui", to make startup of vim a
    lot faster, but still be able to do "vim -g" or ":gui".
```

```
7   More explicit mouse button binding instead of 'mousemodel'?
7   Add option to set the position of the window on the screen.  'windowpos',
    which has a value of "123,456": <x>,<y>.
    Or add a command, like ":winsize"?
7   Add toolbar for more GUIs.
8   Make it possible to use "amenu icon=BuiltIn##", so that the toolbar item
    name can be chosen free.
7   Make it possible to put the toolbar on top, left, right and/or bottom of
    the window?  Allows for softkey-like use.
6   Separate the part of Vim that does the editing from the part that runs the
    GUI.  Communicate through a pseudo-tty.  Vim starts up, creates a
    pty that is connected to the terminal.  When the GUI starts, the pty is
    reconnected to the GUI process.  When the GUI stops, it is connected to
    the terminal again.  Also use the pty for external processes, it looks
    like a vt100 terminal to them.  Vim uses extra commands to communicate GUI
    things.
7   Motif: For a confirm() dialog <Enter> should be ignored when no default
    button selected, <Esc> should close the dialog.
7   When using a pseudo-tty Vim should behave like some terminal (vt52 looks
    simple enough).  Terminal codes to/from shell should be translated.
-   Would it be useful to be able to quit the GUI and go back to the terminal
    where it was started from?
7   Support "-visual <type>" command line argument.


Autocommands:
9   Rework the code from FEAT_OSFILETYPE for autocmd-osfiletypes to use
    'filetype'.  Only for when the current buffer is known.
-   Put autocommand event names in a hashtable for faster lookup?
8   When the SwapExists event is triggered, provide information about the
    swap file, e.g., whether the process is running, file was modified, etc.
    Must be possible to check the situation that it's probably OK to delete
    the swap file. (Marc Merlin)
8   When all the patterns for an event are "*" there is no need to expand
    buffer names to a full path.  This can be slow for NFS.
7   For autocommand events that trigger multiple times per buffer (e.g.,
    CursorHold), go through the list once and cache the result for a specific
    buffer.  Invalidate the cache when adding/deleting autocommands or
    changing the buffer name.
7   Add TagJump event: do something after jumping to a tag.
8   Add "TagJumpFile" autocommand: When jumping to another file for a tag.
    Can be used to open "main.c.gz" when "main.c" isn't found.
8   Use another option than 'updatetime' for the CursorHold event.  The two
    things are unrelated for the user (but the implementation is more
    difficult).
7   Add autocommand event for when a buffer cannot be abandoned.  So that the
    user can define the action taking (autowrite, dialog, fail) based on the
    kind of file. (Yakov Lerner)  Or is BufLeave sufficient?
8   Autocommand for when modified files have been found, when getting input
    focus again (e.g., FileChangedFocus).
    Check when: getting focus, jumping to another buffer, ...
8   Autocommands should not change registers.  And marks?  And the jumplist?
    And anything else?  Add a command to save and restore these things.
8   Add autocommands, user functions and user commands to ":mkvimrc".
6   Add KeymapChanged event, so that the effects of a different keymap can be
    handled (e.g., other font) (Ron Aaron)
7   When trying to open a directory, trigger an OpenDirectory event.
7   Add file type in front of file pattern: <d> for directory, <l> for link,
    <x> for executable, etc.  With commas to separate alternatives.  The
    autocommand is only executed when both the file type AND the file pattern
    match. (Leonard)
5   Add option that specifies extensions which are to be discarded from the
```

```
    file name.  E.g. 'ausuffix', with ".gz,.orig".  Such that file.c.gz will
    trigger the "*.c" autocommands.  (Belabas)
7   Add something to break the autocommands for the current event, and for
    what follows.  Useful for a "BufWritePre" that wants to avoid writing the
    file.
8   When editing "tt.gz", which is in DOS format, 'fileformat' stays at
    "unix", thus writing the file changes it.  Somehow detect that the read
    command used dos fileformat.  Same for 'fileencoding'.
-   Add events to autocommands:
    Error           - When an error happens
    NormalEnter     - Entering Normal mode
    ReplaceEnter    - Entering Replace mode
    CmdEnter        - Entering Cmdline mode (with type of cmdline to allow
                      different mapping)
    VisualEnter     - Entering Visual mode
    *Leave          - Leaving a mode (in pair with the above *Enter)
    VimLeaveCheck    - Before Vim decides to exit, so that it can be cancelled
                      when exiting isn't a good idea.
    CursorHoldC     - CursorHold while command-line editing
    WinMoved        - when windows have been moved around, e.g, ":wincmd J"
    SearchPost      - After doing a search command (e.g. to do "M")
    PreDirChanged/PostDirChanged
                    - Before/after ":cd" has been used (for changing the
                      window title)
    ShutDown        - when the system is about to shut down
    InsertCharPost  - user typed a character in Insert mode, after inserting
                      the char.
    BufModified     - When a buffer becomes modified, or unmodified (for
                      putting a [+] in the window title or checking out the
                      file from CVS).
    BufFirstChange  - When making a change, when 'modified' is set.  Can be
                      used to do a :preserve for remote files.
    BufChange       - after a change was made.  Set some variables to indicate
                      the position and number of inserted/deleted lines, so
                      that marks can be updated.  HierAssist has patch to add
                      BufChangePre, BufChangePost and RevertBuf. (Shah)
    ViewChanged     - triggered when the text scrolls and when the window size
                      changes.
    WinResized      - After a window has been resized
    WinClose        - Just before closing a window
-   Write the file now and then ('autosave'):
                                *'autosave'* *'as'* *'noautosave'* *'noas'*
    'autosave' 'as' number  (default 0)
            Automatically write the current buffer to file N seconds after the
            last change has been made and when |'modified'| is still set.
            Default: 0 = do not autosave the buffer.
    Alternative: have 'autosave' use 'updatetime' and 'updatecount' but make
    them save the file itself besides the swapfile.


Omni completion:
-   Add a flag to 'complete' to be able to do omni completion with CTRL-N (and
    mix it with other kinds of completion).
-   Ideas from the Vim 7 BOF at SANE:
        - For interpreted languages, use the interpreter to obtain information.
          Should work for Java (Eclipse does this), Python, Tcl, etc.
          Richard Emberson mentioned working on an interface to Java.
        - Check Readline for its completion interface.
-   Ideas from others:
        http://www.wholetomato.com/
        http://www.vim.org/scripts/script.php?script_id=747
            http://sourceforge.net/projects/insenvim
```

```
            or http://insenvim.sourceforge.net
          Java, XML, HTML, C++, JSP, SQL, C#
          MS-Windows only, lots of dependencies (e.g. Perl, Internet
            explorer), uses .dll shared libraries.
          For C++ uses $INCLUDE environment var.
          Uses Perl for C++.
          Uses ctags to find the info:
            ctags -f $allTagsFile --fields=+aiKmnsSz --language-force=C++ --C++-
kinds=+cefgmnpsut-dlux -u $files
      www.vim.org script 1213 (Java Development Environment) (Fuchuan Wang)
      IComplete: http://www.vim.org/scripts/script.php?script_id=1265
          and http://stud4.tuwien.ac.at/~e0125672/icomplete/
      http://cedet.sourceforge.net/intellisense.shtml (for Emacs)
      Ivan Villanueva has something for Java.
      Emacs: http://www.xref-tech.com/xrefactory/more_c_completion.html
      Completion in .NET framework SharpDevelop: http://www.icsharpcode.net
-   Pre-expand abbreviations, show which abbrevs would match?


Insert mode completion/expansion:
-   GUI implementation of the popup menu.
7   When searching in other files the name flash by, too fast to read.  Only
    display a name every second or so, like with ":vimgrep".
7   When expanding file names with an environment variable, add the match with
    the unexpanded var.  So $HOME/tm expands to "/home/guy/tmp" and
    "$HOME/tmp"
8   When there is no word before the cursor but something like "sys." complete
    with "sys.".  Works well for C and similar languages.
9   ^X^L completion doesn't repeat correctly.  It uses the first match with
    the last added line, instead of continuing where the last match ended.
    (Webb)
8   Add option to set different behavior for Insert mode completion:
    - ignore/match case
    - different characters than 'iskeyword'
8   Add option 'isexpand', containing characters when doing expansion (so that
    "." and "\" can be included, without changing 'iskeyword'). (Goldfarb)
    Also: 'istagword': characters used for CTRL-].
    When 'isexpand' or 'istagword' are empty, use 'iskeyword'.
    Alternative: Use a pattern so that start and end of a keyword can be
    defined, only allow dash in the middle, etc.
8   Add a command to undo the completion, go back to the original text.
7   Completion of an abbreviation: Can leave letters out, like what Instant
    text does: www.textware.com
8   Use the class information in the tags file to do context-sensitive
    completion.  After "foo." complete all member functions/variables of
    "foo".  Need to search backwards for the class definition of foo.
    Should work for C++ and Java.
    Even more context would be nice: "import java.^N" -> "io", "lang", etc.
7   When expanding $HOME/dir with ^X^F keep the $HOME (with an option?).
7   Add CTRL-X command in Insert mode like CTRL-X CTRL-N, that completes WORDS
    instead of words.
8   Add CTRL-X CTRL-R: complete words from register contents.
8   Add completion of previously inserted texts (like what CTRL-A does).
    Requires remembering a number of insertions.
8   Add 'f' flag to 'complete': Expand file names.
    Also apply 'complete' to whole line completion.
-   Add a flag to 'complete' to only scan local header files, not system
    header files. (Andri Moell)
-   Make it possible to search include files in several places.  Use the
    'path' option?  Can this be done with the dictionary completion (use
    wildcards in the file name)?
-   Make CTRL-X CTRL-K do a binary search in the dictionary (if it's sorted).
```

```
-   Speed up CTRL-X CTRL-K dictionary searching (don't use a regexp?).
-   Set a mark at the position where the match was found (file mark, could
    be in another file).
-   Add CTRL-A command in CTRL-X mode: show all matches.
-   Make CTRL-X CTRL-L use the 'complete' option?
-   Add command in CTRL-X mode to add following words to the completed string
    (e.g. to complete "Pointer->element" with CTRL-X CTRL-P CTRL-W CTRL-W)
-   CTRL-X CTRL-F: Use 'path' to find completions.
-   CTRL-X CTRL-F: Option to use forward slashes on MS-Windows?
-   CTRL-X CTRL-F: Don't replace "$VIM" with the actual value. (Kelly)
-   Allow listing all matches in some way (and picking one from the list).


Command line editing:
7   Add commands (keys) to delete from the cursor to the end of the command
    line.
8   Custom completion of user commands can't use the standard completion
    functions.  Add a hook to invoke a user function that returns the type of
    completion to be done: "file", "tag", "custom", etc.
-   Add flags to 'whichwrap' for command line editing (cursor right at end of
    lines wraps to start of line).
-   Make editing the command line work like Insert mode in a single-line view
    on a buffer that contains the command line history.  But this has many
    disadvantages, only implement it when these can be solved.  Elvis has run
    into these, see remarks from Steve (~/Mail/oldmail/kirkendall/in.00012).
    - Going back in history and editing a line there would change the history.
      Would still need to keep a copy of the history elsewhere.  Like the
      cmdwin does now already.
    - Use CTRL-O to execute one Normal mode command.  How to switch to normal
      mode for more commands?  <Esc> should cancel the command line.  CTRL-T?
    - To allow "/" and "= need to recursively call getcmdline(), overwrite the
      cmdline.  But then we are editing a command-line again.  How to avoid
      that the user gets confused by the stack of command lines?
    - Use edit() for normal cmdline editing?  Would have to integrate
      getcmdline() into edit().  Need to solve conflicts between Insert mode
      and Command-line mode commands.  Make it work like Korn shell and tcsh.
      Problems:
        - Insert: completion with 'wildchar'
        - Insert: use cmdline abbreviations
        - Insert: CTRL-D deletes indent instead of listing matches
        - Normal: no CTRL-W commands
        - Normal: no ":" commands?
        - Normal: allow Visual mode only within one line.
    - where to show insert/normal mode message?  Change highlighting of
      character in first column?
    - Implementation ideas:
      - Set "curwin" and "curbuf" to the command line window and buffer.
      - curwin->w_topline is always equal to curwin->w_cursor.lnum.
      - never set 'number', no folding, etc.  No status line.
      - sync undo after entering a command line?
      - use NV_NOCL flag for commands that are not allowed in Command-line
        Mode.


Command line completion:
8   Change expand_interactively into a flag that is passed as an argument.
8   With command line completion after '%' and '#', expand current/alternate
    file name, so it can be edited.  Also with modifiers, such as "%:h".
8   When completing command names, either sort them on the long name, or list
    them with the optional part inside [].
8   Add an option to ignore case when doing interactive completion.  So that
    ":e file<Tab>" also lists "Filelist" (sorted after matching case matches).
```

```
7   Completion of ":map x ": fill in the current mapping, so that it can be
    edited. (Sven Guckes)
-   For 'wildmenu': Simplify "../bar" when possible.
-   When using <Up> in wildmenu mode for a submenu, should go back to the
    current menu, not the first one.  E.g., ":emenu File.Save<Up>".
8   When using backtick expansion, the external command may write a greeting
    message.  Add an option or commands to remove lines that match a regexp?
7   When listing matches of files, display the common path separately from the
    file names, if this makes the listing shorter. (Webb)
-   Add command line completion for ":ilist" and friends, show matching
    identifiers (Webb).
8   Add command line completion for "old value" of a command.  ":args <key>"
    would result in the current list of arguments, which you can then edit.
7   Add command line completion with CTRL-X, just like Insert mode completion.
    Useful for ":s/word/xx/".
-   Add command to go back to the text as it was before completion started.
    Also to be used for <Up> in the command line.
-   Add 'wildlongest' option: Key to use to find longest common match for
    command line completion (default CTRL-L), like 'wildchar'. (Cregut)
    Also: when there are several matches, show them line a CTRL-D.


