A

Installing *Vim*

YOU CAN OBTAIN *VIM* FROM THE WEB site at www.vim.org. This site contains the source to *Vim* as well as precompiled binaries for many different systems.

UNIX

You can get precompiled binaries for many different UNIX systems from www.vim.org. Go to www.vim.org, click the "Download Vim" link, and then follow the "Binaries Page" link.

This takes you to the "Binaries" page, which lists the various precompiled binaries for many different systems along with the links to download them.

Volunteers maintain the binaries, so they are frequently out of date. It is a good idea to compile your own UNIX version from the source. Also, creating the editor from the source allows you to control which features are compiled.

To compile and install the program, you'll need the following:

- A C compiler (GCC preferred)
- The GNU GZIP program (you can get it from www.gnu.org)

To obtain *Vim*, go to www.vim.org and click the "Download Vim" link.This page displays a list of sites that contain the software. Click a link to one that's near you.This takes you to a directory listing. Go into the "UNIX" directory and you'll find the sources for *Vim*.You'll need to download two files:

- vim-5.7-src.tar.gz
- vim-5.7-rt.tar.gz

Now unpack the sources using these commands:

```
$ gzip –u –d vim-5.7-src.tar.gz | tar xvf –

$ gzip –u –d vim-5.7-rt.tar.gz | tar xvf –
```

Build the Program

Go to the newly created *Vim* source directory:

```
$ cd vim-5.7/src
```

Now is a good time to read the files README.TXT and README_SRC.TXT.The instructions for compiling are in the file INSTALL. Configure the system with the following command:

```
$ ./configure
```

This configuration command assumes that you are going to install the system with a default set of features in the directory /usr/local. If you want to install in another directory, you need to use the **--prefix**=directory, where *directory* is the directory in which you want to install the editor.To install it in /apps/vim, for example, use this command:

```
$ ./configure --prefix=/apps/vim
```

The *Vim* editor has a lot of features that you can turn on and off at compile time. If this is the first time you are compiling *Vim*, you probably want to use the default set of features. After you have become familiar with the editor, you can enable the more exotic ones.

To get more information on *configure*, execute the following command:

```
$ ./configure --help
```

To find out about which features are available, see the file runtime/doc/various.txt or src/features.h.

Next compile the program with this command:

```
$ make
```

Finally, if all goes well, you can install it with the following command:

```
$ make install
```

Installation for Each UNIX User

Each UNIX user should make sure that *Vim* is in his path. If you have an EXRC file, copy it to VIMRC:

```
$ cp ~/.exrc ~/.vimrc
```

If you do not have an EXRC file, create an empty VIMRC file by executing the following command:

```
$ touch ~/.vimrc
```

Note

The presence of the VIMRC file turns on all the fancy features of *Vim*. If this file is not present, *Vim* tries very hard to look like *Vi*, even disabling some of its features to do so.

Installing on Microsoft Windows

To install the *Vim* program for Microsoft Windows, you'll need:

- The Windows binaries for Vim (gvim57.zip)
- The *Vim* runtime package (vim57rt.zip)
- A program to unpack the zip files

To download the *Vim* binaries, go to the *Vim* Web site, www.vim.org. Click "Download Vim." Do not click the "Binaries Page" link. Instead, select a mirror site and click the link provided.This takes you to a directory listing. Click the "pc" directory and download these files:

- gvim57.zip
- vim57rt.zip

If you already have a zip program, such as WinZip, installed, you can use it to unpack the sources. If not, go to the *Vim* home page (www.vim.org), scroll to the bottom of the page, and click the "Utilities" link. Click the "zip" link, which takes you to a link on an FTP site (ftp://ftp.uu.net/pub/archiving/zip). Follow this link to a directory listing. Select "WIN32" and then download the unz540xN.exe file.

Run this program to install the program InfoZip.

To install *Vim* itself, create a directory to be the root of your installation (for example: C:\VIM). Unzip the following archives into this directory:

```
vim\pc\gvim57.zip
vim\pc\vim57rt.zip
```

Open a MS-DOS command window, go the directory in which you installed *Vim* and execute the following command:

```
C:> install
```

The installer starts:

```
This program sets up the installation of Vim 5.7
It prepares the _VIMRC file, $VIM and the EXECUTAB.S
Do you want to continue? (Y/N)
```

Answer Y to continue. Installation continues:

```
Choose the default way to run Vim:
[1] Conventional Vim setup
[2] With syntax highlighting and other features switched on
[3] Vi compatible
Choice:
```

Because we want all the goodies, choose 2. (If you want to, choose 1. Do not choose 3, however; otherwise you turn off all the distinct features of this editor.)

```
Choose the way text is selected:
[1] With Visual mode (the UNIX way)
[2] With Select mode (the Windows way)
[3] Mouse with Select mode, keys with Visual mode
Choice:
```

To be compatible with the examples in this book, choose 1. (You can later change it if you want to customize your editor.)

```
You have chosen:
[2] With syntax highlighting and other features switched on
[1] With Visual mode (the UNIX way)
(You can adjust your _VIMRC file afterwards)

Do you want to write the file C:\VIM\VIM\_VIMRC? (Y/N)
```

Answer Y to continue.The editor creates the file and then asks the following:

```
I can append a command to C:\AUTOEXEC.BAT to set $VIM.
(This will not work if C:\AUTOEXEC.BAT contains sections)
Do you want me to append to your C:\AUTOEXEC.BAT (Y/N)
```

Answer Y if you want to be able to run *Vim* from within an MS-DOS window.

```
I can install an entry in the popup menu for the right
mouse button, so that you can edit any file with Vim.
Do you want me to do this? (Y/N)
```

Note

These installation instructions install only the GUI version of *Vim* named *gvim*. If you are doing a lot of editing inside the MS-DOS command-prompt windows, you might want to install the package:

vim.org\pc\vim56w32.zip

The console-mode *Vim* (for example, the non-BUI version on Windows) is not as good as the GUI version on Windows, for the simple reason that Windows does not support console mode very well or consistently among Windows versions.

This one is up to you.You can always choose N and reinstall later if you want this feature.

```
That finishes the installation. Happy Vimming!
```

Common Installation Problems and Questions

This section describes some of the common problems that occur when installing *Vim* and suggests some solutions. It also contains answers to many installation questions.

I Do Not Have Root Privileges. How Do I Install *Vim*? (UNIX)

Use the following configuration command to install *Vim* in a directory called $HOME/vim:

```
$ configure --prefix=$HOME/vim
```

This gives you a personal copy of *Vim*.You need to put $HOME/vim/bin in your path to access the editor.

The Colors Are Not Right on My Screen. (UNIX)

Check your terminal settings by using the following command:

```
$ echo $TERM
```

If the terminal type listed is not correct, fix it. UNIX has a database called termcap, which describes the capabilities of your terminal. Almost all xterm programs support color. Frequently, however, the termcap entry for xterm defines a terminal without color.To get color to work, you might have to tell the system that you have an xtermc or cxterm terminal. (Another solution is to always use the GUI version of *Vim* called *gvim*.)

I Am Using RedHat Linux. Can I Use the *Vim* That Comes with the System?

By default RedHat installs a minimal version of *Vim*. Check your RPM packages for something named *Vim*-enchanced-*version*.rpm and install that.

How Do I Turn Syntax Coloring On? (All)

Use the following command:

```
:syntax on
```

What Is a Good *vimrc* File to Use? (All)

See the www.vim.org Web site for several good examples.

UNIX Source Checklist

1. Start at www.vim.org.
2. Click "Download *Vim*".
3. Select the mirror site closest to you.
4. Click "UNIX".
5. Click "vim-5.7-src.tar.gz" to download this file.
6. Click "vim-5.7-rt.tar.gz" to download this file.
7. On your local system, execute these commands:

```
$ gzip –u –d vim-5.7-src.tar.gz | tar xvf –
$ gzip –u –d vim-5.7-rt.tar.gz | tar xvf –
```

8. Configure and build the program with these commands:

```
$ cd vim-5.7/src
$ ./configure –prefix=<directory>
$ make
$ make install
```

<directory> is the directory where *Vim* is to be installed.

Microsoft Windows Checklist

1. Start at www.vim.org.
2. Click "Download *Vim*".
3. Select the mirror nearest you and click it.
4. Click "pc".
5. Click "gvim56.zip" to download this file.
6. Click "vim57rt.zip: to download this file.
7. Unzip these files into the installation directory on your machine. (If you do not have an UNZIP program, see the instructions in the following section.)
8. Execute the installation script with this command:

```
C:> install
```

Installing the InfoZip Program

1. Start at the Web site www.vim.org.
2. Near the end of the page, you'll find the "Utilities" link. Click it.
3. Click "zip".
4. Click ftp://ftp.uu.net/pub/archiving/zip.
5. Click "WIN32".
6. Click "unz540xN.exe" to download this file.
7. Run the program "unz540xN.exe" to install InfoZip.

B

The <> Key Names

THIS APPENDIX PROVIDES A QUICK reference for the <> key names in *Vim*.

The Function Keys

<F1>	<F2>	<F3>	<F4>	<F5>	<F6>
<F7>	<F8>	<F9>	<F10>	<F11>	<F12>
<F13>	<F14>	<F15>	<F16>	<F17>	<F18>
<F19>	<F20>	<F21>	<F22>	<F23>	<F24>
<F25>	<F26>	<F27>	<F28>	<F29>	<F30>
<F31>	<F32>	<F33>	<F34>	<F35>	

Line Endings

<CR>	<Return>	<Enter>
<LF>	<LineFeed>	
<NL>	<NewLine>	

Other Special Characters

<BS> <BackSpace>
<Ins> <Insert>
<Del> <Delete>

Editing Keys

<End> <Home> <PageDown> <PageUp>

Arrow Keys

<Left> <Right> <Up> <Down>

Keypad Keys

<kDivide> <kEnd> <kEnter> <kHome> <kMinus> <kMultiply>
<kPlus> <kPageDown> <kPageUp>

VT100 Special Keys

The VT100 terminal has an extra set of function keys, as follows:

<xF1> <xF2> <xF3> <xF4> <xEnd> <xHome>

Printable Characters

<Bar> |
<Bslash> \
<Space>
<Tab>
<Lt> <

Other Keys

<Esc> <Help> <Nul> <Undo>

Termcap Entries

On UNIX systems, the Termcap or Terminfo database contains a description of the terminal, including function keys.The special key <t_*XX*> represents the key defined by *XX* Termcap entry.

See your UNIX documentation for a complete list of keys. One way to get a list (for most systems) is to execute the following command:

```
$ man terminfo
```

Mouse Actions

<LeftDrag>	<Mouse>
<LeftMouse>	<MouseDown>
<LeftRelease>	<MouseUp>
<MiddleDrag>	<RightDrag>
<MiddleMouse>	<RightMouse>
<MiddleRelease>	<RightRelease>

Modifiers

M	Meta (Alt)
C	Control
S	Shift
D	Macintosh command key

Mouse Modifiers

<Blank>	Mouse button one
2	Mouse button two
3	Mouse button three
4	Mouse button four

Note

If you want to find the name of a key on the keyboard, you can go into Insert mode and press **CTRL-K** *key*. The <> name of the key will be inserted. This works for the function keys and many other keys.

C

Normal-Mode Commands

[count] **<BS>** Move count characters to the left. See the **'backspace'** option to change this to delete rather than backspace. (Same as: **<Left>**, **CTRL-H**, **CTRL-K**, **h**. See page 6.)

[count] **<C-End>** Move to the end of line *count*. If no count is specified, go to the end of the file. (Same as: **G**. See pages 18, 40, 161, and 188.)

[count] **<C-Home>** Move to the start (first non-blank character) of line *count*. Default is the beginning of the file. (Same as: **gg**. See page 156.)

[count] **<C-Left>** Move *count* WORDS backward. (Same as: **B**. See page 186.)