Command line history:
-   Add "KeyWasTyped" flag: It's reset before each command and set when a
    character from the keyboard is consumed. Value is used to decide to put a
    command line in history or not. Put line in history if it didn't
    completely resulted from one mapping.
-   When using ":browse", also put the resulting edit command in the history,
    so that it can be repeated. (Demirel)


Insert mode:
9   When 'autoindent' is set, hitting <CR> twice, while there is text after
    the cursor, doesn't delete the autoindent in the resulting blank line.
    (Rich Wales) This is Vi compatible, but it looks like a bug.
8   When using CTRL-O in Insert mode, then executing an insert command
    "a" or "i", should we return to Insert mode after <Esc>? (Eggink)
    Perhaps it can be allowed a single time, to be able to do
    "<C-O>10axyz<Esc>".  Nesting this further is confusing.
    ":map <F2> 5aabc<Esc>" works only once from Insert mode.
8   When using CTRL-G CTRL-O do like CTRL-\ CTRL-O, but when returning with
    the cursor in the same position and the text didn't change continue the
    same change, so that "." repeats the whole insert.
7   Use CTRL-G <count> to repeat what follows.  Useful for inserting a
    character multiple times or repeating CTRL-Y.
-   Make 'revins' work in Replace mode.
7   Use 'matchpairs' for 'showmatch': When inserting a character check if it
    appears in the rhs of 'matchpairs'.
-   In Insert mode (and command line editing?): Allow undo of the last typed
    character.  This is useful for CTRL-U, CTRL-W, delete and backspace, and
    also for characters that wrap to the next line.
    Also: be able to undo CTRL-R (insert register).
    Possibly use 'backspace'="whole" for a mode where at least a <CR> that
    inserts autoindent is undone by a single <BS>.
-   Use CTRL-G in Insert mode for an extra range of commands, like "g" in
    Normal mode.
-   Make 'paste' work without resetting other options, but override their
    value.  Avoids problems when changing files and modelines or autocommands
    are used.
-   When typing CTRL-V and a digit higher than 2, only expect two digits.
-   Insert binary numbers with CTRL-V b.
-   Make it possible to undo <BS>, <C-W> and <C-U>.  Bash uses CTRL-Y.
```

```
'cindent', 'smartindent':
9   Wrapping a variable initialization should have extra indent:
        char * veryLongName =
                 "very long string"
    Also check if "cino=+10" is used correctly.
8   Lisp indenting: "\\" confuses the indenter. (Dorai Sitaram, 2006 May 17)
8   Why are continuation lines outside of a {} block not indented?  E.g.:
        long_type foo =
        value;
8   Java: Inside an anonymous class, after an "else" or "try" the indent is
    too small. (Vincent Bergbauer)
    Problem of using {} inside (), 'cindent' doesn't work then.
8   In C++ it's possible to have {} inside (): (Kirshna)
                func(
                        new String[] {
                            "asdf",
                            "asdf"
                        }
                    );
8   In C++ a function isn't recognized inside a namespace:
    (Chow Loong Jin)
        namespace {
            int
                 func(int arg) {
                 }
        }
6   Add 'cino' flag for this function argument layout: (Spencer Collyer)
            func( arg1
                , arg2
                , arg3
                );
7   Add separate "(0" option into inside/outside a function (Zellner):
        func(
           int x)       // indent like "(4"
        {
           if (a
               && b)    // indent like "(0"
9   Using "{" in a comment: (Helmut Stiegler)
    if (a)
    {
            if (b)
            {
                    // {
            }
            } <-- this is indented incorrect
    Problem is that find_start_brace() checks for the matching brace to be in
    a comment, but not braces in between.  Requires adding a comment check to
    findmatchlimit().
-   Make smartindenting configurable.  Add 'sioptions', e.g. '#' setting the
    indent to 0 should be switched on/off.
7   Support ANSI style function header, with each argument on its own line.
-   "[p" and "]p" should use 'cindent' code if it's on (only for the first
    line).
-   Add option to 'cindent' to set indent for comments outside of {}?
-   Make a command to line up a comment after a code line with a previous
    comment after a code line.  Can 'cindent' do this automatically?
-   When 'cindent'ing a '}', showmatch is done before fixing the indent.  It
    looks better when the indent is fixed before the showmatch. (Webb)
-   Add option to make indenting work in comments too (for commented-out
    code), unless the line starts with "*".
```

```
-   Don't use 'cindent' when doing formatting with "gq"?
-   When formatting a comment after some text, insert the '*' for the new line
    (indent is correct if 'cindent' is set, but '*' doesn't get inserted).
8   When 'comments' has both "s1:/*,mb:*,ex:*/" and "s1:(*,mb:*,ex:*)", the
    'x' flag always uses the first match.  Need to continue looking for more
    matches of "*" and remember all characters that could end the comment.
-   For smartindent: When typing 'else' line it up with matching 'if'.
-   'smartindent': allow patterns in 'cinwords', for e.g. TeX files, where
    lines start with "\item".
-   Support this style of comments (with an option): (Brown)
        /* here is a comment that
        is just autoindented, and
        nothing else */
-   Add words to 'cinwords' to reduce the indent, e.g., "end" or "fi".
7   Use Tabs for the indent of starting lines, pad with spaces for
    continuation lines.  Allows changing 'tabstop' without messing up the
    indents.
    Patch by Lech Lorens, 2010 Mar.  Update by James McCoy, 2014 Mar 15.


Java:
8   Can have {} constructs inside parens.  Include changes from Steve
    Odendahl?
8   Recognize "import java.util.Vector" and use $CLASSPATH to find files for
    "[i" commands and friends.
-   For files found with 'include': handle "*" in included name, for Java.
    (Jason)
-   How to make a "package java.util" cause all classes in the package to be
    searched?  Also for "import java.util.*". (Mark Brophy)


'comments':
8   When formatting C comments that are after code, the "*" isn't repeated
    like it's done when there is no code.  And there is no automatic wrapping.
    Recognize comments that come after code.  Should insert the comment leader
    when it's "#" or "//".
    Other way around: when a C command starts with "* 4" the "*" is repeated
    while it should not.  Use syntax HL comment recognition?
7   When using "comments=fg:--", Vim inserts three spaces for a new line.
    When hitting a TAB, these spaces could be removed.
7   The 'n'esting flag doesn't do the indenting of the last (rightmost) item.
6   Make strings in 'comments' option a RE, to be able to match more
    complicated things. (Phillipps)  Use a special flag to indicate that a
    regexp is used.
8   Make the 'comments' option with "/* * */" lines only repeat the "*" line
    when there is a "/*" before it?  Or include this in 'cindent'?


Virtual edit:
8   Make the horizontal scrollbar work to move the text further left.
7   Allow specifying it separately for Tabs and beyond end-of-line?


Text objects:
8   Add text object for fold, so that it can be yanked when it's open.
8   Add test script for text object commands "aw", "iW", etc.
8   Add text object for part of a CamelHumpedWord and under_scored_word.
    (Scott Graham)  "ac" and "au"?
8   Add a text object for any kind of quoting, also with multi-byte
    characters.  Option to specify what quotes are recognized (default: all)
    use "aq" and "iq".  Use 'quotepairs' to define pairs of quotes, like
    'matchpairs'?
```

```
8   Add text object for any kind of parens, also multi-byte ones.
8   Add a way to make an ":omap" for a user-defined text object.  Requires
    changing the starting position in oap->start.
8   Add "gp" and "gP" commands: insert text and make sure there is a single
    space before it, unless at the start of the line, and after it, unless at
    the end of the line or before a ".".
7   Add objects with backwards extension?  Use "I" and "A".  Thus "2dAs"
    deletes the current and previous sentence. (Jens Paulus)
7   Add "g{" and "g}" to move to the first/last character of a paragraph
    (instead of the line just before/after a paragraph as with "{" and "}").
6   Ignore comment leaders for objects.  Make "das" work in reply-email.
5   Make it possible to use syntax group matches as a text object.  For
    example, define a "ccItem" group, then do "da<ccItem>" to delete one.
    Or, maybe just define "dai", delete-an-item, to delete the syntax item the
    cursor is on.


Select mode:
8   In blockwise mode, typed characters are inserted in front of the block,
    backspace deletes a column before the block. (Steve Hall)
7   Alt-leftmouse starts block mode selection in MS Word.
    See http://vim.wikia.com/wiki/Use_Alt-Mouse_to_select_blockwise.
7   Add Cmdline-select mode.  Like Select mode, but used on the command line.
    - Change gui_send_mouse_event() to pass on mouse events when 'mouse'
      contains 'C' or 'A'.
    - Catch mouse events in ex_getln.c.  Also shift-cursor, etc., like in
      normal_cmd().
    - remember start and end of selection in cmdline_info.
    - Typing text replaces the selection.


Visual mode:
8   Support using "." in Visual mode. Use the operator applied to the Visual
    selection, if possible.
-   When dragging the Visual selection with the mouse and 'scrolloff' is zero,
    behave like 'scrolloff' is one, so that the text scrolls when the pointer
    is in the top line.
-   Displaying size of Visual area: use 24-33 column display.
    When selecting multiple lines, up to about a screenful, also count the
    characters.
8   When using "I" or "A" in Visual block mode, short lines do not get the new
    text.  Make it possible to add the text to short lines too, with padding
    where needed.
7   With a Visual block selected, "2x" deletes a block of double the width,
    "3y" yanks a block of triple width, etc.
7   When selecting linewise, using "itext" should insert "text" at the start
    of each selected line.
8   What is "R" supposed to do in Visual mode?
8   Make Visual mode local to the buffer.  Allow changing to another buffer.
    When starting a new Visual selection, remove the Visual selection in any
    other buffer. (Ron Aaron)
8   Support dragging the Visual area to drop it somewhere else. (Ron Aaron,
    Ben Godfrey)
7   Support dragging the Visual area to drop it in another program, and
    receive dropped text from another program. (Ben Godfrey)
7   With blockwise Visual mode and "c", "C", "I", "A", etc., allow the use of
    a <CR>.  The entered lines are repeated over the Visual area.
7   Filtering a block should only apply to the block, not to the whole lines.
    When the number of lines is increased, add lines. When decreased, pad with
    spaces or delete? Use ":`<,`>" on the command line.
8   After filtering the Visual area, make "gv" select the filtered text?
    Currently "gv" only selects a single line, not useful.
```

```
7   Don't move the cursor when scrolling?  Needed when the selection should
    stay the same.  Scroll to the cursor at any movement command.  With an
    option!
7   In Visual block mode, need to be able to define a corner on a position
    that doesn't have text?  Also: when using the mouse, be able to select
    part of a TAB.  Even more: Add a mode where the cursor can be on a screen
    position where there is no text.  When typing, add spaces to fill the gap.
    Other solution: Always use curswant, so that you can move the cursor to
    the right column, and then use up/down movements to select the line,
    without changing the column.
6   ":left" and ":right" should work in Visual block mode.
7   CTRL-I and CTRL-O should work in Visual mode, but only jump to marks in the
    current buffer.
6   In non-Block mode, "I" should insert the same text in front of each line,
    before the first non-blank, "gI" in column 1.
6   In non-Block mode, "A" should append the same text after each line.
6   When in blockwise visual selection (CTRL-V), allow cursor to be placed
    right of the line.  Could also allow cursor to be placed anywhere on a TAB
    or other special character.
6   Add commands to move selected text, without deselecting.


More advanced repeating commands:
-   Add "." command for visual mode: redo last visual command (e.g. ":fmt").
-   Add command to repeat last movement.  Including count.
-   Add "." command after operator: repeat last command of same operator.  E.g.
    "c." will repeat last change, also when "x" used since then (Webb).
    "y." will repeat last yank.
    "c2." will repeat the last but one change?
    Also: keep history of Normal mode commands, add command to list the history
    and/or pick an older command.
-   History stack for . command?  Use "g." command.


Mappings and Abbreviations:
8   When "0" is mapped (it is a movement command) this mapping should not be
    used after typing another number, e.g. "20l". (Charles Campbell)
    Is this possible without disabling the mapping of the following command?
8   Should mapping <C-A> and <C-S-A> both work?
7   ":abbr b byte", append "b " to an existing word still expands to "byte".
    This is Vi compatible, but can we avoid it anyway?
8   To make a mapping work with a prepended "x to select a register, store the
    last _typed_ register name and access it with "&.
8   Add ":amap", like ":amenu".
7   Add a mapping that works always, for remapping the keyboard.
8   Add ":cab!", abbreviations that only apply to Command-line mode and not to
    entering search strings.
8   Add a flag to ":abbrev" to eat the character that triggers the
    abbreviation.  Thus "abb ab xxx" and typing "ab<Space>" inserts "xxx" and
    not the <Space>.
8   Give a warning when using CTRL-C in the lhs of a mapping.  It will never
    (?) work.
7   Add <0x8f> (hex), <033> (octal) and <123> (decimal) to <> notation?
7   When someone tries to unmap with a trailing space, and it fails, try
    unmapping without the trailing space.  Helps for ":unmap xx | unmap yy".
6   Context-sensitive abbreviations: Specify syntax group(s) in which the
    abbreviations are to be used.
-   Add mappings that take arguments.  Could work like the ":s" command.  For
    example, for a mouse escape sequence:
        :mapexp  <Esc>{\([0-9]*\),\([0-9]*\);   H\1j\2l
-   Add optional <Number> argument for mappings:
    :map <Number>q           ^W^W<Number>G
```

```
    :map <Number>q<Number>t  ^W^W<Number1-1>G<Number2>l
    :map q<Char>            :s/<Char>/\u\0/g
    Or implicit:
    :map q                      <Register>d<Number>$
-   Add command to repeat a whole mapping ("." only repeats the last change in
    a mapping).  Also: Repeat a whole insert command, including any mappings
    that it included.  Sort-of automatic recording?
-   Include an option (or flag to 'cpoptions') that makes errors in mappings
    not flush the rest of the mapping (like nvi does).
-   Use context sensitiveness of completion to switch abbreviations and
    mappings off for :unab and :unmap.
6   When using mappings in Insert mode, insert characters for incomplete
    mappings first, then remove them again when a mapping matches.  Avoids
    that characters that are the start of some mapping are not shown until you
    hit another character.
-   Add mappings for replace mode: ":rmap".  How do we then enter mappings for
    non-replace Insert mode?
-   Add separate mappings for Visual-character/block/line mode?
-   Add 'mapstop' command, to stop recursive mappings.
-   List mappings that have a raw escape sequence both with the name of the key
    for that escape sequence (if there is one) and the sequence itself.
-   List mappings: Once with special keys listed as <>, once with meta chars as
    <M-a>, once with the byte values (octal?).  Sort of "spell mapping" command?
-   When entering mappings: Add the possibility to enter meta keys like they
    are displayed, within <>: <M-a>, <~@> or <|a>.
-   Allow multiple arguments to :unmap.
-   Command to show keys that are not used and available for mapping
    ":freekeys".
-   Allow any character except white space in abbreviations lhs (Riehm).


Incsearch:
-   Add a limit to the number of lines that are searched for 'incsearch'?
-   When no match is found and the user types more, the screen is redrawn
    anyway.  Could skip that.  Esp. if the line wraps and the text is scrolled
    up every time.
-   Temporarily open folds to show where the search ends up.  Restore the
    folds when going to another line.
-   When incsearch used and hitting return, no need to search again in many
    cases, saves a lot of time in big files. (Slootman wants to work on this?)
    When not using special characters, can continue search from the last match
    (or not at all, when there was no match).  See oldmail/webb/in.872.


Searching:
9   Should have an option for :vimgrep to find lines without a match.
8   Add "g/" and "gb" to search for a pattern in the Visually selected text?
    "g?" is already used for rot13.
    The vis.vim script has a ":S" command that does something like this.
    Can use "g/" in Normal mode, uses the '< to '> area.
    Use "&/" for searching the text in the Visual area?
9   Add "v" offset: "/pat/v": search for pattern and start Visual mode on the
    matching text.
8   Add a modifier to interpret a space like "\_s\+" to make it much easier to
    search for a phrase.
8   Add a mechanism for recursiveness: "\@(([^()]*\@g[^()]*)\)".  \@g stands
    for "go recursive here" and \@( \) marks the recursive part.
    Perl does it this way:
            $paren = qr/ \(( [^()] | (??{ $paren }) )* \) /x;
    Here $paren is evaluated when it's encountered.  This is like a regexp
    inside a regexp.  In the above terms it would be:
            \@((\([^()]\|\@g\)*)\)
```

```
8   Show the progress every second.  Could use the code that checks for CTRL-C
    to find out how much time has passed.  Or use SIGALRM.  Where to show the
    number?
7   Support for approximate-regexps to find similar words (agrep
    http://www.tgries.de/agrep/ tre: http://laurikari.net/tre/index.html).
8   Add an item for a big character range, so that one can search for a
    chinese character: \z[234-1234]  or \z[XX-YY] or \z[0x23-0x234].
7   Add an item stack to allow matching ().  One side is "push X on
    the stack if previous atom matched".  Other side is "match with top of
    stack, pop it when it matches".  Use "\@pX" and "\@m"?
        Example: \(\(\@p).\{-}\@m\)*
7   Add a flag to "/pat/" to discard an error.  Useful to continue a mapping
    when a search fails.  Could be "/pat/E" (e is already used for end offset).
7   Add pattern item to use properties of Unicode characters.  In Perl it's
    "\p{L}" for a letter.  See Regular Expression Pocket Reference.
8   Would it be possible to allow ":23,45/pat/flags" to search for "pat" in
    lines 23 to 45?  Or does this conflict with Ex range syntax?
8   Allow identical pairs in 'matchpairs'.  Restrict the search to the current
    line.
7   Allow longer pairs in 'matchpairs'.  Use matchit.vim as an
    example.
8   Make it possible to define the character that "%" checks for in
    #if/#endif.  For nmake it's !if/!endif.
-   For "%" command: set hierarchy for which things include other things that
    should be ignored (like "*/" or "#endif" inside /* */).
    Also: use "%" to jump from start to end of syntax region and back.
    Alternative: use matchit.vim
8   A pattern like "\([^a]\+\)\+" takes an awful long time.  Recognize that
    the recursive "\+" is meaningless and optimize for it.
    This one is also very slow on "/* some comment */": "^\/\*\(.*[^/]\)*$".
7   Recognize "[a-z]", "[0-9]", etc. and replace them with the faster "\l" and
    "\d".
7   Add a way to specify characters in <C-M> or <Key> form.  Could be
    \%<C-M>.
8   Add an argument after ":s/pat/str/" for a range of matches.  For example,
    ":s/pat/str/#3-4" to replace only the third and fourth "pat" in a line.
8   When 'iskeyword' is changed the matches from 'hlsearch' may change. (Benji
    Fisher)  redraw if some options are set while 'hlsearch' is set?
8   Add an option not to use 'hlsearch' highlighting for ":s" and ":g"
    commands. (Kahn)  It would work like ":noh" is used after that command.
    Also: An extra flag to do this once, and a flag to keep the existing
    search pattern.
-   Make 'hlsearch' a local/global option, so that it can be disabled in some
    of the windows.
-   Add \%h{group-name}; to search for a specific highlight group.
    Add \%s{syntax-group}; to search for a specific syntax group.
-   Support Perl regexp.  Use PCRE (Perl Compatible RE) package. (Shade)
    Or translate the pattern to a Vim one.
    Don't switch on with an option for typed commands/mappings/functions, it's
    too confusing.  Use "\@@" in the pattern, to avoid incompatibilities.
8   Add a way to access the last substitute text, what is used for ":s//~/".
    Can't use the ~ register, it's already used for drag & drop.
-   Remember flags for backreferenced items, so that "*" can be used after it.
    Check with "\(\S\)\1\{3}". (Hemmerling)
8   Flags that apply to the whole pattern.
    This works for all places where a regexp is used.
    Add "\q" to not store this pattern as the last search pattern?
-   Add flags to search command (also for ":s"?):
    i   ignore case
    I   use case
    p   use Perl regexp syntax (or POSIX?)
    v   use Vi regexp syntax
```

```
    f   forget pattern, don't keep it for "n" command
    F   remember pattern, keep it for "n" command
    Perl uses these too:
    e   evaluate the right side as an expression (Perl only)
    m   multiple line expression (we don't need it)
    o   compile only once (Perl only)
    s   single line expression (we don't need it)
    x   extended regexp (we don't need it)
    When used after ":g" command, backslash needed to avoid confusion with the
    following command.
    Add 'searchflags' for default flags (replaces 'gdefault').
-   Add command to display the last used substitute pattern and last used
    pattern. (Margo)  Maybe make it accessible through a register (like "/
    for search string)?
7   Use T-search algorithm, to speed up searching for strings without special
    characters.  See C't article, August 1997.
-   Add 'fuzzycase' option, so that case doesn't matter, and '-' and '_' are
    equivalent (for Unix filenames).
-   Add 'v' flag to search command: enter Visual mode, with the matching text
    as Visual area. (variation on idea from Bertin)
-   Searching: "/this//that/" should find "that" after "this".
-   Add global search commands: Instead of wrapping at the end of the buffer,
    they continue in another buffer.  Use flag after search pattern:
    a   for the next file in the argument list
    f   for file in the buffer list
    w   for file edited in a window.
    e.g. "/pat/f".  Then "n" and "N" work through files too.  "f" flag also for
    ":s/pat/foo/f"???  Then when 'autowrite' and 'hidden' are both not set, ask
    before saving files: "Save modified buffer "/path/file"? (Yes/Hide/No
    Save-all/hide-All/Quit) ".
-   ":s/pat/foo/3": find 3rd match of "pat", like sed. (Thomas Koehler)
7   When searching with 'n' give message when getting back where the search
    first started.  Remember start of search in '/ mark.
7   Add option that scrolls screen to put cursor in middle of screen after
    search always/when off-screen/never.  And after a ":tag" command.  Maybe
    specify how many lines below the screen causes a redraw with the cursor in
    the middle (default would be half a screen, zero means always).
6   Support multiple search buffers, so macros can be made without side
    effects.
7   From xvim: Allow a newline in search patterns (also for :s, can delete
    newline).  Add BOW, EOW, NEWL, NLORANY, NLBUTANY, magic 'n' and 'r', etc.
    [not in xvim:] Add option to switch on matches crossing ONE line boundary.
7   Add ":iselect", a combination of ":ilist" and ":tselect". (Aaron) (Zellner)
    Also ":dselect".


Undo:
9   ":gundo" command: global undo.  Undoes changes spread over multiple files
    in the order they were made.  Also ":gredo".  Both with a count.  Useful
    when tests fail after making changes and you forgot in which files.
9   After undo/redo, in the message show whether the buffer is modified or
    not.
8   Search for pattern in undo tree, showing when it happened and the text
    state, so that you can jump to it.
8   Undo tree: visually show the tree somehow (Damian Conway)
    Show only the leaves, indicating how many changed from the branch and the
    timestamp?
    Put branch with most recent change on the left, older changes get more
    indent?
-   Make it possible to undo all the commands from a mapping, including a
    trailing unfinished command, e.g. for ":map K iX^[r".
-   When accidentally hitting "R" instead of Ctrl-R, further Ctrl-R is not
```

```
    possible, even when typing <Esc> immediately. (Grahn)  Also for "i", "a",
    etc.  Postpone saving for undo until something is really inserted?
8   When Inserting a lot of text, it can only be undone as a whole.  Make undo
    sync points at every line or word.  Could recognize the start of a new
    word (white space and then non-white space) and backspacing.
    Can already use CTRL-G u, but that requires remapping a lot of things.
8   Make undo more memory-efficient: Compare text before and after change,
    only remember the lines that really changed.
7   Add undo for a range of lines.  Can change these back to a previous
    version without changing the rest of the file.  Stop doing this when a
    change includes only some of these lines and changes the line count.  Need
    to store these undo actions as a separate change that can be undone.
-   For u_save() include the column number. This can be used to set '[ and '].
    And in the future the undo can be made more efficient (Webb).
-   In out-of-memory situations: Free allocated space in undo, and reduce the
    number of undo levels (with confirmation).
-   Instead of [+], give the number of changes since the last write: [+123].
    When undoing to before the last write, change this to a negative number:
    [-99].
-   With undo with simple line delete/insert: optimize screen updating.
-   When executing macro's: Save each line for undo only once.
-   When doing a global substitute, causing almost all lines to be changed,
    undo info becomes very big.  Put undo info in swap file??


Buffer list:
7   Command to execute a command in another buffer: ":inbuf {bufname} {cmd}".
    Also for other windows: ":inwin {winnr} {cmd}".  How to make sure that
    this works properly for all commands, and still be able to return to the
    current buffer/window?  E.g.: ":inbuf xxx only".
8   Add File.{recent_files} menu entries: Recently edited files.
    Ron Aaron has a plugin for this: mru.vim.
8   Unix: Check all uses of fnamecmp() and fnamencmp() if they should check
    inode too.
7   Add another number for a buffer, which is visible for the user.  When
    creating a new buffer, use the lowest number not in use (or the highest
    number in use plus one?).
7   Offer some buffer selection from the command line?  Like using ":ls" and
    asking for a buffer number. (Zachmann)
-   When starting to edit a file that is already in the buffer list, use the
    file name argument for the new short file name. (Webb)
-   Add an option to make ":bnext" and ":bprev" wrap around the end of the
    buffer list.  Also for ":next" and ":prev"?
7   Add argument to ":ls" which is a pattern for buffers to list.
    E.g. ":ls *.c". (Thompson)
7   Add expansion of buffer names, so that "*.c" is expanded to all buffer
    names.  Needed for ":bdel *.c", ":bunload *.c", etc.
8   Support for <afile> where a buffer name is expected.
7   Add an option to mostly use slashes in file names.  Separately for
    internal use and for when executing an external program?
8   Some file systems are case-sensitive, some are not.  Besides
    'wildignorecase' there might be more parts inside
    CASE_INSENSITIVE_FILENAME that are useful on Unix.


Swap (.swp) files:
8   If writing to the swap file fails, should try to open one in another
    directory from 'dir'.  Useful in case the file system is full and when
    there are short file name problems.
8   Also use the code to try using a short file name for the backup and swap
    file for the Win32 and Dos 32 bit versions.
8   When a file is edited by root, add $LOGNAME to know who did su.
```

```
8   When the edited file is a symlink, try to put the swap file in the same
    dir as the actual file.  Adjust FullName().  Avoids editing the same file
    twice (e.g. when using quickfix).  Also try to make the name of the backup
    file the same as the actual file?
    Use the code for resolve()?
7   When using 64 bit inode numbers, also store the top 32 bits.  Add another
    field for this, using part of bo_fname[], to keep it compatible.
7   When editing a file on removable media, should put swap file somewhere
    else.  Use something like 'r' flag in 'viminfo'.  'diravoid'?
    Also: Be able to specify minimum disk space, skip directory when not
    enough room.
7   Add a configure check for which directory should be used: /tmp, /var/tmp
    or /var/preserve.
-   Add an option to create a swap file only when making the first change to
    the buffer.  (Liang)  Or only when the buffer is not read-only.
-   Add option to set "umask" for backup files and swap files (Antwerpen).
    'backupumask' and 'swapumask'?  Or 'umaskback' and 'umaskswap'?
-   When editing a readonly file, don't use a swap file but read parts from the
    original file.  Also do this when the file is huge (>'maxmem').  We do
    need to load the file once to count the number of lines?  Perhaps keep a
    cached list of which line is where.