<C-LeftMouse> Jump to the location of the tag whose name is under the cursor. (Same as: **CTRL-]**, **g<LeftMouse>**. See pages 12, 79, and109.)

<C-Right> Move *count* WORDS forward. (Same as: **w**. See page 86.)

[count] **count<C-RightMouse>** Jump to a previous entry in the tag stack. (Same as: **CTRL-T**, **g<RightMouse>**. See pages 12, 80-81, and 109.)

[count] **<CR>** Move down *count* lines. Cursor is positioned on the first nonblank character on the line. (Same as: **<ENTER>**, **CTRL-M**, and +. See page 187.)

["{register}] [count] **<Del>** Delete characters. If a "{*register* } is present, the deleted text is stored in it. (Same as: **x**. See pages 7, 13, 36-37, 160, and 196-197.)

[count] **<Down>** Move *count* lines down. (Same as: **<NL>**, **CTRL-J**, **CTRL-N**, **j**. See pages 6 and 235.)

[count] **<End>** Move the cursor to the end of the line. If a *count* is present, move to the end of the *count* line down from the current one. (Same as: **<kEnd>**, **$**. See pages 16 and 234.)

[count] **<Enter>** Move down *count* lines. (Default = 1.) Cursor is positioned on the first nonblank character on the line. (Same as: **<CR>**, **CTRL-M**, **+**. See pages 7 and 187.)

<F1> Go to the initial help screen. (Same as: **<Help>**, **:h**, **:help**. See pages 11, 13, and 157.)

<F8> Toggle between left-to-right and right-to-left modes. (See page 174.)

<F9> Toggles the encoding between ISIR-3342 standard and *Vim* extended ISIR-3342 (supported only in right-to-left mode when **'fkmap'** [Farsi] is enabled). (See page 176.)

<Help> Go to the initial help screen. (Same as: **<F1>**, **:h**, **:help**. See pages 11 and 157.)

<Home> Move to the first character of the line. (Same as: **<kHome>**. See page 16.)

[count] **<Insert>text<Esc>** Insert text. If *count* is present, the text will be inserted *count* times. (Same as: **i**. See pages 5 and 9.)

[count] **<kEnd>** Move the cursor to the end of the line. If a *count* is present, move to the end of the *count* line down from the current one. (Same as: **<End>**, **$**. See page 16.)

<kHome> Move to the first character of the line. (Same as: **<Home>**. See page 16.)

[count] **<Left>** Move left *count* characters. (Same as: **<BS>**, **CTRL-H**, **CTRL-K**, **h**. See page 6.)

<LeftMouse> Move the text cursor to the location of the mouse cursor. (See pages 109 and 332.)

["register] **<MiddleMouse>** Insert the text in register at the location of the mouse cursor. (Same as: **P**. See pages 109 and 332.)

<MouseDown> Scroll three lines down. (See page 332.)

<MouseUp> Scroll three lines up. (See page 332.)

[count] **<M>** Move *count* lines down. (See page 192.)

[count] **<PageDown>** Scroll *count* pages forward. (Same as: **<S-Down>**, **CTRL-F**. See page 192.)

[count] **<PageUp>** Scroll the window *count* pages backward. (Same as: **<S-Up>**, **CTRL-B**. See page 192.)

£ Same as £. (See page 206.)

[count] **<Right>** Move right *count* characters. (Same as: **<Space>**, **l**. See page 6.)

<RightMouse> Start select mode with the text from the text cursor to the mouse cursor highlighted. (See pages 109 and 332.)

[count] **<S-Down>** Scroll *count* pages forward. If you are running Windows GUI version, **<S-Down>** enters visual mode and selects down. (Same as: **<PageDown>**, **CTRL-F**. See page 192.)

[count] **<S-Left>** Move left *count* words. (Same as: **b**. See page 16.)

[count] **<S-LeftMouse>** Find the next occurrence of the word under the cursor. (See page 109.)

<S-MouseDown> Scroll a full page upthree lines down. (See page 332.)

<S-MouseUp> Scroll a full page down three lines up. (See page 332.)

[count] **<S-Right>** Move *count* words forward. (Same as: **w**. See pages 16, 20, and 184.)

[count] **<S-RightMouse>** Search backward for the word under the cursor. (See page 109.)

[count] **<S-Up>** Scroll *count* pages up. (Same as: **<PageUp>**, **CTRL-B**. See page 192.)

[count] **<Space>** Move *count* spaces to the right. (Same as: **<Right>**, **l**. See page 6.)

[count] **<Tab>** Go to the *count* position in the jump list. (Same as: **CTRL-I**. See page 189.)

[count] **<Undo>** Undo the last *count* changes. (Same as: **u**. See page 8.)

[count] **<Up>** Move *count* lines up. (Same as: **CTRL-P**, **k**. See pages 6 and 13.)

CTRL-\ CTRL-N Enter normal mode from any other mode. (See page 58.)

CTRL-] Jump to the function whose name is under the cursor. (In the help system, jump to the subject indicated by a hyperlink.) (Same as: **<C-LeftMouse>**, **g<LeftMouse>**. See pages 12 and 79.)

[count] **CTRL-^** If a *count* is specified, edit the *count* file on the command line. If no *count* is present, edit the previously edited file.Thus repeated **CTRL-^** can be used to toggle rapidly between two files. (See pages 43 and 244.)

CTRL-_ Switch between English and a foreign language keyboard. (See pages 176-177.)

[count] **CTRL-A** Add *count* to the number under the cursor. If no *count* is specified, increment the number. (See pages 43 and 244.)

[count] **CTRL-B** Move back *count* screens. (Default = 1.) (Same as: **<PageUp>**, **<S-Up>**. See pages 191-192.)

CTRL-BREAK Interrupt search (same as **CTRL-C**). (See page 206.)

CTRL-C Interrupt search. (same as **CTRL-BREAK**). (See page 206.)

[count] **CTRL-D** Move down the number of lines specified by the 'scroll' option. If a *count* is specified, set the 'scroll' option to *count* and then move down. (See pages 20 and 190-192.)

[count] **CTRL-E** Move down *count* lines. (See page 192.)

[count] **CTRL-F** Scroll the window *count* pages forward. (Same as: **<PageDown>**, **<S-Down>**. See page 192.)

CTRL-G Display the current file and location of the cursor within that file. (Same as: **:file**. See pages 19-20 and 190.)

1 CTRL-G Same as **CTRL-G**, but include the full path in the filename. (See pages 19-20 and 190.)

2 CTRL-G Same as **1 CTRL-G**, but adds a buffer number. (See pages 19-20 and 190.)

[count] **CTRL-H** Move *count* characters to the left. See the **'backspace'** option to change this to delete rather than backspace. (Same as: **<BS>**, **<Left>**, **CTRL-K**, **h**. See page 6.)

[count] **CTRL-I** Jump to the *count* next item in the jump list. (Same as: **<Tab>**. See page 189.)

[count] **CTRL-J** Move down *count* lines. (Same as: **<Down>**, **<NL>**, **CTRL-J**, **j**. See pages 6 and 235.)

[count] **CTRL-K** Move *count* characters to the left. (Same as: **<BS>**, **<Left>**, **CTRL-H**, **h**. See page 25.)

CTRL-L Redraw screen. (See page 156.)

CTRL-L Leave insert mode if insertmode is set. (See page 179.)

CTRL-M Copy **<CR>** entry. (Same as **<CR>**, +. See page 187.)

[count] **CTRL-N** Move *count* lines down. (Same as: **<Down>**, **<NL>**, **CTRL-J**, **j**. See pages 6 and 235.)

[count] **CTRL-O** Jump to the *count* previous item in the jump list. (See page 189.)

[count] **CTRL-P** Move *count* lines upward. (Same as: **<Up>**, **k**. See pages 6 and 13.)

CTRL-Q Used by some terminals to start output after it was stopped by **CTRL-S**. (See page 156.)

CTRL-R Redo the last change that was undone. (See page 8.)

CTRL-S Used by some terminals to stop output. (See page 156.)

[count] **CTRL-T** Go back *count* tags. If the current buffer has been modified, this command fails unless the force (**!**) option is present.When using the help system, this command returns to the location you were at before making the last hyperlink jump. (Same as: **<C-RightMouse>**, **g<RightMouse>**. See pages 12 and 80-81.)

[count] **CTRL-U** Move up the number of lines specified by the **'scroll'** option. If a *count* is specified, set the **'scroll'** option to *count* and then scroll up. (See pages 20 and 190-192.)

CTRL-V Start visual block mode. (See pages 57 and 60.)

[count] **CTRL-W<Down>** Move down a window. If a *count* is specified, move to window number *count*. (Same as: **CTRL-W CTRL-J**, **CTRL-Wj**. See page 46.)

[count] **CTRL-W<Up>** Move up a window. If a *count* is specified, move to window number *count*. (Same as: **CTRL-W CTRL-K**, **CTRL-Wk**. See page 46.)

[count] **CTRL-W CTRL-]** Split the current window and jump to the function whose name is under the cursor. If a *count* is specified, it is the height of the new window. (Same as: **CTRL-W]**. See page 82.)

CTRL-W CTRL-^ Split the window and edit the alternate file. If a *count* is specified, split the window and edit the *count* file on the command line. (Same as: **CTRL-W^**. See page 244.)

[count] **CTRL-W CTRL-_** Set the height of the current window to *count*. (Same as: **CTRL-W+**, **CTRL-W-**, **CTRL-W_**, :resize. See page 49.)

CTRL-W CTRL-B Move to the bottom window. (Same as: **CTRL-Wb**. See page 240.)

CTRL-W CTRL-C Cancel any pending window command. (See page 46.)

CTRL-W CTRL-D Split the window and find the definition of the word under the cursor. If the definition cannot be found, do not split the window. (Same as: **CTRL-Wd**. See page 245.)

CTRL-W CTRL-F Split the window and edit the file whose name is under the cursor. Looks for the file in the current directory, and then all the directories specified by the **'path'** option. (Same as: **CTRL-Wf**. See page 245.)

[count] **CTRL-W CTRL-G CTRL-]** **:split** followed a **CTRL-]**. If a *count* is specified, make the new window *count* lines high. (Same as: **CTRL-Wg CTRL-]**, **CTRL-Wg**. See page 245.)

[count] **CTRL-W CTRL-G }** Do a :ptjump on the word under the cursor. If a *count* is specified, make the new window *count* lines high. (Same as: **CTRL-W CTRL-G}**. See page 277.)

[count] **CTRL-W CTRL-I** Split the window and search for the *count* occurrence of the word under the cursor. Start the search at the beginning of the file. (Same as: **CTRL-Wi**. See page 244.)

[count] **CTRL-W CTRL-J** Move down a window. If a *count* is specified, move to window number *count*. (Same as: **CTRL-W<Down>**, **CTRL-Wj**. See page 46.)

[count] **CTRL-W CTRL-K** Move *count* windows up. (Same as: **CTRL-W<Up>**, **CTRL-Wk**. See page 46.)

CTRL-W CTRL-N Split the window like **:split**.The only difference is that if no filename is specified, a new window is started on a blank file. (Same as: **CTRL-Wn**, **:new**. See page 48.)

CTRL-W CTRL-O Make the current window the only one. (Same as: **CTRL-Wo**, **:on**, **:only**. See page 243.)

CTRL-W CTRL-P Move to the previous window. (Same as: **CTRL-Wp**. See pages 162-163 and 240.)

CTRL-W CTRL-Q Close a window. If this is the last window, exit *Vim*.The command fails if this is the last window for a modified file, unless the force (**!**) option is present. (Same as: **CTRL-W q**, **:q**, **:quit**. See pages 9, 46, 144, 202, and 242-243.)

[count] **CTRL-W CTRL-R** Rotate windows downward. (Same as: **CTRL-Wr**. See page 241.)

[count] **CTRL-W CTRL-S** Split the current window. (Make the new window *count* lines high.) (Same as: **CTRL-Ws**, **CTRL-WS**, **:sp**, **:split**. See pages 45, 47-48, 162, and 247.)

CTRL-W CTRL-T Move the top window. (Same as: **CTRL-Wt**. See page 240.)

CTRL-W CTRL-W Move to the next window. If there is no next window, move to the first one. If a *count* is specified, move to window number *count*. (Same as: **CTRL-Ww**. See pages 46 and 240.)

[count] **CTRL-W CTRL-X** Exchange the current window with the next one. If there is no next one, exchange the last window with the first. If a *count* is specified, exchange the current window with window number *count*. (Same as: **CTRL-Wx**. See page 242.)

CTRL-W CTRL-Z Close the preview window. Discard any changes if the force (**!**) option is present. (Same as: **CTRL-Wz**, **:pc**, **:pclose**. See page 276.)

[count] **CTRL-W +** Increase the size of the current window by *count*. (Default = 1.) (Same as: **:res +**, **:resize +**. See page 48.)

[count] **CTRL-W - (CTRL-W <dash>)** Decrease the size of the current window by *count*. (Default = 1.) (Same as: **res -**, **:resize -**. See page 48.)

CTRL-W= Make all windows the same size (or as close as possible). (See page 48.)

[count] **CTRL-W]** Split the current window and jump to the function whose name is under the cursor. If a *count* is specified, it is the height of the new window. (Same as: **CTRL-W CTRL-]**. See page 82.)

CTRL-W ^ Split the window and edit the alternate file. If a *count* is specified, split the window and edit the *count* file on the command line. (Same as: **CTRL-W CTRL-^**. See page 244.)

[count] **CTRL-W _** Set the current window to be *count* lines high. If no *count* is specified, make the window as big as possible. (Same as: **CTRL-W CTRL-_**, **:res**, **:resize**. See page 49.)

CTRL-W } Do a **:ptag** on the word under the cursor. (See page 277.)

CTRL-W b Move to the bottom window. (Same as: **CTRL-W CTRL-B**. See page 240.)

CTRL-W c Close the current window. (Same as: **:clo**, **:close**. See page 46.)

CTRL-W d Split the window and find the definition of the word under the cursor. If the definition cannot be found, do not split the window. (Same as: **CTRL-W CTRL-D**. See page 245.)

CTRL-W f Split the window and edit the file whose name is under the cursor. Looks for the file in the current directory, then all the directories specified by the path option. (Same as: **CTRL-W CTRL-F**. See page 245.)

CTRL-W g CTRL-] **:split** followed a **CTRL-]**. (Same as: **CTRL-W CTRL-G]**, **CTRL-Wg]**. See page 245.

CTRL-W g] **:split** followed a **CTRL-]**. (Same as: **CTRL-Wg CTRL]**, **CTRL-W CTRL-G]**. See page 245.)

CTRL-W g } Do a **:ptjump** on the word under the cursor. (Same as: **CTRL-W CTRL-G}**. See page 277.)

[count] **CTRL-W i** Split the window and search for the *count* occurrence of the word under the cursor. Start the search at the beginning of the file. (Same as: **CTRL-W CTRL-I**. See page 244.)

[count] **CTRL-W j** Move down a window. If a *count* is specified, move to window number *count*. (Same as: **CTRL-W CTRL-J**, **CTRL-W<Down>**. See page 46.)

[count] **CTRL-W k** Go up *count* windows. (Same as: **CTRL-W CTRL-K**, **CTRL-W<Up>**. See page 46.)

CTRL-W n Split the window like **:split**.The only difference is that if no filename is specified, a new window is started on a blank file. (Same as: **CTRL-W CTRL-N**, **:new**. See page 48.)

CTRL-W o Make the current window the only one. If ! is specified, modified files whose windows are closed will have their contents discarded. (Same as: **CTRL-W CTRL-O**, **:on**, **:only**. See page 243.)

CTRL-W p Move to the previous window. (Same as: **CTRL-W CTRL-P**. See pages 162-163 and 240.)

CTRL-W q Close a window. If this is the last window, exit *Vim*.The command fails if this is the last window for a modified file, unless the force (**!**) option is present. (Same as: **CTRL-W CTRL-Q**, **:q**, **:quit**. See pages 9, 46, 144, 202 and 242-243.)

[count] **CTRL-W r** Rotate windows downward. (Same as: **CTRL-W CTRL-R**, **CTRL-WR**. See page 241.)

[count] **CTRL-W R** Rotate windows upward. (Same as: **CTRL-W CTRL-R**, **CTRL-Wr**. See page 241.)

[count] **CTRL-W s** Split the current window. Make the new window *count* lines high (same as [count] **CTRL-W S**). (Same as: **CTRL-W CTRL-S**, **CTRL-WS**, **:sp**, **:split**. See pages 45-48, 162, 247.)

[count] **CTRL-W S** Split the current window. Make the new window *count* lines high (same as [count] **CTRL-W s**). (Same as: **CTRL-W CTRL-S**, **CTRL-Ws**, **:sp**, **:split**. See page 162.)

CTRL-W t Move the top window. (Same as: **CTRL-W CTRL-T**. See page 240.)

[count] **CTRL-W w** Move to the next window. If there is no next window, move to the first one. If a *count* is specified, move to window number *count*. (Same as: **CTRL-W CTRL-W**. See pages 46 and 240.)

[count] **CTRL-W W** Move to the previous window. If at the top window, go to the bottom one. If a *count* is specified, move to window number *count*. (See page 240.)

[count] **CTRL-W x** Exchange the current window with the next one. If there is no next one, exchange the last window with the first. If a *count* is specified, exchange the current window with window number *count*. (Same as: **CTRL-W CTRL-X**. See page 242.)

CTRL-W z Close the preview window. (Same as: **CTRL-W CTRL-Z**, **:pc**, **:pclose**. See page 276.)

[count] **CTRL-X** Subtract *count* to the number under the cursor. If no *count* is specified, decrement the number. (See pages 197-198 and 395.)

[count] **CTRL-Y** Move up *count* lines. (See pages 191-192.)

CTRL-Z Suspend the editor (Unix only). (See page 156.)

!{motion}{command} Filter the block of text represented by {motion} through the an external *{command}*command. (See pages 40, 85, 120, 164, and 166-167.)

[count] **!!**{command} Filter the current line (or *count* lines} through the an external command. (See page 40.)

[count] £ Search for the word under the cursor, backward. (Same as: **£**. See page 206.)

[count] $ Move the cursor to the end of the line. If a *count* is present, move to the end of the *count* line down from the current one. (Same as: **<End>**, **<kEnd>**. See pages 16 and 234.)

[count] % Jump to the line whose *count* percent of the way through the file. (See pages 73, 76-77, and 278.)

& Synonym for ":s//~/" – repeat last substitution. (See page 311.)

'{letter} Go to the line containing mark named {letter}. (See pages 37, 85, 161-162, 164, and 228.)

[count] **(** Move backward *count* sentences. (See page 121.)

[count] **)** Move forward *count* sentences. (See page 121.)

[count] * Search for the word under the cursor, forward. (See page 206.)

[count] + Move down *count* lines. (Default = 1.)Cursor is positioned on the first nonblank character on the line. (Same as: **<CR>**, **CTRL-M**. See page 187.)

[count] **,** Reverse the direction of the last single character and perform the search *count* times. (See page 187.)

[count] - Move up *count* lines. (Default = 1.) Cursor is positioned on the first non-blank character on the line. (See page 187.)

[count] / Repeat last search in the forward direction. (See pages 27-30, 32, 161, 203, and 227.)

[count] /{pattern} Search forward. (See pages 27-30, 32, 161, 203, and 227.)

[count] /{pattern}/{offset} Search forward, position the cursor at **{offset}** from the search pattern. (See page 208.)

[count] //{offset} Repeat last search in the forward direction with a new offset. (See page 208.)

[count] **;** Repeat the last single character search *count* times. (Default = 1.) (See page 187.)

[count] **<<** Shift *count* lines to the left. (See pages 69-70.)

<{motion} Shift lines from cursor to {motion} to the left. (See pages 69-70.)

[count] >> Shift *count* lines to the right. (See page 67.)

>{motion} Shift lines from cursor to {motion} to the right. (See pages 69-70.)

={motion} Filter {motion} lines through the external program given with the **'equalprg'** option. (See page 73.)

[count] **?** Repeat last search in the backward direction. (See pages 31-32.)

[count] **?**{pattern} Search backward. (See page 29.)

[count] **?**{pattern}**?**{offset} Search backward, position the cursor at {offset} from the search pattern. (See page 208.)

[count] **??**{offset} Repeat last search in the backward direction with a new {offset}. (See page 208.)

[count] @{character} Execute the macro in register {character}. (See page 24.)

["{register}] **[<MiddleMouse>** Put the **{register}** in the buffer like the p command, but adjust the text to fit the indent of the current line. (Same as: **[p**, **[P**, **]P**. See page 265.)

[count] **[CTRL-D** Find definition of the macro currently sitting under the cursor. Start the search from the beginning of the file. (See page 75.)

[count] **[CTRL-I** Search for the word under the cursor starting at the beginning of the file. (See pages 73-74 and 284.)

[count] ["{register}**<MiddleMouse>** Put the {register} in the buffer like the p command, but adjust the text to fit the indent of the current line. (See page 265.)

[£ Finds the previous unmatched **£if/£else/£endif**. (See page 279.)

[count] **[*** Move backward to the beginning of the *count* comment from the cursor. (Same as: **[/**. See page 280.)

[count] **[/** Same as: **[***. (See page 280.)

[count] **[(** Move backward to the *count* previous unmatched '(' in column 1. (See page 279.)

[count] **[)** Move backward to the *count* previous unmatched ')'. (See page 279.)

[count] **[[** Move backward *count* sections or to the previous { in column 1. (See pages 122 and 279.)

[count] **[]** Move *count* sections backwards or to the previous } in column 1. (See pages 122 amd 279.)

[count] **[}** Finds the *count* previous unmatched }. (See page 278.)

[count] **[d** List the definition of the macro. Start search at the current cursor locationbeginning of the file. (See pages 73, 75, and 284.)

[count] **[D** List all definitions of the macro whose name is under the cursor. Start the list with the next first definition in the file. (See pages 73 and 76.)

[f Deprecated. Use **gf** instead. (Same as: **gf**, **]f**. See page 281.)

[count] **[i** Display the *count* line that contains the keyword under the cursor.The search starts from the beginning of the file. (See page 284.)

[I List all lines in the current and included files that contain the word under the cursor. (See page 284.)

[m Search backward for the start of a method. (See page 279.)

[M Search backward for the end of a method. (See page 279.)

["{register}] **[p** Put the *{register}* in the buffer like the **P** command, but adjust the text to fit the indent of the current line. (Same as: **[<MiddleMouse>**, **[P**, **]P**. See page 265.)

["{register}] **[P** Put the {register} in the buffer like the P command, but adjust the text to fit the indent of the current line. (Same as: **[<MiddleMouse>**, **[p**, **[P**. See page 265.)

["{register}] **]<MiddleMouse>** Put the *{register}* in the buffer like the **p** command, but adjust the text to fit the indent of the current line. (Same as: **]p**. See page 265.)

[count] **]CTRL-D** Find definition of the macro currently sitting under the cursor. Start the search from the beginning current location.of the file. (See page 73.)

[count] **]CTRL-I** Search for the word under the cursor starting at the beginning of the filecurrent cursor location. (See pages 73-74 and 284.)

[count] **]#** Finds the next unmatched **#if/#else/#endif**. (See page 279.)

[count] **])** Move forward to the *count* next unmatched **)**. (See page 280.)

[count] **]/** -or-

[count] **]*** Move forward to the end of the *count* comment from the cursor. (See page 280.)