Viminfo:
7   Can probably remove the code that checks for a writable viminfo file,
    because we now do the chown() for root, and others can't overwrite someone
    else's viminfo file.
8   When there is no .viminfo file and someone does "su", runs Vim, a
    root-owned .viminfo file is created.  Is there a good way to avoid this?
    Perhaps check the owner of the directory.  Only when root?
8   Add argument to keep the list of buffers when Vim is started with a file
    name. (Schild)
8   Keep the last used directory of the file browser (File/Open menu).
8   Remember the last used register for "@@".
8   Remember the redo buffer, so that "." works after restarting.
8   Remember a list of last accessed files.  To be used in the
    "File.Open Recent" menu.  Default is to remember 10 files or so.
    Also remember which files have been read and written.  How to display
    this?
7   Also store the ". register (last inserted text).
7   Make it possible to store buffer names in viminfo file relative to some
    directory, to make them portable over a network. (Aaron)
6   Store a snapshot of the currently opened windows.  So that when quitting
    Vim, and then starting again (without a file name argument), you see the
    same files in the windows.  Use ":mksession" code?
-   Make marks present in .viminfo usable as file marks: Display a list of
    "last visited files" and select one to jump to.


Modelines:
8   Before trying to execute a modeline, check that it looks like one (valid
    option names).  If it's very wrong, silently ignore it.
    Ignore a line that starts with "Subject: ".
-   Add an option to whitelist options that are allowed in a modeline.  This
    would allow careful users to use modelines, e.g., only allowing
    'shiftwidth'.
-   Add an option to let modelines only set local options, not global ones
    such as 'encoding'.
-   When an option value is coming from a modeline, do not carry it over to
    another edited file?  Would need to remember the value from before the
    modeline setting.
-   Allow setting a variable from a modeline?  Only allow using fixed strings,
```

```
    no function calls, to avoid a security problem.
-   Allow ":doauto BufRead x.cpp" in modelines, to execute autocommands for
    .cpp files.
-   Support the "abbreviate" command in modelines (Kearns).  Careful for
    characters after <Esc>, that is a security leak.
-   Add option setting to ask user if he wants to have the modelines executed
    or not.  Same for .exrc in local dir.


Sessions:
8   DOS/Windows: ":mksession" generates a "cd" command where "aa\#bb" means
    directory "#bb" in "aa", but it's used as "aa#bb". (Ronald Hoellwarth)
7   When there is a "help.txt" window in a session file, restoring that
    session will not get the "LOCAL ADDITIONS" back.
8   With ":mksession" always store the 'sessionoptions' option, even when
    "options" isn't in it. (St-Amant)
8   When using ":mksession", also store a command to reset all options to
    their default value, before setting the options that are not at their
    default value.
7   With ":mksession" also store the tag stack and jump history. (Michal
    Malecki)
7   Persistent variables: "p:var"; stored in viminfo file and sessions files.


Options:
7   ":with option=value | command": temporarily set an option value and
    restore it after the command has executed.
8   Make "old" number options that really give a number of effects into string
    options that are a comma separated list.  The old number values should
    also be supported.
8   Add commands to save and restore an option, which also preserves the flag
    that marks if the option was set.  Useful to keep the effect of setting
    'compatible' after ":syntax on" has been used.
7   There is 'titleold', why is there no 'iconold'? (Chazelas)
7   Make 'scrolloff' a global-local option, so that it can be different in the
    quickfix window, for example. (Gary Holloway)
    Also do 'sidescrolloff'.


External commands:
8   When filtering text, redirect stderr so that it can't mess up the screen
    and Vim doesn't need to redraw it.  Also for ":r !cmd".
4   Set separate shell for ":sh", piping "range!filter", reading text "r !ls"
    and writing text "w !wc". (Deutsche)  Allow arguments for fast start (e.g.
    -f).
4   Allow direct execution, without using a shell.
4   Run an external command in the background.  But how about I/O in the GUI?
    Careful: don't turn Vim into a shell!
4   Add feature to disable using a shell or external commands.


Multiple Windows:
7   "vim -oO file ..." use both horizontal and vertical splits.
8   Add CTRL-W T: go to the top window in the column of the current window.
    And CTRL-W B: go to bottom window.
7   Use CTRL-W <Tab>, like alt-tab, to switch between buffers.  Repeat <Tab>
    to select another buffer (only loaded ones?), <BS> to go back, <Enter> to
    select buffer, <Esc> to go back to original buffer.
7   Make it possible to edit a new buffer in the preview window.  A script can
    then fill it with something.  ":popen"?
7   Add a 'tool' window: behaves like a preview window but there can be
    several.  Don't count it in only_one_window(). (Alexei Alexandrov)
```

```
6   Add an option to resize the shell when splitting and/or closing a window.
    ":vsp" would make the shell wider by as many columns as needed for the new
    window.  Specify a maximum size (or use the screen size).  ":close" would
    shrink the shell by as many columns as come available. (Demirel)
7   When starting Vim several times, instantiate a Vim server, that allows
    communication between the different Vims.  Feels like one Vim running with
    multiple top-level windows.  Esp. useful when Vim is started from an IDE
    too.  Requires some form of inter process communication.
-   Support a connection to an external viewer.  Could call the viewer
    automatically after some seconds of non-activity, or with a command.
    Allow some way of reporting scrolling and cursor positioning in the viewer
    to Vim, so that the link between the viewed and edited text can be made.


Marks:
8   Add ten marks for last changed files: ':0, ':1, etc.  One mark per file.
8   When cursor is first moved because of scrolling, set a mark at this
    position.  (Rimon Barr)  Use '-.
8   Add a command to jump to a mark and make the motion inclusive.  g'm and g`m?
8   The '" mark is set to the first line, even when doing ":next" a few times.
    Only set the '" mark when the cursor was really moved in a file.
8   Make `` and '', which would position the new cursor position in the middle
    of the window, restore the old topline (or relative position) from when
    the mark was set.
7   Make a list of file marks in a separate window.  For listing all buffers,
    matching tags, errors, etc.  Normal commands to move around.  Add commands
    to jump to the mark (in current window or new window).  Start it with
    ":browse marks"?
6   Add a menu that lists the Marks like ":marks". (Amerige)
7   For ":jumps", ":tags" and ":marks", for not loaded buffers, remember the
    text at the mark.  Highlight the column with the mark.
7   Highlight each mark in some way (With "Mark" highlight group).
    Or display marks in a separate column, like 'number' does.
7   Use d"m to delete rectangular area from cursor to mark m (like Vile's \m
    command).
7   Try to keep marks in the same position when:
    - replacing with a line break, like in ":s/pat/^M/", move marks after the
      line break column to the next line. (Acevedo)
    - inserting/deleting characters in a line.
5   Include marks for start/end of the current word and section.  Useful in
    mappings.
6   Add "unnamed mark" feature: Like marks for the ":g" command, but place and
    unplace them with commands before doing something with the lines.
    Highlight the marked lines somehow.


Digraphs:
7   Make "ga" show the keymap for a character, if it exists.
    Also show the code of the character after conversion to 'fileencoding'.
-   Use digraph table to tell Vim about the collating sequence of special
    characters?
8   Add command to remove one or more (all) digraphs. (Brown)
7   Support different sets of digraphs (depending on the character set?).  At
    least Latin1/Unicode, Latin-2, MS-DOS (esp. for Win32).


Writing files:
-   In vim_rename(), should lock "from" file when deleting "to" file for
    systems other than Amiga.  Avoids problems with unexpected longname to
    shortname conversion.
8   write mch_isdevice() for Amiga, Mac, VMS, etc.
8   When appending to a file, Vim should also make a backup and a 'patchmode'
```

```
    file.
8   'backupskip' doesn't write a backup file at all, a bit dangerous for some
    applications.  Add 'backupelsewhere' to write a backup file in another
    directory?  Or add a flag to 'backupdir'?
6   Add an option to write a new, numbered, backup file each time.  Like
    'patchmode', e.g., 'backupmode'.
6   Make it possible to write 'patchmode' files to a different directory.
    E.g., ":set patchmode=~/backups/*.orig". (Thomas)
6   Add an option to prepend something to the backup file name.  E.g., "#".
    Or maybe allow a function to modify the backup file name?
8   Only make a backup when overwriting a file for the first time.  Avoids
    losing the original when writing twice. (Slootman)
7   On non-Unix machines, also overwrite the original file in some situations
    (file system full, it's a link on an NFS partition).
7   When editing a file, check that it has been change outside of Vim more
    often, not only when writing over it.  E.g., at the time the swap file is
    flushed.  Or every ten seconds or so (use the time of day, check it before
    waiting for a character to be typed).
8   When a file was changed since editing started, show this in the status
    line of the window, like "[time]".
    Make it easier to reload all outdated files that don't have changes.
    Automatic and/or with a command.


Substitute:
8   Substitute with hex/unicode number "\%xff" and "\%uabcd".  Just like
    "\%uabcd" in search pattern.
8   Make it easier to replace in all files in the argument list.  E.g.:
    ":argsub/oldword/newword/".  Works like ":argdo %s/oldword/newword/g|w".
-   :s///p prints the line after a substitution.
-   With :s///c replace \&, ~, etc. when showing the replacement pattern.
8   With :s///c allow scrolling horizontally when 'nowrap' is effective.
    Also allow a count before the scrolling keys.
-   Add number option to ":s//2": replace second occurrence of string?  Or:
    :s///N substitutes N times.
-   Add answers to ":substitute" with 'c' flag, used in a ":global", e.g.:
    ":g/pat1/s/pat2/pat3/cg": 'A' do all remaining replacements, 'Q' don't do
    any replacements, 'u' undo last substitution.
7   Substitute in a block of text.  Use {line}.{column} notation in an Ex
    range, e.g.: ":1.3,$.5s" means to substitute from line 1 column 3 to the
    last line column 5.
5   Add commands to bookmark lines, display bookmarks, remove bookmarks,
    operate on lines with bookmarks, etc.  Like ":global" but with the
    possibility to keep the bookmarks and use them with several commands.
    (Stanislav Sitar)


Mouse support:
8   Add 'o' flag to 'mouse'?
7   Be able to set a 'mouseshape' for the popup menu.
8   Add 'mouse' flag, which sets a behavior like Visual mode, but automatic
    yanking at the button-up event.  Or like Select mode, but typing gets you
    out of Select mode, instead of replacing the text. (Bhaskar)
-   Implement mouse support for the Amiga console.
-   Using right mouse button to extend a blockwise selection should attach to
    the nearest corner of the rectangle (four possible corners).
-   Precede mouse click by a number to simulate double clicks?!?
-   When mouse click after 'r' command, get character that was pointed to.


Argument list:
6   Add command to put all filenames from the tag files in the argument list.
```

```
    When given an argument, only use the files where that argument matches
    (like `grep -l ident`) and jump to the first match.
6   Add command to form an args list from all the buffers?


Registers:
8   Don't display empty registers with ":display". (Etienne)
8   Add put command that overwrites existing text.  Should also work for
    blocks.  Useful to move text around in a table.  Works like using "R ^R r"
    for every line.
6   When yanking into the unnamed registers several times, somehow make the
    previous contents also available (like it's done for deleting).  What
    register names to use?  g"1, g"2, etc.?
-   When appending to a register, also report the total resulting number of
    lines.  Or just say "99 more lines yanked", add the "more".
-   When inserting a register in Insert mode with CTRL-R, don't insert comment
    leader when line wraps?
-   The ":@r" commands should take a range and execute the register for each
    line in the range.
-   Add "P" command to insert contents of unnamed register, move selected text
        to position of previous deleted (to swap foo and bar in " + foo")
8   Should be able to yank and delete into the "/ register.
    How to take care of the flags (offset, magic)?


Debug mode:
8   Add breakpoints for setting an option
8   Add breakpoints for assigning to a variable.
7   Store the history from debug mode in viminfo.
7   Make the debug mode history available with histget() et al.


Various improvements:
7   Add plugins for formatting?  Should be able to make a choice depending on
    the language of a file (English/Korean/Japanese/etc.).
    Setting the 'langformat' option to "chinese" would load the
    "format/chinese.vim" plugin.
    The plugin would set 'formatexpr' and define the function being called.
    Edward L. Fox explains how it should be done for most Asian languages.
    (2005 Nov 24)
    Alternative: patch for utf-8 line breaking. (Yongwei Wu, 2008 Feb 23)
7   [t to move to previous xml/html tag (like "vatov"), ]t to move to next
    ("vatv").
7   [< to move to previous xml/html tag, e.g., previous <li>. ]< to move to
    next <li>, ]< to next </li>, [< to previous </li>.
8   Add ":rename" command: rename the file of the current buffer and rename
    the buffer.  Buffer may be modified.
7   Instead of filtering errors with a shell script it should be possible to
    do this with Vim script.  A function that filters the raw text that comes
    from the 'makeprg'?
-   Add %b to 'errorformat': buffer number. (Yegappan Lakshmanan / Suresh
    Govindachar)
7   Add a command that goes back to the position from before jumping to the
    first quickfix location.  ":cbefore"?
7   Allow a window not to have a statusline.  Makes it possible to use a
    window as a buffer-tab selection.
8   Allow non-active windows to have a different statusline. (Yakov Lerner)
7   Add an invisible buffer which can be edited.  For use in scripts that want
    to manipulate text without changing the window layout.
8   Add a command to revert to the saved version of file; undo or redo until
    all changes are gone.
6   "vim -q -" should read the list of errors from stdin. (Gautam Mudunuri)
```