[count] **](** Move forward to the *count* next unmatched **(**. (See page 279.)

[count] **][** Move *count* sections forward or to the next } in column 1. (See pages 122 and 279.)

[count] **]]** Move *count* sections forward or to the next { in column 1. (See pages 122 and 279.)

[count] **]{** Finds the *count* previous unmatched {.(See page 278.)

[count] **]}** Finds the *count* previous unmatched }. (See page 278.)

[count] **]d** List the definition of the macro. Start search at the beginning of the current cursor position. (See pages 73, 75, and 284.)

[count] **]D** List all definitions of the macro whose name is under the cursor. Start the list with the first next definition. (See pages 73 and 76.)

]f Deprecated. Use **gf** instead. (Same as: gf, [f. See page 281.)

[count] **]i** Display the *count* line that contains the keyword under the cursor.The search starts from the beginning of the filecurrent cursor position. (See page 284.)

]I List all lines in the current and included files that contain the word under the cursor starting at the current location. (See page 284.)

]m Search forward for the start of a method. (See page 279.)

]M Search forward for the end of a method. (See page 279.)

["{register}] **]p** Put the *{register}* in the buffer like the P command, but adjust the text to fit the indent of the current line. (Same as: **]<MiddleMouse>**. See page 265.)

["{register}] **]P** Put the {*register*} in the buffer like the **P** command, but adjust the text to fit the indent of the current line. (Same as **]<MiddleMouse>**, **[p**, **[P**. See page 265.)

^ Move to the first nonblank character of the line. (See pages 16, 187, and 234.)

[count] _ Move to the first printing character of the *count*-1 line below the cursor. (See page 188.)

`{mark} Go to the mark named mark. Cursor is positioned exactly on the mark. (See page 37.)

'{mark} Go to the line containing mark. Position the cursor at the first non-blank character on the line. (See page 37.)

[count] { Move backward *count* paragraphs. (See page 121.)

[count] | Move to the column *count* on the current line. (See page 235.)

[count] } Move forward *count* paragraphs. (See page 121.)

~{motion} Change the case of the indicated characters. (This version of the command depends on the **'tildeop'** option being on. (The default is off.) To turn on the option, use the **:set tildeop** command. (See pages 24, 200-201, and 401.)

[count] ~ Change the case of *count* characters. (This version of the command depends on the **'tildeop'** option being off (the default).To turn off the option, use the :set notildeop command. (See pages 24, 200-201, and 401.)

[count] **£** Search for the word under the cursor, backward. (See page 202.)

0 (Zero) Move to the first character on the line. (See pages 16, 187, and 234.)

[count] **a**{text}**<Esc>** Insert text starting after the character under the cursor. If a *count* is specified, the text is inserted *count* times. (See page 9.)

[count] **A**{text}**<Esc>** Append the text on to the end of the line. (See page 197.)

[count] **b** Move backward *count* words. (Same as: **<S-Left>**. See page 16.)

[count] **B** Move *count* WORDS backward. (Same as: **<C-Left>**. See page 186.)

c{motion} Delete from the cursor to the location specified by the *{motion}* to enter insert mode. (See pages 21 and 161.)

[count] **C** Delete from the cursor to the end of the current line and *count*-1 more lines, and then enter insert mode. (See page 195.)

[count] **cc** Delete *count* entire lines (default = 1) and enter insert mode. (See page 22.)

["{register}] **d**{motion} Delete from the cursor location to where {*motion*} goes. (See pages 36, 161, and 195.)

[count] **D** Delete from the cursor to the end of the line. If a *count* is specified, delete an additional *count*-1 lines. (See pages 21 and 195.)

["{register}]count[count] **dd** Delete *count* lines. (See pages 10, 20, 36, and 224.)

[count] **e** Move *count* words forward, stop at the end of the word. (See page 184.)

[count] **E** Move *count* WORDS forward to the end of the WORD. (See page 186.)

[count] **f**{char} Search forward for character {char} on the current line. Stop on the character. (See pages 17 and 187.)

[count] **F**{char} Search backward for character {char} on the current line. Stop on the character. (See page 17.)

[count] **G** Go to the line *count*. If no line is specified, go to the last line in the file. (Same as: <C-End>. See pages 18, 40, 161, and 188.)

[count] **g**<Down> Move down one line on the screen. (Same as: **gj**. See page 235.)

g<End> Move to the rightmost character on the screen. (Same as: **g$**. See page 234.)

g<Home> Move to the leftmost character on the screen. (In other words, move to column 1.) (Same as: **g0**. See page 234.)

g<LeftMouse> Jump to the location of the tag whose name is under the cursor. (Same as: **<C-LeftMouse>**, **CTRL-]**. See page 12.)

[count] **g <RightMouse>** Jump to a previous entry in the tag stack. (Same as: **<C-RightMouse>**, **CTRL-T**. See page 12.)

[count] **]g<Up>** Move up lines in one the screen space. (Same as: **gk**. See page 235.)

g CTRL-] Do a :**tjump** on the word under the cursor. (See page 83.)

g CTRL-G Display detailed information about where you are in the file. (See page 156.)

g CTRL-H Start select block mode. (See page 258.)

[count] **g£** Search for the word under the cursor, backward. Unlike **£**, this finds partial words. (Same as: **g£**. See page 206.)

g$ Move to the rightmost character on the screen. (Same as: **g<End>**. See page 234.)

[count] **g*** Search for the word under the cursor, forward. Unlike *, this finds partial words. (See page 206.)

[count] **g??** Encrypt the lines using the rot13 encryption. (Same as: **g?g?**. See page 123.)

[count] **g?g?** Encrypt the lines using the rot13 encryption. (Same as: **g??**. See page 123.)

g£ Search for the word under the cursor, backwards. Unlike **£**, this finds partial words. (Same as: **g£**. See page 206.)

g?{motion} Encrypt the text from the current cursor location to where {motion} takes you using rot13 encryption. (See page 123.)

g] Do a :tselect on the word under the cursor. (See page 83.)

g^ Move to the leftmost printing character visible on the current line. (See page 234.)

g~{motion} Reverse the case of the text from the cursor to **{motion}**(same as **[count]g~g~**). (See pages 201 and 401.)

[count] **g~g~** -or-

[count] **g~~** Reverse the case of the entire line. If a *count* is specified, change the case of *count* lines. (Same as: **g~g~**. See page 201.)

g0 (zero) Move to the leftmost character on the screen. (In other words, move to column 1.) (Same as: **g<Home>**. See page 234.)

ga Print the ASCII value of the character under the cursor. (Same as: **gs**, **:as**, **:ascii**, **:sleep**. See page 155.)

gd Find the local definition of the variable under the cursor. (See pages 73-74.)

gD Find the global definition of the variable under the cursor. (See pages 73-74.)

[count] **ge** Move *count* words backward stopping on the end of the word. (See page 184.)

[count] **gE** Move *count* WORDS backward to the end of the WORD. (See page 186.)

gf Edit the file whose name is under the cursor. If the file is not in the current directory, search the directory list specified by the 'path' option. (Same as: **[f**, **]f**. See page 281.)

[count] **gg** Move to line *count*. Default is the first line. (Same as: **<C-Home>**. See page 156.)

gh Start select mode characterwise. (See page 258.)

gH Start select mode linewise. (See page 258.)

[count] **gI**{text}**<Esc>** Insert text in column 1, *count* times. (See page 197.)

[count] **gj** Move down one line on the screen. (Same as: **g<Down>**. See page 231.)

[count] **gJ** Join lines. No spaces are put between the assembled parts. If a *count* is specified, *count* lines are joined (minimum of two lines). (See page 198.)

[count] **] gk** Move up lines in inone the screen space. (Same as: **g<Up>**. See page 235.)

gm Move to the middle of the screen. (See page 234.)

[count] **go** Go to *count* byte of the file. (Same as: **:go**, **:goto**. See page 156.)

["{register}] **gp** Paste the text before the cursor, but do not move the cursor. (See page 220.)

["{register}] **gP** Paste the text after the cursor, but do not move the cursor. (See page 220.)

gq{motion} Format the text from the line the cursor is on to the line where {motion} takes you. (See pages 115, 117, 119, 269, and 271.)

gqq Format the current line. (Same as: **gqgq**. See page 115.)

gqgq Format the current line. (Same as: **gqq**. See page 115.)

[count] **gr**{character} Replace the virtual character under the cursor with {character}. (See pages 199-200.)

[count] **gR**{string}**<Esc>** Enter virtual replace mode until **<Esc>** is pressed. (See page 200.)

seconds gs Sleep for the specified number of seconds. (Same as: **:sl**, **:sleep**. See page 156.)

gu{motion} Uppercase of the text from the cursor to {motion}. (See page 201.)

gU{motion} Uppercase of the text from the cursor to {***motion***}. (See page 201.)

[count] **gugu** -or-

[count] **guu** Uppercase of the entire line. If a *count* is specified change the case of *count* lines. (See page 201.)

[count] **gUgU** -or-

[count] **gUU** Uppercase of the entire line. If a *count* is specified change the case of *count* lines. (See page 201.)

gv Repeat the last visual-mode selection. (See pages 252-253.)

gV Do not automatically reselect the selected text. (See page 260.)

[count] **h** Left. (Same as: **<BS>**, **<Left>**, **CTRL-H**, **CTRL-K**. See page 6.)

[count] **H** Move to the cursor to the top of the screen. If a *count* is specified, move to the *count* line from the top. (See page 188.)

[count] **i**{text}**<Esc>** Insert text starting before the character under the cursor. If a *count* is specified, the text is inserted *count* times. (Same as: **<Insert>**. See pages 5, 7, and 9.)

[count] **I**{text}**<Esc>** Insert the text at the beginning of the line. (See page 197.)

[count] **j** Down. (Same as: **<Down>**, **<NL>**, **CTRL-J**, **CTRL-N**. See pages 6 and 235.)

[count] **J** Join lines. Spaces are put between the assembled parts. If a *count* is specified, *count* lines are joined (minimum of 2 lines). (See pages 23, 116, and 198.)

[count] **k** Up. (Same as: **<Up>**, **CTRL-P**. See pages 6 and 13.)

[count] **K** Run the "man" command on the word under the cursor. If a *count* is specified, use *count* as the section number. On Microsoft Windows, by default, this command performs a :help on the word under the cursor. (See page 78.)

[count] **l** Right. (Same as: **<Right>**, **<Space>**. See page 6.)

[count] **L** Move the cursor to the bottom of the screen. If a *count* is specified, move to the *count* line from the bottom. (See page 188.)

m{letter} Mark the current text with the name {letter}. If {letter} is lowercase, the mark is local to the buffer being edited. In other words, just the location in the file is marked, and you have a different set of marks for each file. If an uppercase letter is specified, the mark is global. Both the file and the location within are marked. If you execute a **"go to mark(')"** command to jump to a global mark, you may switch files. (See pages 85, 161-162, 164, and 227.)

M Move to the cursor to the middle of the screen. (See page 188.)

[count] **n** Repeat last search. Search in the same direction. (See pages 31-32 and 161.)

[count] **N** Repeat last search. Search in the reverse direction. (See pages 31-32.)

[count] **o** Open a new line below the cursor and put the editor into insert mode. (See page 10.)

[count] **O** Open a new line above the cursor and put the editor into insert mode. (See page 11.)

["{register}] **p** Paste the test in the unnamed register (") after the cursor. (If the register contains complete lines, the text will be placed after the current line.) (See pages 36-37, 39, 160-161, 220, and 224.)

["""{register}] **P** Paste the text in the {register} before the cursor. If no {register} is specified, the unnamed register is used. (Same as: **<MiddleMouse>**. See pages 37, 162-163, and 220.)

q{character} Begin recording keys in register {character} (character is a–z). Stop recording with a **q** command. (See page 24.)

Q Enter ex mode. (See page 100.)

[count] **r**{char} Replace *count* characters with the given character. (See pages 23 and 199.)

[count] **R**{text}**<Esc>** Enter replace mode and replace each character in the file with a character from {*text*}. If a *count* is specified, repeat the command *count* times. (See page 199.)

[count] **s** Delete *count* characters and enter insert mode. (See page 196.)

[count] **S** Delete *count* lines and enter insert mode. (See page 196.)

[count] **t**{char} Search forward for character *{char}* on the current line. Stop one before the character. (See page 17.)

[count] **T**{char} Search backward for character *{char}* on the current line. Stop one after the character. (See page 17.)

u Undo the last change. (Same as: **<Undo>**. See page 8.)

U Undo all the changes on the last line edited. (A second **U** redoes the edits.) (See page 8.)

v Start visual character mode. (See pages 56 and 164.)

V Start visual line mode. (See pages 56, 86, 162, and 163.)

[count] **w** Move *count* words forward. (Same as: **<S-Right>**. See pages 16, 20, and 184.)

[count] **W** Move *count* WORDS forward. (Same as: **<C-Right>**. See page 186.)

["{register}] count[count] **x** Delete *count* characters. (Default = 1.) Deleted text goes into {register} or the unnamed register if no register specification is present. (Same as: **<Del>**. See pages 7, 13, 36-37, 160, and 196-197.)

["{register}]count[count] **X** Delete the characters before the cursor. (See pages 196-197.)

xp Exchange the character under the cursor with the next one. Useful for turning "teh" into "the". (See pages 37 and 160.)

["{register}] **y**{motion} Yank the text from the current location to {*motion*} into the register named {register}. Lowercase register specifications cause the register to be overwritten by the yanked text. Uppercase register specifications append to the contents of the register. (See pages 39 and 162.)

["{register}]count[count] **Y** -or-

["{register}] count[count] **yy** -or-

[count] ["{register}] **yy** Yank *count* lines into the register named {register}. Lowercase register specifications cause the register to be overwritten by the yanked text. Uppercase register specifications append to the contents of the register. (See pages 39 and 221-222.)

[count] **z<CR>** (Same as: **z<Enter>**. See page 193.)

[count] **z<Enter>** Position the line *count* at the top of the screen. If no *count* is specified, the current line is used. Cursor is positioned on the first nonblank character after this command. (Same as: **z:<CR>**. See page 193.)

[count] **z<Left>** Scroll the screen *count* characters to the right. (Same as: **zh**. See page 235.)

[count] **z<Right>** Scroll the screen *count* characters to the left. (Same as: **zl**. See page 235.)

[count] **z-** Position the line *count* at the bottom of the screen. If no *count* is specified, the current line is used. Cursor is positioned on the first nonblank character after this command. (See page 194.)

[count] **z.** Position the line *count* at the middle of the screen. If no *count* is specified, the current line is used. Cursor is positioned on the first nonblank character after this command. (See pages 194-195.)

[count] **zb** Position the line *count* at the bottom of the screen. If no *count* is specified, the current line is used. Cursor is positioned on the same column after this command. (See page 194.)

[count] **zh** Scroll the screen *count* character to the right. (Same as z<Left>. See page 235.)

[count] **zl** Scroll the screen *count* character to the left. (Same as z<Right>. See page 235.)

ZQ Do a **:quit!** command. (See page 202.)

[count] **zt** Position the line *count* at the top of the screen. If no *count* is specified, the current line is used. Cursor is positioned on the same column after this command. (See pages 193-194.)

[count] **zz** Position the line *count* at the middle of the screen. If no *count* is specified, the current line is used. Cursor is positioned on the same column after this command. (See pages 194-195.)

ZZ Write file and exit. (See pages 151 and 168.)

Motion Commands

[count] **a(** From with text enclosed in (), select the text up to and including the ().

[count] **a)** From with text enclosed in (), select the text up to and including the ().

[count] **ab** From with text enclosed in (), select the text up to and including the ().

[count] **a<** Select matching <> pair, include the "<>".

[count] **a>** Select matching <> pair, include the "<>".

[count] **a** Select matching pair, including the "".

[count] **a{** Select matching {} pair, including the "{}".

[count] **a}** Select matching {} pair, including the "{}".

[count] **aB** Select matching {} pair, including the "{}".

[count] **ap** Select a paragraph and the following space.

[count] **as** Select a sentence (and spaces after it).

[count] **aw** Select a word and the space after it. (Word is defined by the **'iskeyword'** option.)

[count] **aW** Select a word and the space after it. (Word is defined to be any series of printable characters.)

[count] **i(** Like **ab**, but the () characters are not selected.

[count] **i)** Like **ab**, but the () characters are not selected.

[count] **ib** Like **ab**, but the () characters are not selected.

[count] **i<** Select matching <> pair, excluding the "<>".

[count] **a>** Select matching <> pair, excluding the "<>".

[count] **i** Select matching pair, excluding the "".

[count] **i{** Select matching {} pair, excluding the "{}".

[count] **iB** Select matching {} pair, excluding the "{}".

[count] **ip** Select a paragraph only.

[count] **is** Select the sentence only. Do no select whitespace after a sentence.

[count] **iw** Select inner word (the word only). (Word is defined by the 'iskeyword' option.)

[count] **iW** Select inner word (the word only). (Word is defined to be any series of printable characters.)

D

Command-Mode Commands

:!{cmd} Execute shell command. (See page 319.)

:!! Repeat last **:!**{cmd}. (See page 319.)

:[range] **£** Print the lines with line numbers. (See pages 293 and 308.)

:[count] **&** Repeat the last **:substitute** command on the next count lines. (Default = 1.) (See page 311.)

:[range] **&** flags count Repeat the last substitution with a different range and flags. (See page 304.)

:[line] *****{register} Execute the contents of the register as an ex-mode command. (Same as: **:@**. See page 317.)

:[line] < {count} Shift lines left. (See page 317.)

:= Print line number. (See page 316.)

:[line] > {count} Shift lines right. (See page 317.)

:[line] **@**{register} Go to line and execute *{register}* as a command. (Same as: **:***. See page 317.)

:[line] **@**: Repeat the last command-mode command. (See page 317.)

:[line] **@@** Repeat the last **:@**{register} command. (See page 317.)

: [range]~ flags count Repeat the last substitution, but the last search string as the *{from}* pattern instead of the *{from}* from the last substitution. (See page 311.)

:[line] **a** Insert text after the specified line. (Default = current.) (Same as: **:append**. See page 307.)

:ab List all abbreviations. (Same as: **:abbreviate**. See pages 91-92, 95, 97, 166 and 296.)

:ab {lhs} {rhs} Define an abbreviation.When *{lhs}* is entered, put *{rhs}* in the text.

:abbreviate List all abbreviations. (Same as: **:ab**. See pages 91-92, 95, 97, 166, and 296.)

:abbreviate {lhs} {rhs} Define an abbreviation.When *{lhs}* is entered, put *{rhs}* in the text.

:abc -or-

:abclear Remove all abbreviations. (See page 296.)

:[count] **al** -or-

:[count] **all** Open a window for all the files being edited.When a [count] is specified, open up to [count] windows. (Note that the count can be specified after the command, such as **":all** [count].) (Same as: **:al**, **:sal**, **:sall**. See page 243.)

:{[priority} **ame** {menu-item} {command-string} -or-

:{[priority} **amenu** {menu-item} {command-string} Define a menu item that's that is valid for all modes . (Same as: **:am**, **:ame**. See page 334.) The following characters are automatically inserted for some modes:

Mode	Prefix Character Inserted	Meaning
Normal	(Nothing)	—N/A-
Visual	**<Esc>**	Exit visual mode
Insert	**CTRL-O**	Execute one normal command.
Command Line	**CTRL-C**	Exit command-line mode
Operator pending	<ESC>	End operator-Pending mode

:{priority} **an** {menu-item} {command-string} -or-

:{priority} **anoremenu** {menu-item} {command-string} Perform a **:amenu** command in which the {command-string} is not remapped. (Same as: **:an**. See page 338.)

:[line] **append** Insert text after the specified line. (Default = current). (Same as: **:a**. See page 307.)

:ar List the files being edited.The name of the current file is enclosed in square brackets ([]). (See page 222.)

:ar {file-list} Change the list of files to *{file-list}* and start editing the first one. (Same as: **:args**, **:n**, **:next**. See pages 43, 170, and 226.)

:args List the files being edited.The name of the current file is enclosed in square brackets ([]). (Same as: **:ar**, **:n**, **:next**. See pages 43, 170, and 226.)

:args {file-list} Change the list of files to *{file-list}* and start editing the first one. (See page 226.)

:argu {number} -or-

:argument {number} Edit the *{number}* file in the file list. (Same as: **:argu**. See pages 225 and 243.)

:as -or-

:ascii Print the number of the character under the cursor. (Same as: **ga**, **:as**. See page 155.)

:au List all the autocommands. (Same as: **:autocmd**. See pages 71, 97, 134, 135, 293, and 422.)

:au {group} {event} {pattern} Lists the autocommands that match the given specification. If "*" is used for the event, all events will match. (See pages 134-135.)

:au {group} {events} {file_pattern} **nested** {command} Define an autocommand to be executed when one of the *{events}* happens on any files that match *{pattern}*.The group parameters enables you to put this command in a named group for easier management.The nested flag allows for nested events. (See page 134-135.)

:au! {group} {event} {pattern} **nested** {command} Remove any matching autocommands and replace them with a new version. (See page 135.)

:aug {name} -or-

:augroup {name} Start an autocommand group.The group ends with a **:augroup** END statement. (Same as: **:aug**. See page 134.)

:aun {menu-item} -or-

:aunmenu {menu-item} Remove the menu item named {menu-item} that was defined with an **:amenu** command.The wildcard "*" will match all menu items. (Same as: **:aun**. See page 339.)

:autocmd List all the autocommands. (Same as: **:au**. See pages 71, 97, 134, 135, 293, and 422.)

:autocmd {group} {event} {pattern} Lists the autocommands that match the given specification. If **"*"** is used for the event, all events will match. (See pages 134-135.)

:autocmd {group} {events} {file_pattern} [**nested**] {command} Define an autocommand to be executed when one of the {events} happens on any files that match {pattern}.The group parameters enables you to put this command in a named group for easier management.The nested flag allows for nested events. (See pages 134-135.)

:autocmd! Delete all the autocommands. (See page 136.)

:autocmd! {group} {event} {pattern} Remove the specified autocommands. (See page 136.)

:autocmd! {group} {event} {pattern} [**nested**] {command} Remove any matching autocommands and replace them with a new version. (See page 136.)

:[count] **b[!]** Switch the current window to buffer number count. (If a count is not specified, the current buffer is used.) If **!** is specified, if the switch abandons a file, any changes might be discarded. (An alternative version of this command has count at the end—for example, **:buffer 5**.) (Same as: **:buffer**. See page 50.)

:b[!] {file-name} Switch the current window to the buffer containing {file-name}. If **!** is specified, if the switch abandons a file, any changes might be discarded. (See page 50.)

:[count] **ba** Open a window for each buffer. If a count is specified, open at most count windows. (Same as: **:ball**, **:sba**, **:sball**. See page 246.)

:bad [+line] {file} -or-

:badd [+line] {file} Add the file to the buffer list. If a +line is specified, the cursor will be positioned on that line when editing starts. (Same as: **:bad**. See page 246.)

:[count] **ball** Open a window for each buffer. If a count is specified, open at most count windows. (Same as: **:ba**, **:sba**, **:sball**. See page 246.)