```
8   Add "--remote-fail": When contacting the server fails, exit Vim.
    Add "--remote-self": When contacting the server fails, do it in this Vim.
    Overrules the default of "--remote-send" to fail and "--remote" to do it
    in this Vim.
8   When Vim was started without a server, make it possible to start one, as
    if the "--servername" argument was given.  ":startserver <name>"?
8   No address range can be used before the command modifiers.  This makes
    them difficult to use in a menu for Visual mode.  Accept the range and
    have it apply to the following command.
8   Add the possibility to set 'fileformats' to force a format and strip other
    CR characters.  For example, for "dos" files remove CR characters at the
    end of the line, so that a file with mixed line endings is cleaned up.
    To just not display the CR characters: Add a flag to 'display'?
7   Some compilers give error messages in which the file name does not have a
    path.  Be able to specify that 'path' is used for these files.
7   Xterm sends <Esc>O3F for <M-End>.  Similarly for other <M-Home>, <M-Left>,
    etc.  Combinations of Alt, Ctrl and Shift are also possible.  Recognize
    these to avoid inserting the raw byte sequence, handle like the key
    without modifier (unless mapped).
6   Add "gG": like what "gj" is to "j": go to the N'th window line.
8   Add command like ":normal" that accepts <Key> notation like ":map".
9   Support ACLs on more systems.
7   Add ModeMsgVisual, ModeMsgInsert, etc. so that each mode message can be
    highlighted differently.
7   Add a message area for the user.  Set some option to reserve space (above
    the command line?).  Use an ":echouser" command to display the message
    (truncated to fit in the space).
7   Add %s to 'keywordprg': replace with word under the cursor. (Zellner)
8   Support printing on Unix.  Can use "lpansi.c" as an example. (Bookout)
8   Add put command that replaces the text under it.  Esp. for blockwise
    Visual mode.
7   Enhance termresponse stuff: Add t_CV(?): pattern of term response, use
    regexp: "\e\[[>?][0-9;]*c", but only check just after sending t_RV.
7   Add "g|" command: move to N'th column from the left margin (after wrapping
    and applying 'leftcol').  Works as "|" like what "g0" is to "0".
7   Support setting 'equalprg' to a user function name.
7   Highlight the characters after the end-of-line differently.
7   When 'whichwrap' contains "l", "$dl" should join lines?
8   Add an argument to configure to use $CFLAGS and not modify it? (Mooney)
8   Enabling features is a mix of configure arguments and defines in
    feature.h.  How to make this consistent?  Feature.h is required for
    non-unix systems.  Perhaps let configure define CONF_XXX, and use #ifdef
    CONF_XXX in feature.h?  Then what should min-features and max-features do?
8   Add "g^E" and "g^Y", to scroll a screen-full line up and down.
8   Add ":confirm" handling in open_exfile(), for when file already exists.
8   When quitting with changed files, make the dialog list the changed file
    and allow "write all", "discard all", "write some".  The last one would
    then ask "write" or "discard" for each changed file.  Patch in HierAssist
    does something like this. (Shah)
7   Use growarray for replace stack.
7   Have a look at viH (Hellenic or Greek version of Vim).  But a solution
    outside of Vim might be satisfactory (Haritsis).
3   Make "2d%" work like "d%d%" instead of "d2%"?
7   "g CTRL-O" jumps back to last used buffer.  Skip CTRL-O jumps in the same
    buffer.  Make jumplist remember the last ten accessed buffers?
7   Make it possible to set the size of the jumplist (also to a smaller number
    than the default). (Nikolai Weibull)
-   Add code to disable the CAPS key when going from Insert to Normal mode.
-   Set date/protection/etc. of the patchfile the same as the original file.
-   Use growarray for termcodes[] in term.c
-   Add <window-99>, like <cword> but use filename of 99'th window.
7   Add a way to change an operator to always work characterwise-inclusive
```

```
    (like "v" makes the operator characterwise-exclusive).  "x" could be used.
-   Make a set of operations on list of names: expand wildcards, replace home
    dir, append a string, delete a string, etc.
-   Remove using mktemp() and use tmpname() only?  Ctags does this.
-   When replacing environment variables, and there is one that is not set,
    turn it into an empty string?  Only when expanding options? (Hiebert)
-   Option to set command to be executed instead of producing a beep (e.g. to
    call "play newbeep.au").
-   Add option to show the current function name in the status line.  More or
    less what you find with "[[k", like how 'cindent' recognizes a function.
    (Bhatt).
-   "[r" and "]r": like "p" and "P", but replace instead of insert (esp. for
    blockwise registers).
-   Add 'timecheck' option, on by default.  Makes it possible to switch off the
    timestamp warning and question. (Dodt).
-   Add an option to set the time after which Vim should check the timestamps
    of the files.  Only check when an event occurs (e.g., character typed,
    mouse moved).  Useful for non-GUI versions where keyboard focus isn't
    noticeable.
-   Make 'smartcase' work even though 'ic' isn't set (Webb).
7   When formatting text, allow to break the line at a number of characters.
    Use an option for this: 'breakchars'?  Useful for formatting Fortran code.
-   Add flag to 'formatoptions' to be able to format book-style paragraphs
    (first line of paragraph has larger indent, no empty lines between
    paragraphs).  Complements the '2' flag.  Use '>' flag when larger indent
    starts a new paragraph, use '<' flag when smaller indent starts a new
    paragraph.  Both start a new paragraph on any indent change.
8   The 'a' flag in 'formatoptions' is too dangerous.  In some way only do
    auto-formatting in specific regions, e.g. defined by syntax highlighting.
8   Allow using a trailing space to signal a paragraph that continues on the
    next line (MIME text/plain; format=flowed, RFC 2646).  Can be used for
    continuous formatting.  Could use 'autoformat' option, which specifies a
    regexp which triggers auto-formatting (for one line).
    ":set autoformat=\\s$".
-   Be able to redefine where a sentence stops.  Use a regexp pattern?
-   Support multi-byte characters for sentences.  Example from Ben Peterson.
7   Add command "g)" to go to the end of a sentence, "g(" to go back to the
    end of a sentence. (Servatius Brandt)
-   Be able to redefine where a paragraph starts.  For "[[" where the '{' is
    not in column 1.
6   Add ":cdprev": go back to the previous directory.  Need to remember a
    stack of previous directories.  We also need ":cdnext".
7   Should ":cd" for MS-DOS go to $HOME, when it's defined?
-   Make "gq<CR>" work on the last line in the file.  Maybe for every operator?
-   Add more redirecting of Ex commands:
        :redir #>  bufname
        :redir #>> bufname   (append)
-   Give error message when starting :redir: twice or using END when no
    redirection was active.
-   Setting of options, specifically for a buffer or window, with
    ":set window.option" or ":set buffer.option=val".  Or use ":buffer.set".
    Also: "buffer.map <F1> quit".
6   Would it be possible to change the color of the cursor in the Win32
    console?  (Klaus Hast)
-   Add :delcr command:
                            *:delcr*
     :[range]delcr[!]   Check [range] lines (default: whole buffer) for lines
                        ending in <CR>.  If all lines end in <CR>, or [!] is
                        used, remove the <CR> at the end of lines in [range].
                        A CTRL-Z at the end of the file is removed.  If
                        [range] is omitted, or it is the whole file, and all
                        lines end in <CR> 'textmode' is set.  {not in Vi}
```

```
-   Should integrate addstar() and file_pat_to_reg_pat().
-   When working over a serial line with 7 bit characters, remove meta
    characters from 'isprint'.
-   Use fchdir() in init_homedir(), like in FullName().
-   In win_update(), when the GUI is active, always use the scrolling area.
    Avoid that the last status line is deleted and needs to be redrawn.
-   That "cTx" fails when the cursor is just after 'x' is Vi compatible, but
    may not be what you expect.  Add a flag in 'cpoptions' for this?  More
    general: Add an option to allow "c" to work with a null motion.
-   Give better error messages by using errno (strerror()).
-   Give "Usage:" error message when command used with wrong arguments (like
    Nvi).
-   Make 'restorescreen' option also work for xterm (and others), replaces the
    SAVE_XTERM_SCREEN define.
7   Support for ":winpos" In xterm: report the current window position.
-   Give warning message when using ":set t_xx=asdf" for a termcap code that
    Vim doesn't know about.  Add flag in 'shortmess'?
6   Add ":che <file>", list all the include paths which lead to this file.
-   For a commandline that has several commands (:s, :d, etc.) summarize the
    changes all together instead of for each command (e.g. for the rot13
    macro).
-   Add command like "[I" that also shows the tree of included files.
-   ":set sm^L" results in ":set s", because short names of options are also
    expanded.  Is there a better way to do this?
-   Add ":@!" command, to ":@" like what ":source!" is to ":source".
8   Add ":@:!": repeat last command with forceit set.
-   Add 't_normal': Used whenever t_me, t_se, t_ue or t_Zr is empty.
-   ":cab map test ^V| je", ":cunab map" doesn't work.  This is vi compatible!
-   CTRL-W CTRL-E and CTRL-W CTRL-Y should move the current window up or down
    if it is not the first or last window.
-   Include-file-search commands should look in the loaded buffer of a file (if
    there is one) instead of the file itself.
7   Change 'nrformats' to include the leader for each format.  Example:
        nrformats=hex:$,binary:b,octal:0
    Add setting of 'nrformats' to syntax files.
-   'path' can become very long, don't use NameBuff for expansion.
-   When unhiding a hidden buffer, put the same line at top of the window as
    the one before hiding it.  Or: keep the same relative cursor position (so
    many percent down the windows).
-   Make it possible for the 'showbreak' to be displayed at the end of the
    line.  Use a comma to separate the part at the end and the start of the
    line?  Highlight the linebreak characters, add flag in 'highlight'.
    Make 'showbreak' local to a window.
-   Some string options should be expanded if they have wildcards, e.g.
    'dictionary' when it is "*.h".
-   Use a specific type for number and boolean options, making it possible to
    change it for specific machines (e.g. when a long is 64 bit).
-   Add option for <Insert> in replace mode going to normal mode. (Nugent)
-   Add a next/previous possibility to "[^I" and friends.
-   Add possibility to change the HOME directory.  Use the directory from the
    passwd file? (Antwerpen)
8   Add commands to push and pop all or individual options. ":setpush tw",
    ":setpop tw", ":setpush all".  Maybe pushing/popping all options is
    sufficient.  ":setflush" resets the option stack?
    How to handle an aborted mapping?  Remember position in tag stack when
    mapping starts, restore it when an error aborts the mapping?
-   Change ":fixdel" into option 'fixdel', t_del will be adjusted each time
    t_bs is set? (Webb)
-   "gc": goto character, move absolute character positions forward, also
    counting newlines.  "gC" goes backwards (Weigert).
-   When doing CTRL-^, redraw buffer with the same topline. (Demirel)  Store
    cursor row and window height to redraw cursor at same percentage of window
```

```
    (Webb).
-   Besides remembering the last used line number of a file, also remember the
    column.  Use it with CTRL-^ et. al.
-   Check for non-digits when setting a number option (careful when entering
    hex codes like 0xff).
-   Add option to make "." redo the "@r" command, instead of the last command
    executed by it.  Also to make "." redo the whole mapping.  Basically: redo
    the last TYPED command.
-   Support URL links for ^X^F in Insert mode, like for "gf".
-   Support %name% expansion for "gf" on Windows.
-   Make "gf" work on "file://c:/path/name".  "file:/c:/" and "file:///c:/"
    should also work?
-   Add 'urlpath', used like 'path' for when "gf" used on a URL?
8   When using "gf" on an absolute file name, while editing a remote file
    (starts with scp:// or http://) should prepend the method and machine
    name.
-   When finding a URL or file name, and it doesn't exist, try removing a
    trailing '.'.
-   Add ":path" command modifier.  Should work for every command that takes a
    file name argument, to search for the file name in 'path'.  Use
    find_file_in_path().
-   Highlight control characters on the screen: Shows the difference between
    CTRL-X and "^" followed by "X" (Colon).
-   Integrate parsing of cmdline command and parsing for expansion.
-   Create a program that can translate a .swp file from any machine into a
    form usable by Vim on the current machine.
-   Add ":noro" command: Reset 'ro' flag for all buffers, except ones that have
    a readonly file.  ":noro!" will reset all 'ro' flags.
-   Add a variant of CTRL-V that stops interpretation of more than one
    character.  For entering mappings on the command line where a key contains
    several special characters, e.g. a trailing newline.
-   Make '2' option in 'formatoptions' also work inside comments.
-   Add 's' flag to 'formatoptions': Do not break when inside a string. (Dodt)
-   When window size changed (with the mouse) and made too small, set it back
    to the minimal size.
-   Add "]>" and "[<", shift comment at end of line (command;  /* comment */).
-   Should not call cursorcmd() for each vgetc() in getcmdline().
-   ":split file1 file2" adds two more windows (Webb).
-   Don't give message "Incomplete last line" when editing binary file.
-   Add ":a", ":i" for preloading of named buffers.
-   When entering text, keep other windows on same buffer updated (when a line
    entered)?
-   Check out how screen does output optimizing.  Apparently this is possible
    as an output filter.
-   In dosub() regexec is called twice for the same line.  Try to avoid this.
-   Window updating from memline.c: insert/delete/replace line.
-   Optimize ml_append() for speed, esp. for reading a file.
-   V..c should keep indent when 'ai' is set, just like [count]cc.
-   Updatescript() can be done faster with a string instead of a char.
-   Screen updating is inefficient with CTRL-F and CTRL-B when there are long
    lines.
-   Uppercase characters in Ex commands can be made lowercase?
8   Add option to show characters in text not as "|A" but as decimal ("^129"),
    hex ("\x81") or octal ("\201") or meta (M-x).  Nvi has the 'octal' option
    to switch from hex to octal.  Vile can show unprintable characters in hex
    or in octal.
7   Tighter integration with xxd to edit binary files.  Make it more
    easy/obvious to use.  Command line argument?
-   How does vi detect whether a filter has messed up the screen?  Check source.
    After ":w !command" a wait_return?
-   Improve screen updating code for doput() (use s_ins()).
-   With 'p' command on last line: scroll screen up (also for terminals without
```