:bd[[!]] {file} -or-

:bdelete[[!]] {file} -or-

:[[n] **]bd[[!]]** -or-

:[[n] **]bdelete[!]** -or-

:[[n,m] **]bd[[!]]** -or-

:[[n,m] **]bdelete[!]** -or-

:bd[[!]] [n] -or-

:bdelete[!] n Delete the specified buffer, but leave it on the buffer list. (Reclaims all the memory allocated to the buffer and closes all windows associated with.) If the override option (**!**) is specified, any changes made are discarded. If *{file}* is specified, the buffer for that file is deleted. A buffer number [n] or a range of buffer numbers [n,m] can be specified as well. (Same as: **:bd**. See page 246.)

:be {mode} -or-

:behave {mode} Sets the behavior of the mouse.The {mode} is either "xterm" for X Windows System–style mouse usage or "mswin" for Microsoft Windows–style usage. (Same as: **:be**. See page 108.)

:bl[!] -or-

:blast[!] Go to the last buffer in the list. (Same as: **:bl**. See page 52.)

:bm [count] -or-

:bmodified [count] Go to count-modified buffer. (Same as: **:bm**. See page 52.)

:[count] **bn[!]** -or-

:[count] **bnext[!]** -or-

:[count] **bNext** Go to the next buffer. If **!** is specified, if the switch abandons a file, any changes might be discarded. If a count is specified, go to the count next buffer. (Same as: **:bN**, **:bp**, **:bprevious**. See page 51.)

:[count] **bp** -or-

:[count] **bprevious** Go to previous buffer. If a count is specified, go to the count previous buffer. (Same as: **:bN**, **:bNext**, **:bp**. See page 51.)

:br[!] Go to the first buffer in the list. (Same as: **:brewind**. See page 52.)

:brea -or-

:break Break out of a loop. (Same as: **:brea**. See page 356.)

:brewind[!] Go to the first buffer in the list. (Same as: **:br**. See page 52.)

:bro {command} Open a file browser window and then run *{command}* on the chosen file. (Same as: **:browse**. See page 339.)

:bro set Enter an option browsing window that enables you to view and set all the options. (Same as: **:browse set**, **:options**. See page 342.)

:browse {command} Open a file browser window and then run *{command}* on the chosen file. (Same as: **:bro**. See page 339.)

:browse set Enter an option browsing window that enables you to view and set all the options. (Same as: **:bro set**, **:options**. See page 342.)

:[count] **buffer[!]** Switch the current window to buffer number count. (If a count is not specified, the current buffer is used.) If **!** is specified, if the switch abandons a file, any changes might be discarded. (An alternative version of this command has count at the end—for example, **:buffer 5**.) (Same as: **:b**. See page 50.)

:buffer[!] {file-name} Switch the current window to the buffer containing *{file-name}*. If **!** is specified, if the switch abandons a file, any changes might be discarded. (See page 50.)

:buffers List all the specified buffers. (Same as: **:bu**, **:files**, **:ls**. See pages 49-50.)

:bun[[!]] {file} -or-

:bunload[[!]] {file} -or-

:[n]**bun[[!]]** -or-

:[n]**bunload[[!]]** -or-

:[n,m]**bun[[!]]** -or

:[n,m]**bunload[[!]]** -or-

:bun[[!]] [n] -or-

:bunload[!] n Unload the specified buffer. If the override option is specified, if there are any changes, discard them. (Same as: **:bun**. See page 246.)

:[range] **c** Delete the specified lines, and then do a **:insert**. (Same as: **:change**. See page 317.)

:ca {lhs} {rhs} -or-

:cabbrrev {lhs} {rhs} Define an abbreviation for command-mode only. (Same as: **:ca**. See page 296.)

:cabc -or-

:cabclear Remove all for command mode. (Same as: **:cabc**. See page 296.)

:[range] **cal** {name}(argument list) -or-

:[range] **call** {name}(argument list) Call a function. (Same as: **:cal**. See page 134, 358, and 361.)

:cc[!] number Display error number. If the number is omitted, display the current error. Position the cursor on the line that caused it. (See page 88.)

:cd {path} Change the directory to the specified path. If path is "-", change the previous path. If no path is specified, on UNIX go to the home directory. On Microsoft Windows, print the current directory. (See page 314.)

:[range] **ce** [width] -or-

:[range] **center** [width] Center the specified lines. If the width of a line is not specified, use the value of the **'textwidth'**. (If **'textwidth'** is 0, 80 is used.) (Same as: **:ce**. See page 115.)

:cf[!] [errorfile] -or-

:cfile[!] [errorfile] Read an error list from file. (Default = the file specified by the errorfile option.) Go to the first error. If the override option is specified and a file switch is made, any unsaved changes might be lost. (Same as: **:cf**. See page 89.)

:[range] **ch** (Same as: **:chdir**. See page 314.)

:[range] **change** Delete the specified lines, and then do an **:insert**. (Same as: **:c**. See page 317.)

:chd [path] -or-

:chdir [path] Change the directory to the specified path. If path is **"-"**, change the previous path. If no path specified, on UNIX go to the home directory. On Microsoft Windows, print the current directory. (Same as: **:chd**, **:chdir**. See page 314.)

:che[!] -or-

:checkpath[!] Check all the **£include** directives and make sure that all the files listed can be found. If the override option (**!**) is present, list all the files. If this option is not present, only the missing files are listed. (Same as: **:che**. See page 282.)

:cl[!] [from], [to] List out the specified error messages. If the override option is present, list out all the errors. (Same as: **:clist**. See page 88.)

:cla [number] -or-

:clast [number] Go the last error in the list. If a number is specified, display that error. (Same as: **:cla**. See page 88.)

:clist[!] [from], [to] List out the specified error messages. If the override option is present, list out all the errors. (Same as: **:cl**. See page 88.)

:clo[!] -or-

:close[!] Close a window. If this is the last window, exit *Vim*.The command fails if this is the last window for a modified file, unless the force (**!**) option is present. (Same as: **CTRL-Wc**, **:clo**. See page 46.)

:cm Listing all the mappings for command-line mode maps. (Same as: **:cmap**. See page 298.)

:cm {lhs} List the command-line mapping of *{lhs}*. (See page 298.)

:cm {lhs} {rhs} Define a keyboard mapping for command-line mode. (See page 298.)

:cmap Listing all the command-line mode mappings. (Same as: **:cm**. See page 298.)

:cmap {lhs} List the command-line mode mapping of *{lsh}*. (See page 298.)

:cmap {lhs} {rhs} Define a keyboard mapping for command-line mode. (See page 298.)

:cmapc -or-

:cmapclear Clear all the command-mode mappings. (Same as: **:cmapc**. See page 301.)

:[priority] **cme** {menu-item} {command-string} -or-

:[priority] **cmenu** {menu-item} {command-string} Define a menu item that is available for command-line mode only.The priority determines its placement in a menu. Higher numbers come first.The name of the menu item is *{menu-item}*, and when the command is selected, the command *{command-string}* is executed. (Same as: **:cme**. See page 334.)

:[count] **cn[!]** Go to the count next error. (Same as: **:cnext**. See pages 87-88 and 170.)

:[count] **cN[!]** Go the previous error in the error list. (Same as: **:cN**. See page 88.)

:cnew [count] -or-

:cnewer [count] Go to the count newer error list. (Same as: **:cnew**. See page 285.)

:[count] **cnext[!]** Go to the count next error. (Same as: **:cn**. See pages 87-88 and 170.)

:[count] **cNext[!]** Go the previous error in the error list. (Same as: **:cN**. See page 88.)

:[count] **cnf[!]** -or-

:[count] **cnfile[!]** Go the first error in the next file. If the override option **(!)** is present, if there are any unsaved changes, they will be lost. (Same as: **:cnf**. See pages 88 and 90.)

:cno {lhs} {rhs} Same as **:cmap**, but does not allow remapping of the {rhs}. (Same as: **:cnoremap**. See page 301.)

:cnorea {lhs} {rhs} -or-

:cnoreabbr {lhs} {rhs} Do a **:noreabbrev** that works in command-mode only. (Same as: **:cnorea**. See page 301.)

:[priority] **cnorem** {menu-item} {command-string} Like **:cmenu**, except the {command-string} is not remapped. (Same as: **:cnoremenu**. See page 338.)

:cnoremap {lhs} {rhs} Same as **:cmap**, but does not allow remapping of the {rhs}. (Same as: **:cno**. See page 301.)

:[priority] **cnoremenu** {menu-item} {command-string} Like **:cmenu**, except the {command-string} is not remapped. (Same as: **:cnorem**. See page 338.)

:[range] **co** {address} Copy the range of lines below *{address}*. (Same as: **:copy**, **:t**. See page 306.)

:col [count] -or-

:colder [count] Go to the count older error list. (Same as: **:col**. See page 284.)

:com List the user-defined commands. (Same as: **:command**. See pages 360-361.)

:com {definition} Define a user-defined command. (See pages 360-361.)

:comc -or-

:comclear Clear all user-defined commands. (Same as: **:comc**. See page 360.)

:command List the user-defined commands. (See page 354.)

:command {definition} Define a user-defined command. (Same as: **:com**. See pages 360-361.)

:con {command} (Same as: **:continue**. See page 356.)

:conf {command} Execute the *{command}*. If this command would result in the loss of data, display a dialog box to confirm the command. (Same as: **:confirm**. See page 341.)

:confirm {command} Execute the *{command}*. If this command would result in the loss of data, display a dialog box to confirm the command. (Same as: **:conf**. See page 341.)

:con -or-

:continue Start a loop over. (Same as: **:con**. See page 356.)

:[range] **copy** {address} Copy the range of lines below *{address}*. (Same as: **:co**, **:t**. See page 306.)

:[count] **cp[!]** -or-

:[count] **cprevious[!]** Go to the count previous error. (Same as: **:cp**, **:cN**, **:cNext**. See pages 88 and 170.)

:cq -or-

:cquit Exit *Vim* with an error code. (This is useful in integrating *Vim* into an IDE.) (Same as: **:cq**. See page 89.)

:cr[!] number -or-

:crewind[!] number Go the first error in the list. If a number is specified, display that error. (Same as: **:cr**. See page 88.)

:cs {arguments} -or-

:cscope {argument} Handle various activities associated with the CScope program. (Same as: **:cs**. See page 172.)

:cst {procedure} -or-

:cstag {procedure} Go to the tag in the CScope database named *{procedure}*. (Same as: **:cst**. See page 172.)

:cu {lhs} (Same as: **:cunmap**. See page 301.)

:cuna {lhs} -or-

:cunabbreviate {lhs} Remove the command-line mode abbreviation. (Same as: **:cuna**. See page 296.)

:cunm Same as: **:cunmenu**. (See page 339.)

:cunmap {lhs} Remove a command-mode mapping. (Same as: **:cu**. See page 301.)

:cunmenu {menu-item} Remove the command-mode menu item named {menu-item}.The wildcard "*" will match all menu items. (Same as: **:cunm**. See page 339.)

:[range] **d** register [count] Delete text. (Same as: **:delete**. See page 303.)

:delc {command} -or-

:delcommand {command} Delete a user-defined command. (Same as: **:delc**. See page 360.)

:[range] delete register [count] Delete text. (Same as: **:d**. See page 303.)

:delf {name} -or-

:delfunction {name} Delete the function named *{name}*. (Same as: **:delf**. See page 360.)

:di list Edit the last file in the list. (Same as: **:display**. See page 222.)

:dig List all the digraph definitions. (Same as: **:digraphs**. See page 25.)

:dig {character1}{character2} {number} Define a digraph.When **CTRL-K**{character1}{character2} is pressed, inset character whose number is *{number}*. (See page 200.)

:digraphs List all the digraph definitions. (Same as: **:dig**. See page 25.)

:digraphs {character1}{character2} {number} Define a digraph.When **CTRL-K**{character1}{character2} is pressed, insert character whose number is *{number}*. (See page 200.)

:display [arg] Same as **:registers**, **:di**. (See page 222.)

:[range] **dj** count[count] /{pattern}/ -or-

:[range] **djump** count[count] /{pattern}/ Search the range (default = whole file) for the definition of the macro named *{pattern}* and jump to it. If a count is specified, jump to the count definition. If the pattern is enclosed in slashes (/), it is a regular expression; otherwise, it is the full name of the macro. (Same as: **:dj**. See page 314.)

:[range] **dl** /{pattern}/ -or-

:[range] **dlist** /{pattern}/ List the all the definitions of the macro named *{pattern}* in the range. (Default = the whole file.) If the pattern is enclosed in slashes (/),it is a regular expression; otherwise, it is the full name of the macro. (Same as: **:dl**. See page 314.)

:do {group} {event} [file_name] Execute a set of autocommands pretending that *{event}* has just happened. If a group is specified, execute only the commands for that group. If a filename is given, pretend that the filename is file_name rather than the current file during the execution of this command. (Same as: **:doautocmd**. See pages 135 and 137.)

:doautoa {group} {event} [file_name] -or-

:doautoall {group} {event} [file_name] Like **:doautocmd**, but repeated for every buffer. (Same as: **:doautoa**. See page 135.)

:doautocmd {group} {event} [file_name] Execute a set of autocommands, pretending that *{event}* has just happened. If a group is specified, execute only the commands for that group. If a filename is given, pretend that the filename is file_name rather than the current file during the execution of this command. (Same as: **:do**. See pages 135 and 137.)

:[range] **ds** /{pattern}/ -or-

:[range] **dsearch** /{pattern}/ List the first definition of the macro named *{pattern}* in the *range*. (Default = the whole file.) If the pattern is enclosed in slashes (/), it is a regular expression; otherwise, it is the full name of the macro. (Same as: **:ds**. See page 314.)

:[range] **dsp** count[count] /{pattern}/ -or-

:[range] **dsplit** count[count] /{pattern}/ Do a **:split** and a **:djump**. (Same as: **:dsp**. See page 314.)

:e [+cmd] [file] Close the current file and start editing the named file. If no file is specified, re-edit the current file. If *+cmd* is specified, execute it as the first editing command. (Same as: **:edit**. See page 41.)

:ec {arguments} -or-

:echo {arguments} Print the arguments. (Same as: **:ec**. See pages 354-355.)

:echoh {name} -or-

:echohl {name} Change the color of future echoes to be in the color of highlight group *{name}*. (Same as: **:echoh**. See page 355.)

:echon {arguments} Echo the arguments without a newline. (See page 355.)

:edit [+cmd] [file] Close the current file and start editing the named file. If no file is specified, re-edit the current file. If *+cmd* is specified, execute it as the first editing command. (Same as: **:e**. See page 41.)

:el -or-

:else Reverse the condition of an **:if**. (Same as: **:el**. See page 356.)

:elsei -or-

:elseif A combination of **:else** and **:if**. (Same as: **:elsei**. See page 356.)

:em (Same as: **:ememu**. See page 338.)

:ememu {menu-item} Execute the given {menu-item} as if the user had selected it. (Same as: **:em**. See page 338.)

:en End an **:if** statement. (Same as: **:endif**. See page 356.)

:endf End a function. (Same as: **:endfunction**. See pages 134 and 357.)

:endfunction End a function. (Same as: **:endf**. See pages 134 and 357.)

:endif End an :if statement. (Same as: **:en**. See page 356.)

:endw -or-

:endwhile End a loop. (Same as: **:endw**. See page 356.)

:ex[!] +command filename Enter ex mode. If a filename is specified, edit that file; otherwise, use the current file.The *+command* argument is a single command that will be executed before any editing begins. If the override option (**!**) is specified, switching files will discard any changes that have been made. (See page 100.)

:[range] **ex[!]** file If the buffer has been modified, write the file and exit. If a range is specified, write only the specified lines. If a file is specified, write the data to that file.When the override option (**!**) is present, attempt to overwrite existing files or read-only files. (See page 322.)

:exe {string} -or-

:execute {string} Execute a string as a command. (Same as: **:exe**. See page 357.)

:exi -or-

:exit -or-

:[range] **exit[!]** [file] If the buffer has been modified, write the file and exit. If a range is specified, write only the specified lines. If a file is specified, write the data to that file.When the override option (**!**) is present, attempt to overwrite existing files or read-only files. (Same as: **:exi**. See page 322.)

:f Print the current filename. If a file is specified, set the name of the current file to file. (Same as: **:file**, **CTRL-G**. See pages 135 and 315.)

:file [file] Print the current filename. If a file is specified, set the name of the current file to file. (Same as: **:f**, **CTRL-G**. See pages 135 and 315.)

:files List all the specified buffers. (Same as: **:buffers**, **:ls**. See pages 49-50.)

:filet {on|off} -or-

:filetype {on|off} Tell *Vim* to turn on or off the file type detection logic. (Same as: **:filet**. See page 71.)

:fin[!] {+command} {file} -or-

:find[!] {+command} {file} Like :**vi**, but searches for the file in the directories specified by the path option. (Same as: **:fin**. See page 281.)

:fix -or-

:fixdel Make the Delete key do the right thing on UNIX systems. (Same as: **:fix**. See page 94.)

:fu List all functions. (Same as: **:function**. See pages 134, 357, and 359.)

:fu {name} List the contents of function *{name}*. (See page 359.)

:fu {function definition} Start a function definition. (See page 359.)

:function List all functions. (Same as: **:fu**. See pages 134, 357, and 359.)

:function {name} List the contents of function *{name}*. (See page 359.)

:function {function definition} Start a function definition. (See page 359.)

: [range] **g** /{pattern}/ {command} Perform *{command}* on all lines that have *{pattern}* in them in the given range. (Same as: **:global**. See page 311.)

: [range] **g!** /{pattern}/ {command} Perform *{command}* on all lines that do not have *{pattern}* in them in the given range. (See page 311.)

: [range] **global** /{pattern}/ {command} Perform *{command}* on all lines that have *{pattern}* in them in the given range. (Same as: **:g**. See page 311.)

: [range] **global!** /{pattern}/ {command} Perform *{command}* on all lines that do not have *{pattern}* in them in the given range. (See page 311.)

:go {[count] } -or-

:goto {[count] } Go to count byte of the file. (Same as: **:go**, **go**. See page 156.)

:gr {arguments} -or-

:grep {arguments} Run the gGrep program with the given *{arguments}* and capture the output so that the **:cc**, **:cnext**, and other commands will work on it. (Like **:make**, but with gGrep rather than mMake.) (Same as: **:gr**. See pages 89, 170, 225, 284, and 289.)

:gu [+command] [-f|-b] [files...] -or-

:gui [+command] [-f|-b] [files...] (Same as: **:gu**. See page 323.)

:gv [+command] [-f|-b] [files...] -or-

:gvim [+command] [-f|-b] [files...] Start GUI mode. If a +command is specified, execute that after loading the files. If the -b flag is specified, execute the command in the background (the default).The -f flag tells *Vim* to run in the foreground. If a list of files is specified, they will be edited; otherwise, the current file is edited. (Same as: **:gv**. See page 323.)

:h [topic] -or-

:help [topic] Display help on the given topic. If no topic is specified, display general help. (Same as: **<F1>**, **<Help>**, **:h**. See pages 11, 13, 50, 157, and 401.)

:helpf –or-

:helpfind Open a dialog box that enables you to type in a help subject. (Same as: **:helpf**. See page 341.)

:hi List all highlight groups. (Same as: **:highlight**. See pages 290 and 355.)

:hi {options} Customize the syntax coloration. (See page 290.)

:hi link {new-group} {old-group} Highlight the *{new-group}* the same as *{old-group}*. (See page 414.)

:hid -or-

:hide Hide the current buffer. (Same as: **:hid**. See page 49.)

:highlight List all highlight groups.

:highlight link {new-group} {old-group} Highlight the *{new-group}* the same as {old-group}. (See page 408.)

:highlight {options} Customize the syntax coloration. (Same as: **:hi**. See pages 290 and 355.)

:his {code} [first] ,[last] -or-

:history {code} [first] ,[last] Print the last few commands or search strings (depending on the code).The code parameter defaults to **"cmd"** for command-mode command history.The first parameter defaults to the first entry in the list and last defaults to the last. (Same as: **:his**. See page 320.)

:[line] **i** Start inserting text before line. Insert ends with a line consisting of just **".**". (Same as: **:insert**. See page 308.)

:ia {lhs} {rhs} -or-

:iabbrev {lhs} {rhs} Define an abbreviation for insert mode only. (Same as: **:ia**. See page 296.)

:iabc -or-

:iabclear Remove all for insert mode. (Same as: **:iabc**. See page 296.)

:if {expression} Start a conditional statement. (See page 356.)

:[range] **ij** [count] /{pattern}/ -or-

:[range] **ijump** [count] /{pattern}/ Search the range (default = whole file) for the *{pattern}* and jump to it. If a count is specified, jump to the count occurrence. If the pattern is enclosed in slashes (/), it is a regular expression; otherwise, it is just a string. (Same as: **:ij**. See page 312.)

:[range] **il** /{pattern}/ -or-

:[range] **ilist** /{pattern}/ List all the occurrences *{pattern}* in the range. (Default = the whole file.) If the pattern is enclosed in slashes (/), it is a regular expression; otherwise, it is a string. (Same as: **:il**. See page 312.)

:im List all the insert-mode mappings. (Same as: **:imap**. See page 298.)

:im {lhs} List the insert-mode mapping of *{lhs}*. (See page 298.)

:im {lhs} {rhs} Define a keyboard mapping for insert mode. (See page 298.)

:imap List all the insert-mode mappings. (Same as: **:im**. See page 298.)

:imap {lhs} List the insert-mode mapping of *{lhs}*. (See page 298.)

:imap {lhs} {rhs} Define a keyboard mapping for insert mode. (See page 298.)

:imapc -or-

:imapclear Clear all the insert-mode mappings. (Same as: **:imapc**. See page 301.)

:[priority] **ime** {menu-item} {command-string} -or-

:[priority] **imenu** {menu-item} {command-string} Define a menu item that is available for insert mode only.The priority determines its placement in a menu. Higher numbers come first.The name of the menu item is *{menu-item}*, and when the command is selected, the command {command-string} is executed. (Same as: **:ime**. See page 334.)

:[line] **in** Start inserting text before line. Insert ends with a line consisting of just **".**". (Same as: **:inoremap**. See page 301.)

:inorea {lhs} {rhs} -or-

:inoreabbrev {lhs} {rhs} Do a **:noreabbrev** that works in insert mode only. (Same as: **:inorea**. See page 296.)

:inorem {lhs} {rhs} Same as: **:inoremenu**. (See page 338.)

:inoremap {lhs} {rhs} Same as **:imap**, but does not allow remapping of the *{*rhs}. (Same as: **:in**. See page 301.)

:[priority] **inoremenu** {menu-item} {command-string} Like **:imenu**, except the {command-string} is not remapped. (Same as: **:inorem**. See page 338.)

:[line] **insert** Start inserting text before line. Insert ends with a line consisting of just **".**". (Same as: **:i**. See page 308.)

:int -or-

:intro Display the introductory screen. (Same as: **:int**. See page 157.)

:range[range] **is** /{pattern}/ -or-

:range[range] **isearch** /{pattern}/ List the first occurrence *{pattern}* in the range. (Default = the whole file.) If the pattern is enclosed in slashes (/), it is a regular expression; otherwise, it is a string. (Same as: **:is**. See page 313.)

:[range] **isp** count[count] /{pattern}/ -or-

:[range] **isplit** count[count] /{pattern}/ Remove an insert-mode mapping. (Same as: **:isp**. See page 313.)

:iu {lhs} Remove an insert-mode mapping. (Same as: **:iunmap**. See page 301.)

:iuna {lhs} -or-

:iunabbreviate {lhs} Remove the insert line-mode abbreviation. (Same as: **:iuna**. See page 296.)