```
    insert line command).
-   Use insert/delete char when terminal supports it.
-   Optimize screen redraw for slow terminals.
-   Optimize "dw" for long row of spaces (say, 30000).
-   Add "-d null" for editing from a script file without displaying.
-   In Insert mode: Remember the characters that were removed with backspace
    and re-insert them one at a time with <key1>, all together with <key2>.
-   Amiga: Add possibility to set a keymap.  The code in amiga.c does not work
    yet.
-   Implement 'redraw' option.
-   Add special code to 'sections' option to define something else but '{' or
    '}' as the start of a section (e.g. one shiftwidth to the right).
7   Allow using Vim in a pipe: "ls | vim -u xxx.vim - | yyy".  Only needs
    implementing ":w" to stdout in the buffer that was read from stdin.
    Perhaps writing to stdout will work, since stderr is used for the terminal
    I/O.
8   Allow opening an unnamed buffer with ":e !cmd" and ":sp !cmd".  Vile can
    do it.
-   Add commands like ]] and [[ that do not include the line jumped to.
-   When :unab without matching "from" part and several matching "to" parts,
    delete the entry that was used last, instead of the first in the list.
-   Add text justification option.
-   Set boolean options on/off with ":set paste=off", ":set paste=on".
-   After "inv"ing an option show the value: ":set invpaste" gives "paste is
    off".
-   Check handling of CTRL-V and '\' for ":" commands that do not have TRLBAR.
-   When a file cannot be opened but does exist, give error message.
-   Amiga: When 'r' protection bit is not set, file can still be opened but
    gives read errors.  Check protection before opening.
-   When writing check for file exists but no permission, "Permission denied".
-   If file does not exist, check if directory exists.
-   Settings edit mode: make file with ":set opt=xx", edit it, parse it as ex
    commands.
-   ":set -w all": list one option per line.
-   Amiga: test for 'w' flag when reading a file.
-   :table command (Webb)
-   Add new operator: clear, make area white (replace with spaces): "g ".
-   Add command to ":read" a file at a certain column (blockwise read?).
-   Add sort of replace mode where case is taken from the old text (Goldfarb).
-   Allow multiple arguments for ":read", read all the files.
-   Support for tabs in specific columns: ":set tabcol=8,20,34,56" (Demirel).
-   Add 'searchdir' option: Directories to search for file name being edited
    (Demirel).
-   Modifier for the put command: Change to linewise, charwise, blockwise, etc.
-   Add commands for saving and restoring options ":set save" "set restore",
    for use in macro's and the like.
-   Keep output from listings in a window, so you can have a look at it while
    working in another window.  Put cmdline in a separate window?
-   Add possibility to put output of Ex commands in a buffer or file, e.g. for
    ":set all".  ":r :set all"?
-   When the 'equalalways' option is set, creating a new window should not
    result in windows to become bigger.  Deleting a window should not result in
    a window to become smaller (Webb).
-   When resizing the whole Vim window, the windows inside should be resized
    proportionally (Webb).
-   Include options directly in option table, no indirect pointers.  Use
    mkopttab to make option table?
-   When doing ":w dir", where "dir" is a directory name, write the current
    file into that directory, with the current file name (without the path)?
-   Support for 'dictionary's that are sorted, makes access a lot faster
    (Haritsis).
-   Add "^Vrx" on the command line, replace with contents of register x.  Used
```

```
    instead of CTRL-R to make repeating possible. (Marinichev)
-   Add "^Vb" on the command line, replace with word before or under the
    cursor?
-   Support mapping for replace mode and "r" command (Vi doesn't do this)?
8   Sorting of filenames for completion is wrong on systems that ignore
    case of filenames.  Add 'ignorefncase' option.  When set, case in
    filenames is ignored for sorting them. Patch by Mike Williams:
    ~/vim/patches/ignorefncase.  Also change what matches?  Or use another
    option name.
8   Should be able to compile Vim in another directory, with $(srcdir) set to
    where the sources are. Add $(srcdir) in the Makefile in a lot of places.
    (Netherton)
6   Make it configurable when "J" inserts a space or not.  Should not add a
    space after "(", for example.
5   When inserting spaces after the end-of-line for 'virtualedit', use tabs
    when the user wants this (e.g., add a "tab" field to 'virtualedit').
    (Servatius Brandt)


From Elvis:
-   Use "instman.sh" to install manpages?
-   Add ":alias" command.
-   Search patterns:
      \@        match word under cursor.
    but do:
      \@w       match the word under the cursor?
      \@W       match the WORD under the cursor?
8   ":window" command:
    :win +      next window (up)
    :win ++     idem, wrapping
    :win -      previous window (down)
    :win --     idem, wrapping
    :win nr     to window number "nr"
    :win name   to window editing buffer "name"
7   ":cc" compiles a single file (default: current one).  'ccprg'   option is
    program to use with ":cc".  Use ":compile" instead of ":cc"?


From xvi:
-   CTRL-_ : swap 8th bit of character.
-   Add egrep-like regex type, like xvi (Ned Konz) or Perl (Emmanuel Mogenet)


From vile:
-   When horizontal scrolling, use '>' for lines continuing right of a window.
-   Support putting .swp files in /tmp: Command in rc.local to move .swp files
    from /tmp to some directory before deleting files.


Far future and "big" extensions:
-   Instead of using a Makefile and autoconf, use a simple shell script to
    find the C compiler and do everything with C code.  Translate something
    like an Aap recipe and configure.ac to C.  Avoids depending on Python,
    thus will work everywhere.  With batch file to find the C compiler it
    would also work on MS-Windows.
-   Make it easy to setup Vim for groups of users: novice vi users, novice
    Vim users, C programmers, xterm users, GUI users,...
-   Change layout of blocks in swap file: Text at the start, with '\n' in
    between lines (just load the file without changes, except for Mac).
    Indexes for lines are from the end of the block backwards.  It's the
    current layout mirrored.
-   Make it possible to edit a register, in a window, like a buffer.
```

```
-   Add stuff to syntax highlighting to change the text (upper-case keywords,
    set indent, define other highlighting, etc.).
-   Mode to keep C-code formatted all the time (sort of on-line indent).
-   Several top-level windows in one Vim session.  Be able to use a different
    font in each top-level window.
-   Allow editing above start and below end of buffer (flag in 'virtualedit').
-   Smart cut/paste: recognize words and adjust spaces before/after them.
-   Add open mode, use it when terminal has no cursor positioning.
-   Special "drawing mode": a line is drawn where the cursor is moved to.
    Backspace deletes along the line (from jvim).
-   Support for underlining (underscore-BS-char), bold (char-BS-char) and other
    standout modes switched on/off with , 'overstrike' option (Reiter).
-   Add vertical mode (Paul Jury, Demirel): "5vdw" deletes a word in five
    lines, "3vitextESC" will insert "text" in three lines, etc..
4   Recognize l, #, p as 'flags' to EX commands:
    :g/RE/#l shall print lines with line numbers and in list format.
    :g/RE/dp shall print lines that are deleted.
    POSIX: Commands where flags shall apply to all lines written: list,
    number, open, print, substitute, visual, &, z.  For other commands, flags
    shall apply to the current line after the command completes.  Examples:
    :7,10j #l Join the lines 7-10 and print the result in list
-   Allow two or more users to edit the same file at the same time.  Changes
    are reflected in each Vim immediately.  Could work with local files but
    also over the internet.  See http://www.codingmonkeys.de/subethaedit/.

vim:tw=78:sw=4:sts=4:ts=8:ft=help:norl:
vim: set fo+=n :
*develop.txt*   For Vim version 8.0.  Last change: 2017 Jul 31


                  VIM REFERENCE MANUAL    by Bram Moolenaar


Development of Vim.                                        *development*

This text is important for those who want to be involved in further developing
Vim.

1. Design goals         |design-goals|
2. Coding style         |coding-style|
3. Design decisions     |design-decisions|
4. Assumptions          |design-assumptions|

See the file README.txt in the "src" directory for an overview of the source
code.

Vim is open source software.  Everybody is encouraged to contribute to help
improving Vim.  For sending patches a context diff "diff -c" is preferred.
Also see http://vim.wikia.com/wiki/How_to_make_and_submit_a_patch.

==============================================================================
1. Design goals                                            *design-goals*

Most important things come first (roughly).

Note that quite a few items are contradicting.  This is intentional.  A
balance must be found between them.


VIM IS... VI COMPATIBLE                                    *design-compatible*

First of all, it should be possible to use Vim as a drop-in replacement for
```

Vi. When the user wants to, he can use Vim in compatible mode and hardly
notice any difference with the original Vi.

Exceptions:
- We don't reproduce obvious Vi bugs in Vim.
- There are different versions of Vi. I am using Version 3.7 (6/7/85) as a
  reference. But support for other versions is also included when possible.
  The Vi part of POSIX is not considered a definitive source.
- Vim adds new commands, you cannot rely on some command to fail because it
  didn't exist in Vi.
- Vim will have a lot of features that Vi doesn't have. Going back from Vim
  to Vi will be a problem, this cannot be avoided.
- Some things are hardly ever used (open mode, sending an e-mail when
  crashing, etc.). Those will only be included when someone has a good reason
  why it should be included and it's not too much work.
- For some items it is debatable whether Vi compatibility should be
  maintained. There will be an option flag for these.

VIM IS... IMPROVED *design-improved*

The IMproved bits of Vim should make it a better Vi, without becoming a
completely different editor. Extensions are done with a "Vi spirit".
- Use the keyboard as much as feasible. The mouse requires a third hand,
  which we don't have. Many terminals don't have a mouse.
- When the mouse is used anyway, avoid the need to switch back to the
  keyboard. Avoid mixing mouse and keyboard handling.
- Add commands and options in a consistent way. Otherwise people will have a
  hard time finding and remembering them. Keep in mind that more commands and
  options will be added later.
- A feature that people do not know about is a useless feature. Don't add
  obscure features, or at least add hints in documentation that they exist.
- Minimize using CTRL and other modifiers, they are more difficult to type.
- There are many first-time and inexperienced Vim users. Make it easy for
  them to start using Vim and learn more over time.
- There is no limit to the features that can be added. Selecting new features
  is one based on (1) what users ask for, (2) how much effort it takes to
  implement and (3) someone actually implementing it.

VIM IS... MULTI PLATFORM *design-multi-platform*

Vim tries to help as many users on as many platforms as possible.
- Support many kinds of terminals. The minimal demands are cursor positioning
  and clear-screen. Commands should only use key strokes that most keyboards
  have. Support all the keys on the keyboard for mapping.
- Support many platforms. A condition is that there is someone willing to do
  Vim development on that platform, and it doesn't mean messing up the code.
- Support many compilers and libraries. Not everybody is able or allowed to
  install another compiler or GUI library.
- People switch from one platform to another, and from GUI to terminal
  version. Features should be present in all versions, or at least in as many
  as possible with a reasonable effort. Try to avoid that users must switch
  between platforms to accomplish their work efficiently.
- That a feature is not possible on some platforms, or only possible on one
  platform, does not mean it cannot be implemented. [This intentionally
  contradicts the previous item, these two must be balanced.]

VIM IS... WELL DOCUMENTED *design-documented*

- A feature that isn't documented is a useless feature. A patch for a new

feature must include the documentation.
- Documentation should be comprehensive and understandable. Using examples is recommended.
- Don't make the text unnecessarily long. Less documentation means that an item is easier to find.

VIM IS... HIGH SPEED AND SMALL IN SIZE *design-speed-size*

Using Vim must not be a big attack on system resources. Keep it small and fast.
- Computers are becoming faster and bigger each year. Vim can grow too, but no faster than computers are growing. Keep Vim usable on older systems.
- Many users start Vim from a shell very often. Startup time must be short.
- Commands must work efficiently. The time they consume must be as small as possible. Useful commands may take longer.
- Don't forget that some people use Vim over a slow connection. Minimize the communication overhead.
- Items that add considerably to the size and are not used by many people should be a feature that can be disabled.
- Vim is a component among other components. Don't turn it into a massive application, but have it work well together with other programs.

VIM IS... MAINTAINABLE *design-maintain*

- The source code should not become a mess. It should be reliable code.
- Use the same layout in all files to make it easy to read |coding-style|.
- Use comments in a useful way! Quoting the function name and argument names is NOT useful. Do explain what they are for.
- Porting to another platform should be made easy, without having to change too much platform-independent code.
- Use the object-oriented spirit: Put data and code together. Minimize the knowledge spread to other parts of the code.

VIM IS... FLEXIBLE *design-flexible*

Vim should make it easy for users to work in their preferred styles rather than coercing its users into particular patterns of work. This can be for items with a large impact (e.g., the 'compatible' option) or for details. The defaults are carefully chosen such that most users will enjoy using Vim as it is. Commands and options can be used to adjust Vim to the desire of the user and its environment.

VIM IS... NOT *design-not*

- Vim is not a shell or an Operating System. It does provide a terminal window, in which you can run a shell or debugger. E.g. to be able to do this over an ssh connection. But if you don't need a text editor with that it is out of scope (use something like screen or tmux instead).
  A satirical way to say this: "Unlike Emacs, Vim does not attempt to include everything but the kitchen sink, but some people say that you can clean one with it. ;-)"
  To use Vim with gdb see: http://www.agide.org and http://clewn.sf.net.
- Vim is not a fancy GUI editor that tries to look nice at the cost of being less consistent over all platforms. But functional GUI features are welcomed.

==
2. Coding style *coding-style*

These are the rules to use when making changes to the Vim source code. Please
stick to these rules, to keep the sources readable and maintainable.

This list is not complete. Look in the source code for more examples.

MAKING CHANGES *style-changes*

The basic steps to make changes to the code:
1. Get the code from github. That makes it easier to keep your changed
   version in sync with the main code base (it may be a while before your
   changes will be included). You do need to spend some time learning git,
   it's not the most user friendly tool.
2. Adjust the documentation. Doing this first gives you an impression of how
   your changes affect the user.
3. Make the source code changes.
4. Check ../doc/todo.txt if the change affects any listed item.
5. Make a patch with "git diff". You can also create a pull request on
   github, but it's the diff that matters.
6. Make a note about what changed, preferably mentioning the problem and the
   solution. Send an email to the |vim-dev| maillist with an explanation and
   include the diff. Or create a pull request on github.

C COMPILER *style-compiler*

The minimal C compiler version supported is C89, also known as ANSI C.
Later standards don't add much and C89 is the widest supported.

One restriction that this implies: no // comments, only /* comments */.

USE OF COMMON FUNCTIONS *style-functions*

Some functions that are common to use, have a special Vim version. Always
consider using the Vim version, because they were introduced with a reason.

NORMAL NAME	VIM NAME	DIFFERENCE OF VIM VERSION
free()	vim_free()	Checks for freeing NULL
malloc()	alloc()	Checks for out of memory situation
malloc()	lalloc()	Like alloc(), but has long argument
strcpy()	STRCPY()	Includes cast to (char *), for char_u * args
strchr()	vim_strchr()	Accepts special characters
strrchr()	vim_strrchr()	Accepts special characters
isspace()	vim_isspace()	Can handle characters > 128
iswhite()	vim_iswhite()	Only TRUE for tab and space
memcpy()	mch_memmove()	Handles overlapped copies
bcopy()	mch_memmove()	Handles overlapped copies
memset()	vim_memset()	Uniform for all systems

NAMES *style-names*

Function names can not be more than 31 characters long (because of VMS).

Don't use "delete" or "this" as a variable name, C++ doesn't like it.

Because of the requirement that Vim runs on as many systems as possible, we
need to avoid using names that are already defined by the system. This is a
list of names that are known to cause trouble. The name is given as a regexp
pattern.

```
is.*()          POSIX, ctype.h
to.*()          POSIX, ctype.h

d_.*            POSIX, dirent.h
l_.*            POSIX, fcntl.h
gr_.*           POSIX, grp.h
pw_.*           POSIX, pwd.h
sa_.*           POSIX, signal.h
mem.*           POSIX, string.h
str.*           POSIX, string.h
wcs.*           POSIX, string.h
st_.*           POSIX, stat.h
tms_.*          POSIX, times.h
tm_.*           POSIX, time.h
c_.*            POSIX, termios.h
MAX.*           POSIX, limits.h
__.*            POSIX, system
_[A-Z].*        POSIX, system
E[A-Z0-9]*      POSIX, errno.h

.*_t            POSIX, for typedefs.  Use .*_T instead.

wait            don't use as argument to a function, conflicts with types.h
index           shadows global declaration
time            shadows global declaration
new             C++ reserved keyword
try             Borland C++ doesn't like it to be used as a variable.

clear           Mac curses.h
echo            Mac curses.h
instr           Mac curses.h
meta            Mac curses.h
newwin          Mac curses.h
nl              Mac curses.h
overwrite       Mac curses.h
refresh         Mac curses.h
scroll          Mac curses.h
typeahead       Mac curses.h

basename()      GNU string function
dirname()       GNU string function
get_env_value() Linux system function


VARIOUS                                                 *style-various*

Typedef'ed names should end in "_T": >
    typedef int some_T;
Define'ed names should be uppercase: >
    #define SOME_THING
Features always start with "FEAT_": >
    #define FEAT_FOO

Don't use '\"', some compilers can't handle it.  '"' works fine.

Don't use:
    #if HAVE_SOME
Some compilers can't handle that and complain that "HAVE_SOME" is not defined.
Use
    #ifdef HAVE_SOME
or
```

```
    #if defined(HAVE_SOME)


STYLE                                                   *style-examples*

General rule: One statement per line.

Wrong:      if (cond) a = 1;

OK:         if (cond)
                a = 1;

Wrong:      while (cond);

OK:         while (cond)
                ;

Wrong:      do a = 1; while (cond);

OK:         do
                a = 1;
            while (cond);

Wrong:      if (cond) {
               cmd;
               cmd;
            } else {
               cmd;
               cmd;
            }

OK:         if (cond)
            {
               cmd;
               cmd;
            }
            else
            {
               cmd;
               cmd;
            }

Use ANSI (new style) function declarations with the return type on a separate
indented line.

Wrong:  int function_name(int arg1, int arg2)

OK:     /*
         * Explanation of what this function is used for.
         *
         * Return value explanation.
         */
            int
        function_name(
            int         arg1,           /* short comment about arg1 */
            int         arg2)           /* short comment about arg2 */
        {
            int         local;          /* comment about local */

            local = arg1 * arg2;
```

```
SPACES AND PUNCTUATION                                        *style-spaces*

No space between a function name and the bracket:

Wrong:  func (arg);
OK:     func(arg);

Do use a space after if, while, switch, etc.

Wrong:  if(arg)         for(;;)
OK:     if (arg)        for (;;)

Use a space after a comma and semicolon:

Wrong:  func(arg1,arg2);        for (i = 0;i < 2;++i)
OK:     func(arg1, arg2);       for (i = 0; i < 2; ++i)

Use a space before and after '=', '+', '/', etc.

Wrong:  var=a*5;
OK:     var = a * 5;

In general: Use empty lines to group lines of code together.  Put a comment
just above the group of lines.  This makes it easier to quickly see what is
being done.

OK:     /* Prepare for building the table. */
        get_first_item();
        table_idx = 0;

        /* Build the table */
        while (has_item())
            table[table_idx++] = next_item();

        /* Finish up. */
        cleanup_items();
        generate_hash(table);

==============================================================================
3. Design decisions                                        *design-decisions*

Folding

Several forms of folding should be possible for the same buffer.  For example,
have one window that shows the text with function bodies folded, another
window that shows a function body.

Folding is a way to display the text.  It should not change the text itself.
Therefore the folding has been implemented as a filter between the text stored
in a buffer (buffer lines) and the text displayed in a window (logical lines).


Naming the window

The word "window" is commonly used for several things: A window on the screen,
the xterm window, a window inside Vim to view a buffer.
To avoid confusion, other items that are sometimes called window have been
given another name.  Here is an overview of the related items:

screen          The whole display.  For the GUI it's something like 1024x768
                pixels.  The Vim shell can use the whole screen or part of it.
```

shell The Vim application. This can cover the whole screen (e.g., when running in a console) or part of it (xterm or GUI).
window View on a buffer. There can be several windows in Vim, together with the command line, menubar, toolbar, etc. they fit in the shell.

Spell checking *develop-spell*

When spell checking was going to be added to Vim a survey was done over the available spell checking libraries and programs. Unfortunately, the result was that none of them provided sufficient capabilities to be used as the spell checking engine in Vim, for various reasons:

- Missing support for multi-byte encodings. At least UTF-8 must be supported, so that more than one language can be used in the same file. Doing on-the-fly conversion is not always possible (would require iconv support).
- For the programs and libraries: Using them as-is would require installing them separately from Vim. That's mostly not impossible, but a drawback.
- Performance: A few tests showed that it's possible to check spelling on the fly (while redrawing), just like syntax highlighting. But the mechanisms used by other code are much slower. Myspell uses a hashtable, for example. The affix compression that most spell checkers use makes it slower too.
- For using an external program like aspell a communication mechanism would have to be setup. That's complicated to do in a portable way (Unix-only would be relatively simple, but that's not good enough). And performance will become a problem (lots of process switching involved).
- Missing support for words with non-word characters, such as "Etten-Leur" and "et al.", would require marking the pieces of them OK, lowering the reliability.
- Missing support for regions or dialects. Makes it difficult to accept all English words and highlight non-Canadian words differently.
- Missing support for rare words. Many words are correct but hardly ever used and could be a misspelled often-used word.
- For making suggestions the speed is less important and requiring to install another program or library would be acceptable. But the word lists probably differ, the suggestions may be wrong words.

Spelling suggestions *develop-spell-suggestions*

For making suggestions there are two basic mechanisms:
1. Try changing the bad word a little bit and check for a match with a good word. Or go through the list of good words, change them a little bit and check for a match with the bad word. The changes are deleting a character, inserting a character, swapping two characters, etc.
2. Perform soundfolding on both the bad word and the good words and then find matches, possibly with a few changes like with the first mechanism.

The first is good for finding typing mistakes. After experimenting with hashtables and looking at solutions from other spell checkers the conclusion was that a trie (a kind of tree structure) is ideal for this. Both for reducing memory use and being able to try sensible changes. For example, when inserting a character only characters that lead to good words need to be tried. Other mechanisms (with hashtables) need to try all possible letters at every position in the word. Also, a hashtable has the requirement that word boundaries are identified separately, while a trie does not require this. That makes the mechanism a lot simpler.

Soundfolding is useful when someone knows how the words sounds but doesn't know how it is spelled. For example, the word "dictionary" might be written

as "daktonerie". The number of changes that the first method would need to try is very big, it's hard to find the good word that way. After soundfolding the words become "tktnr" and "tkxnry", these differ by only two letters.

To find words by their soundfolded equivalent (soundalike word) we need a list of all soundfolded words. A few experiments have been done to find out what the best method is. Alternatives:
1. Do the sound folding on the fly when looking for suggestions. This means walking through the trie of good words, soundfolding each word and checking how different it is from the bad word. This is very efficient for memory use, but takes a long time. On a fast PC it takes a couple of seconds for English, which can be acceptable for interactive use. But for some languages it takes more than ten seconds (e.g., German, Catalan), which is unacceptable slow. For batch processing (automatic corrections) it's too slow for all languages.
2. Use a trie for the soundfolded words, so that searching can be done just like how it works without soundfolding. This requires remembering a list of good words for each soundfolded word. This makes finding matches very fast but requires quite a lot of memory, in the order of 1 to 10 Mbyte. For some languages more than the original word list.
3. Like the second alternative, but reduce the amount of memory by using affix compression and store only the soundfolded basic word. This is what Aspell does. Disadvantage is that affixes need to be stripped from the bad word before soundfolding it, which means that mistakes at the start and/or end of the word will cause the mechanism to fail. Also, this becomes slow when the bad word is quite different from the good word.

The choice made is to use the second mechanism and use a separate file. This way a user with sufficient memory can get very good suggestions while a user who is short of memory or just wants the spell checking and no suggestions doesn't use so much memory.

Word frequency

For sorting suggestions it helps to know which words are common. In theory we could store a word frequency with the word in the dictionary. However, this requires storing a count per word. That degrades word tree compression a lot. And maintaining the word frequency for all languages will be a heavy task. Also, it would be nice to prefer words that are already in the text. This way the words that appear in the specific text are preferred for suggestions.

What has been implemented is to count words that have been seen during displaying. A hashtable is used to quickly find the word count. The count is initialized from words listed in COMMON items in the affix file, so that it also works when starting a new file.

This isn't ideal, because the longer Vim is running the higher the counts become. But in practice it is a noticeable improvement over not using the word count.

==
4. Assumptions *design-assumptions*

Size of variables:
char 8 bit signed
char_u 8 bit unsigned
int 32 or 64 bit signed (16 might be possible with limited features)
unsigned 32 or 64 bit unsigned (16 as with ints)
long 32 or 64 bit signed, can hold a pointer

Note that some compilers cannot handle long lines or strings. The C89

```
standard specifies a limit of 509 characters.

 vim:tw=78:ts=8:ft=help:norl:
*debug.txt*     For Vim version 8.0.  Last change: 2017 Jul 15


                  VIM REFERENCE MANUAL    by Bram Moolenaar


Debugging Vim                                              *debug-vim*

This is for debugging Vim itself, when it doesn't work properly.
For debugging Vim scripts, functions, etc. see |debug-scripts|

1. Location of a crash, using gcc and gdb       |debug-gcc|
2. Locating memory leaks                        |debug-leaks|
3. Windows Bug Reporting                        |debug-win32|

==============================================================================

1. Location of a crash, using gcc and gdb               *debug-gcc* *gdb*

When Vim crashes in one of the test files, and you are using gcc for
compilation, here is what you can do to find out exactly where Vim crashes.
This also applies when using the MingW tools.

1. Compile Vim with the "-g" option (there is a line in the src/Makefile for
   this, which you can uncomment).  Also make sure "strip" is disabled (do not
   install it, or use the line "STRIP = /bin/true").

2. Execute these commands (replace "11" with the test that fails): >
        cd testdir
        gdb ../vim
        run -u unix.vim -U NONE -s dotest.in test11.in

3. Check where Vim crashes, gdb should give a message for this.

4. Get a stack trace from gdb with this command: >
        where
<  You can check out different places in the stack trace with: >
        frame 3
<  Replace "3" with one of the numbers in the stack trace.

==============================================================================

2. Locating memory leaks                        *debug-leaks* *valgrind*

If you suspect Vim is leaking memory and you are using Linux, the valgrind
tool is very useful to pinpoint memory leaks.

First of all, build Vim with EXITFREE defined.  Search for this in MAKEFILE
and uncomment the line.

Use this command to start Vim:
>
        valgrind --log-file=valgrind.log --leak-check=full ./vim

Note: Vim will run much slower.  If your .vimrc is big or you have several
plugins you need to be patient for startup, or run with the "--clean"
argument.

There are often a few leaks from libraries, such as getpwuid() and
```

XtVaAppCreateShell(). Those are unavoidable. The number of bytes should be very small a Kbyte or less.

==

3. Windows Bug Reporting *debug-win32*

If the Windows version of Vim crashes in a reproducible manner, you can take some steps to provide a useful bug report.

3.1 GENERIC ~

You must obtain the debugger symbols (PDB) file for your executable: gvim.pdb for gvim.exe, or vim.pdb for vim.exe. The PDB should be available from the same place that you obtained the executable. Be sure to use the PDB that matches the EXE (same date).

If you built the executable yourself with the Microsoft Visual C++ compiler, then the PDB was built with the EXE.

Alternatively, if you have the source files, you can import Make_ivc.mak into Visual Studio as a workspace. Then select a debug configuration, build and you can do all kinds of debugging (set breakpoints, watch variables, etc.).

If you have Visual Studio, use that instead of the VC Toolkit and WinDbg.

For other compilers, you should always use the corresponding debugger: TD for a Vim executable compiled with the Borland compiler; gdb (see above |debug-gcc|) for the Cygwin and MinGW compilers.

debug-vs2005
3.2 Debugging Vim crashes with Visual Studio 2005/Visual C++ 2005 Express ~

First launch vim.exe or gvim.exe and then launch Visual Studio. (If you don't have Visual Studio, follow the instructions at |get-ms-debuggers| to obtain a free copy of Visual C++ 2005 Express Edition.)

On the Tools menu, click Attach to Process. Choose the Vim process.

In Vim, reproduce the crash. A dialog will appear in Visual Studio, telling you about the unhandled exception in the Vim process. Click Break to break into the process.

Visual Studio will pop up another dialog, telling you that no symbols are loaded and that the source code cannot be displayed. Click OK.

Several windows will open. Right-click in the Call Stack window. Choose Load Symbols. The Find Symbols dialog will open, looking for (g)vim.pdb. Navigate to the directory where you have the PDB file and click Open.

At this point, you should have a full call stack with vim function names and line numbers. Double-click one of the lines and the Find Source dialog will appear. Navigate to the directory where the Vim source is (if you have it.)

If you don't know how to debug this any further, follow the instructions at ":help bug-reports". Paste the call stack into the bug report.

If you have a non-free version of Visual Studio, you can save a minidump via the Debug menu and send it with the bug report. A minidump is a small file (<100KB), which contains information about the state of your process.

Visual C++ 2005 Express Edition cannot save minidumps and it cannot be
installed as a just-in-time debugger. Use WinDbg, |debug-windbg|, if you
need to save minidumps or you want a just-in-time (postmortem) debugger.

debug-windbg
3.3 Debugging Vim crashes with WinDbg ~

See |get-ms-debuggers| to obtain a copy of WinDbg.

As with the Visual Studio IDE, you can attach WinDbg to a running Vim process.
You can also have your system automatically invoke WinDbg as a postmortem
debugger. To set WinDbg as your postmortem debugger, run "windbg -I".

To attach WinDbg to a running Vim process, launch WinDbg. On the File menu,
choose Attach to a Process. Select the Vim process and click OK.

At this point, choose Symbol File Path on the File menu, and add the folder
containing your Vim PDB to the sympath. If you have Vim source available,
use Source File Path on the File menu. You can now open source files in WinDbg
and set breakpoints, if you like. Reproduce your crash. WinDbg should open the
source file at the point of the crash. Using the View menu, you can examine
the call stack, local variables, watch windows, and so on.

If WinDbg is your postmortem debugger, you do not need to attach WinDbg to
your Vim process. Simply reproduce the crash and WinDbg will launch
automatically. As above, set the Symbol File Path and the Source File Path.

To save a minidump, type the following at the WinDbg command line: >
        .dump vim.dmp
<
debug-minidump
3.4 Opening a Minidump ~

If you have a minidump file, you can open it in Visual Studio or in WinDbg.

In Visual Studio 2005: on the File menu, choose Open, then Project/Solution.
Navigate to the .dmp file and open it. Now press F5 to invoke the debugger.
Follow the instructions in |debug-vs2005| to set the Symbol File Path.

In WinDbg: choose Open Crash Dump on the File menu. Follow the instructions in
|debug-windbg| to set the Symbol File Path.

get-ms-debuggers
3.5 Obtaining Microsoft Debugging Tools ~

The Debugging Tools for Windows (including WinDbg) can be downloaded from
    http://www.microsoft.com/whdc/devtools/debugging/default.mspx
This includes the WinDbg debugger.

Visual C++ 2005 Express Edition can be downloaded for free from:
    http://msdn.microsoft.com/vstudio/express/visualC/default.aspx

==
 vim:tw=78:ts=8:ft=help:norl:
uganda.txt   For Vim version 8.0. Last change: 2013 Jul 06

                  VIM REFERENCE MANUAL   by Bram Moolenaar

                        uganda *Uganda* *copying* *copyright* *license*
SUMMARY

iccf *ICCF*

Vim is Charityware. You can use and copy it as much as you like, but you are encouraged to make a donation for needy children in Uganda. Please see |kcc| below or visit the ICCF web site, available at these URLs:

http://iccf-holland.org/
http://www.vim.org/iccf/
http://www.iccf.nl/

You can also sponsor the development of Vim. Vim sponsors can vote for features. See |sponsor|. The money goes to Uganda anyway.

The Open Publication License applies to the Vim documentation, see |manual-copyright|.

=== begin of license ===

VIM LICENSE

I) There are no restrictions on distributing unmodified copies of Vim except that they must include this license text. You can also distribute unmodified parts of Vim, likewise unrestricted except that they must include this license text. You are also allowed to include executables that you made from the unmodified Vim sources, plus your own usage examples and Vim scripts.

II) It is allowed to distribute a modified (or extended) version of Vim, including executables and/or source code, when the following four conditions are met:
1) This license text must be included unmodified.
2) The modified Vim must be distributed in one of the following five ways:
a) If you make changes to Vim yourself, you must clearly describe in the distribution how to contact you. When the maintainer asks you (in any way) for a copy of the modified Vim you distributed, you must make your changes, including source code, available to the maintainer without fee. The maintainer reserves the right to include your changes in the official version of Vim. What the maintainer will do with your changes and under what license they will be distributed is negotiable. If there has been no negotiation then this license, or a later version, also applies to your changes. The current maintainer is Bram Moolenaar <Bram@vim.org>. If this changes it will be announced in appropriate places (most likely vim.sf.net, www.vim.org and/or comp.editors). When it is completely impossible to contact the maintainer, the obligation to send him your changes ceases. Once the maintainer has confirmed that he has received your changes they will not have to be sent again.
b) If you have received a modified Vim that was distributed as mentioned under a) you are allowed to further distribute it unmodified, as mentioned at I). If you make additional changes the text under a) applies to those changes.
c) Provide all the changes, including source code, with every copy of the modified Vim you distribute. This may be done in the form of a context diff. You can choose what license to use for new code you add. The changes and their license must not restrict others from making their own changes to the official version of Vim.
d) When you have a modified Vim which includes changes as mentioned under c), you can distribute it without the source code for the changes if the following three conditions are met:
- The license that applies to the changes permits you to distribute the changes to the Vim maintainer without fee or restriction, and permits the Vim maintainer to include the changes in the official version of Vim without fee or restriction.

```
          - You keep the changes for at least three years after last
            distributing the corresponding modified Vim.  When the maintainer
            or someone who you distributed the modified Vim to asks you (in
            any way) for the changes within this period, you must make them
            available to him.
          - You clearly describe in the distribution how to contact you.  This
            contact information must remain valid for at least three years
            after last distributing the corresponding modified Vim, or as long
            as possible.
       e) When the GNU General Public License (GPL) applies to the changes,
          you can distribute the modified Vim under the GNU GPL version 2 or
          any later version.
    3) A message must be added, at least in the output of the ":version"
       command and in the intro screen, such that the user of the modified Vim
       is able to see that it was modified.  When distributing as mentioned
       under 2)e) adding the message is only required for as far as this does
       not conflict with the license used for the changes.
    4) The contact information as required under 2)a) and 2)d) must not be
       removed or changed, except that the person himself can make
       corrections.

III) If you distribute a modified version of Vim, you are encouraged to use
     the Vim license for your changes and make them available to the
     maintainer, including the source code.  The preferred way to do this is
     by e-mail or by uploading the files to a server and e-mailing the URL.
     If the number of changes is small (e.g., a modified Makefile) e-mailing a
     context diff will do.  The e-mail address to be used is
     <maintainer@vim.org>

IV)  It is not allowed to remove this license from the distribution of the Vim
     sources, parts of it or from a modified version.  You may use this
     license for previous Vim releases instead of the license that they came
     with, at your option.

=== end of license ===

Note:

- If you are happy with Vim, please express that by reading the rest of this
  file and consider helping needy children in Uganda.

- If you want to support further Vim development consider becoming a
  |sponsor|.  The money goes to Uganda anyway.

- According to Richard Stallman the Vim license is GNU GPL compatible.
  A few minor changes have been made since he checked it, but that should not
  make a difference.

- If you link Vim with a library that goes under the GNU GPL, this limits
  further distribution to the GNU GPL.  Also when you didn't actually change
  anything in Vim.

- Once a change is included that goes under the GNU GPL, this forces all
  further changes to also be made under the GNU GPL or a compatible license.

- If you distribute a modified version of Vim, you can include your name and
  contact information with the "--with-modified-by" configure argument or the
  MODIFIED_BY define.

==============================================================================
Kibaale Children's Centre              *kcc* *Kibaale*  *charity*
```

Kibaale Children's Centre (KCC) is located in Kibaale, a small town in the south of Uganda, near Tanzania, in East Africa. The area is known as Rakai District. The population is mostly farmers. Although people are poor, there is enough food. But this district is suffering from AIDS more than any other part of the world. Some say that it started there. Estimations are that 10 to 30% of the Ugandans are infected with HIV. Because parents die, there are many orphans. In this district about 60,000 children have lost one or both parents, out of a population of 350,000. And this is still continuing.

The children need a lot of help. The KCC is working hard to provide the needy with food, medical care and education. Food and medical care to keep them healthy now, and education so that they can take care of themselves in the future. KCC works on a Christian base, but help is given to children of any religion.

The key to solving the problems in this area is education. This has been neglected in the past years with president Idi Amin and the following civil wars. Now that the government is stable again, the children and parents have to learn how to take care of themselves and how to avoid infections. There is also help for people who are ill and hungry, but the primary goal is to prevent people from getting ill and to teach them how to grow healthy food.

Most of the orphans are living in an extended family. An uncle or older sister is taking care of them. Because these families are big and the income (if any) is low, a child is lucky if it gets healthy food. Clothes, medical care and schooling is beyond its reach. To help these needy children, a sponsorship program was put into place. A child can be financially adopted. For a few dollars a month KCC sees to it that the child gets indispensable items, is healthy, goes to school and KCC takes care of anything else that needs to be done for the child and the family that supports it.

Besides helping the child directly, the environment where the child grows up needs to be improved. KCC helps schools to improve their teaching methods. There is a demonstration school at the centre and teacher trainings are given. Health workers are being trained, hygiene education is carried out and households are stimulated to build a proper latrine. I helped setting up a production site for cement slabs. These are used to build a good latrine. They are sold below cost price.

There is a small clinic at the project, which provides children and their family with medical help. When needed, transport to a hospital is offered. Immunization programs are carried out and help is provided when an epidemic is breaking out (measles and cholera have been a problem).

donate

Summer 1994 to summer 1995 I spent a whole year at the centre, working as a volunteer. I have helped to expand the centre and worked in the area of water and sanitation. I learned that the help that the KCC provides really helps. When I came back to Holland, I wanted to continue supporting KCC. To do this I'm raising funds and organizing the sponsorship program. Please consider one of these possibilities:

1. Sponsor a child in primary school: 17 euro a month (or more).
2. Sponsor a child in secondary school: 25 euro a month (or more).
3. Sponsor the clinic: Any amount a month or quarter
4. A one-time donation

Compared with other organizations that do child sponsorship the amounts are very low. This is because the money goes directly to the centre. Less than 5% is used for administration. This is possible because this is a small organization that works with volunteers. If you would like to sponsor a child, you should have the intention to do this for at least one year.

How do you know that the money will be spent right? First of all you have my personal guarantee as the author of Vim. I trust the people that are working at the centre, I know them personally. Furthermore, the centre has been co-sponsored and inspected by World Vision, Save the Children Fund and is now under the supervision of Pacific Academy Outreach Society. The centre is visited about once a year to check the progress (at our own cost). I have visited the centre myself many times, starting in 1993. The visit reports are on the ICCF web site.

If you have any further questions, send me e-mail: <Bram@vim.org>.

The address of the centre is:
Kibaale Children's Centre
p.o. box 1658
Masaka, Uganda, East Africa

Sending money: *iccf-donations*

Check the ICCF web site for the latest information! See |iccf| for the URL.

USA: The methods mentioned below can be used.
Sending a check to the Nehemiah Group Outreach Society (NGOS) is no longer possible, unfortunately. We are looking for another way to get you an IRS tax receipt.
For sponsoring a child contact KCF in Canada (see below). US checks can be sent to them to lower banking costs.

Canada: Contact Kibaale Children's Fund (KCF) in Surrey, Canada. They take care of the Canadian sponsors for the children in Kibaale. KCF forwards 100% of the money to the project in Uganda. You can send them a one time donation directly. Please send me a note so that I know what has been donated because of Vim. Ask KCF for information about sponsorship.
Kibaale Children's Fund c/o Pacific Academy
10238-168 Street
Surrey, B.C. V4N 1Z4
Canada
Phone: 604-581-5353
If you make a donation to Kibaale Children's Fund (KCF) you will receive a tax receipt which can be submitted with your tax return.

Holland: Transfer to the account of "Stichting ICCF Holland" in Lisse. This will allow for tax deduction if you live in Holland.
Postbank, nr. 4548774
IBAN: NL95 INGB 0004 5487 74

Germany: It is possible to make donations that allow for a tax return. Check the ICCF web site for the latest information:
http://iccf-holland.org/germany.html

World: Use a postal money order. That should be possible from any country, mostly from the post office. Use this name (which is in my passport): "Abraham Moolenaar". Use Euro for the currency if possible.

Europe: Use a bank transfer if possible. Your bank should have a form that you can use for this. See "Others" below for the swift code and IBAN number.
Any other method should work. Ask for information about sponsorship.

```
Credit Card:    You can use PayPal to send money with a Credit card.  This is
                the most widely used Internet based payment system.  It's
                really simple to use.  Use this link to find more info:
                    https://www.paypal.com/en_US/mrb/pal=XAC62PML3GF8Q
                The e-mail address for sending the money to is:
                    Bram@iccf-holland.org
                For amounts above 400 Euro ($500) sending a check is
                preferred.

Others:         Transfer to one of these accounts if possible:
                    Postbank, account 4548774
                                Swift code: INGB NL 2A
                                IBAN: NL95 INGB 0004 5487 74
                        under the name "stichting ICCF Holland", Lisse
                    If that doesn't work:
                    Rabobank Lisse, account 3765.05.117
                                Swift code: RABO NL 2U
                        under the name "Bram Moolenaar", Lisse
                Otherwise, send a check in euro or US dollars to the address
                below.  Minimal amount: $70 (my bank does not accept smaller
                amounts for foreign check, sorry)

Address to send checks to:
                        Bram Moolenaar
                        Finsterruetihof 1
                        8134 Adliswil
                        Switzerland

This address is expected to be valid for a long time.

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