:iunm {lhs} -or-

:iunmap {lhs} Remove an insert-mode mapping. (Same as: **:iunm**. See page 301.)

:iunm {menu-item} -or-

:iunmenu {menu-item} Remove the insert-mode menu item named *{menu-item}*.The wildcard "*" will match all menu items. (Same as: **:iunm**. See page 339.)

:[range] **j[!]** -or-

:[range] **join[!]** Join the lines in range into one line. Spaces are used to separate the parts unless the **!** is specified. (Same as: **:j**. See page 318.)

:ju -or-

:jumps List out the jump list. (Same as: **:ju**. See page 188.)

:[line] **k**{letter} Place mark *{letter}* on the indicated line. (Same as: **:mar**, **:mark**. See page 318.)

:[range] **l** [count] Like **:print**, but assumes that the **'list'** option is on. (Same as: **:l**. See page 308.)

:la [+command] -or-

:last [+command] Edit the last file in the list. (Same as: **:la**. See page 44.)

:[range] **le** [margin] -or-

:[range] **left** [margin] Left justify the text putting each line margin characters from the left margin. (Default = 0.) (Same as: **:le**. See page 116.)

:let {variable} = {expression} Assign a *{variable}* a value. (See page 349.)

:[range] **list** [count] Like **:print**, but assumes that the **'list'** option is on. (Same as: **:l**. See page 308.)

:ls List all the buffers. (Same as: **:buffers**, **:files**. See pages 49-50.)

:[range] **m** {address} Move the range of lines from their current location to below *{address}*. (Same as: **:move**. See pages 161 and 306.)

:[line] **ma**{letter} Mark the current line with mark *{letter}*. (Same as: **:k**, **:mar**, **:mark**. See page 318.)

:mak {arguments} -or-

:make {arguments} Run the external Mmake program, giving it the arguments indicated. Capture the output in a file so that error-finding commands such as **:cc** and **:cnext** can be used. (Same as: **:mak**. See pages 87, 284, and 288.)

:map List all the mappings. Note: Only **:map** and **:map!** list the mappings for all modes.The other mode-dependent versions of these commands list the mapping for their modes only. (See pages 93, 95, 134, 156, 293, 297, and 394.)

:map {lhs} List the mapping of *{lhs}*. (See page 297.)

:map {lhs} {rhs} Define a keyboard mapping.When the *{lhs}* is typed in normal mode, pretend that *{rhs}* was typed. (See page 297.)

:map[!] List all mappings for insert and command line notes. (See page 298.)

:{mode}**mapc** Clear all the mappings. (Same as: **:mapclear**. See page 299.)

:{mode}**mapclear** Clear all the mappings. (Same as: **:mapc**. See page 299.)

:mapclear[!] See page 301.

:mar -or-

:[line] **mark**{letter} Mark the given line with mark *{letter}*. (Same as: **:k**, **:mar**. See page 38.)

:marks List all the marks. (See page 38.)

:marks {chars} List the marks specified by the character list: *{chars}*. (See page 36.)

:[priority][mode] **me** {menu-item} {command-string} -or-

:[priority][mode] **menu** {menu-item} {command-string} Define a menu item.The priority determines its placement in a menu. Higher numbers come first.The mode parameter defines which *Vim* mode the item works in.The name of the menu item is *{menu-item}*, and when the command is selected, the command *{command-string}* is executed. (Same as: **:me**. See pages 333 and 338.)

:mes -or-

:messages View previous messages. (Same as: **:mes**. See page 321.)

:mk[!] {[file]} -or-

:mkexrc[!] {[file]} Like **:mkvimrc**, except the file defaults to .exrc.This command has been superseded by the **:mkvimrc** command. (Same as: **:mk**. See page 96.)

:mks[!] [file]} -or-

:mksession[!] [file] } Create a session file and save the current settings. If **!** is specified, overwrite any existing session file. (Same as: **:mks**. See page 350.)

:mkv[!] {file} -or-

:mkvimrc[!] {file} Write out setting to *{file}* in a manner suitable for including in a *vim*rc file. In fact, if you do not specify *{file}*, it defaults to *vim*rc. If the file exists, it will be overwritten if the override option (**!**) is used. (Same as: **:mkv**. See page 95.)

:[range] **mo** {address} Move the range of lines from their current location to below *{address}*. (See page 307.)

:mod {mode} -or-

:mode {mode} Set the screen mode for an MS-DOS editing session. (Same as: **:mod**. See page 347.)

:[range] **move** {address} Move the range of lines from their current location to below {address}. (Same as: **:m**. See pages 161 and 306.)

:[count] **n** {+cmd} {file-list} When editing multiple files, go to the next one. If count is specified, go to the count next file. (Same as: **:args**, **:next**. See pages 42, 170, 226-228.)

:[count] **N** {+cmd} {file-list} When editing multiple files, go to the previous one. If a count is specified, go to the count previous file. (Same as: **:Next**. See page 43.)

:new[!] [+command] [file-name] Split the window like **:split**.The only difference is that if no filename is specified, a new window is started on a blank file. (Same as: **CTRL-W CTRL-N**, **CTRL-Wn**. See page 48.)

:[count] **next** {[+cmd]} [{file-list]} When editing multiple files, go to the next one. If count is specified, go to the count next file. (Same as: **:args**, **:n**. See pages 42, 170, and 226-228.)

:[count] **Next** [{+cmd]} {[file-list]} When editing multiple files, go to the previous one. If count is specified, go to the count previous file. (Same as: **:N**. See page 43.)

:nm Listing all the mappings for normal-mode maps. (Same as: **:nmap**. See page 298.)

:nm {lhs} List the normal mapping of *{lhs}*.

:nm {lhs} {rhs} Define a keyboard mapping for normal mode.

:nmap Listing all the normal-mode mappings. (Same as: **:nm**. See page 298.)

:nmap {lhs} List the normal-mode mapping of *{lhs}*. (See page 298.)

:nmap {lhs} {rhs} Define a keyboard mapping for normal mode. (See page 298.)

:nmapc -or-

:nmapclear Clear all the normal mappings. (Same as: **:nmapc**. See page 301.)

:[priority]**nme** {menu-item} {command-string} -or-

:[priority]**nmenu** {menu-item} {command-string} Define a menu item that is available for normal mode only.The priority determines its placement in a menu. Higher numbers come first.The name of the menu item is *{menu-item}*, and when the command is selected, the command *{command-string}* is executed. (Same as: **:nme**. See page 334.)

:nn {lhs} {rhs} -or-

:nnoremap {lhs} {rhs} Same as **:nmap**, but does not allow remapping of the *{rhs}*. (Same as: **:nn**. See page 301.)

:[priority] **nnoreme** {menu-item} {command-string} -or-

:[priority] **nnoremenu** {menu-item} {command-string} Like **:nmenu**, but the {command-string} is not remapped. (Same as: **:nnorem**. See page 301.)

:no {lhs} {rhs} Same as **:map**, but does not allow remapping of the *{rhs}*. (Same as: **:noremap**. See pages 298 and 301.)

:noh -or-

:nohlsearch Turn off the search highlighting. (It will be turned on by the next search.To turn it off permanently, use the **:set nohisearch** command.) (Same as: **:noh**. See page 29.)

:nor {lhs} {rhs} -or-

:noreabbrev {lhs} {rhs} Define an abbreviation, but do not allow remapping of the right side. (Same as: **:nor**. See page 301.)

:norem {lhs} {rhs} Same as: **:noremenu**. (See page 338.)

:noremap {lhs} {rhs} Same as **:map**, but does not allow remapping of the *{rhs}*. (Same as: **no**. See pages 298 and 301.)

:nremap[!] See page 301.

:[priority][mode] **norem** {menu-item} {command-string} -or-

:[priority][mode] **noremenu** {menu-item} {command-string} Define a menu item like defined with **:menu**, but do not allow remapping of the *{command-string}*. (Same as: **:norem**. See page 338.)

:norm[!] {commands} -or-

:normal[!] {commands} Execute the commands in normal mode. (Same as: **:norm**. See page 322.)

:[range] **nu** -or-

:[range] **number** Print the lines with line numbers. (Same as: **:nu**. See pages 293-294.)

:nun {lhs} -or-

:nunmap {lhs} Remove a normal mapping. (Same as: **:nun**. See page 301.)

:nunme {menu-item} -or-

:nunmenu {menu-item} Remove the normal menu item named *{menu-item}*.The wildcard "*" will match all menu items. (Same as: **:nunme**. See page 339.)

:o The one command that Vi has that *Vim* does not. (In *Vi*, this command puts the editor into "open" mode, a mode that no sane persons ever use if they can avoid it.) (Same as: **:open**. See page 157.)

:om List all the mappings for operator-pending-mode maps. (Same as: **:omap**. See page 298.)

:om {lhs} List the operator-pending mapping of *{lhs}*. (See page 298.)

:om {lhs} {rhs} Define a keyboard mapping for operator-pending mode. (See page 298.)

:omap List all the operator-pending-mode mappings. (Same as: **:om**. See page 298.)

:omap {lhs} List the operator-pending-mode mapping of *{lhs}*. (See page 292.)

:omap {lhs} {rhs} Define a keyboard mapping for operator-pending mode. (See page 298.)

:omapc -or-

:omapclear Clear all the operator-pending-mode mappings. (Same as: **:omapclear**. See page 301.)

:[priority] **ome** {menu-item} {command-string} -or-

:[priority] **omenu** {menu-item} {command-string} Define a menu item that is available for operator-pending mode only.The priority determines its placement in a menu. Higher numbers come first.The name of the menu item is *{menu-item}*, and when the command is selected, the command *{command-string}* is executed. (Same as: **:ome**. See page 334.)

:on[!] -or-

:only[!] Make the current window the only one. If **!** is specified, modified files whose windows are closed will have their contents discarded. (Same as: **:CTRL-W CTRL-O**, **CTRL-Wo**, **:on**. See page 243.)

:ono {lhs} {rhs} -or-

:onoremap {lhs} {rhs} Same as **:omap**, but does not allow remapping of the *{rhs}*. (Same as: **:ono**. See page 301.)

:[priority] **onoreme** {menu-item} {command-string} -or-

:priorityonoremenu {menu-item} {command-string} Like **:omenu**, but the the *{command-string}* is not remapped. (Same as: **:onoreme**. See page 338.)

:op Enter an option-browsing window that enables you to view and set all the options. (Same as: **:bro set**, **:browse set**, **:options**. See page 342.)

:open The one command that *Vi* has that *Vim* does not. (In *Vi*, this command puts the editor into "open" mode, a mode that no sane persons ever use if they can avoid it.) (Same as: **:o**. See page 157.)

:options Enter an option-browsing window that enables you to view and set all the options. (Same as: **:bro set**, **:browse set**, **:op**. See page 342.)

:ou {lhs} -or-

:ounmap {lhs} Remove an operator-pending-mode mapping. (Same as: **:ounmap**. See page 301.)

:ounme {menu-item} -or-

:ounmenu {menu-item} Remove the command-mode menu item named *{menu-item}*.The wildcard "*" will match all menu items. (Same as: **:ounme**. See page 339.)

:range[range] **p** -or-

:range[range] **P** Print the specified lines. (Same as: **:Print**. See pages 100 and 308.)

:pc[!] -or-

:pclose[!] Close the preview window. Discard any changes if the force **(!)** option is present. (Same as: **CTRL-W CTRL-Z**, **CTRL-Wz**, **:pc**. See page 276.)

:pe {command} -or-

:perl {command} Execute a single Perl command. Requires *Vim* be compiled with Perl support (not on by default). (Same as: **:pe**. See page 173.)

:[range] **perld** {command} -or-

:[range] **perldo** {command} Execute a Perl command on a range of lines.The Perl variable $_ is set to each line in range. (Same as: **:perld**. See page 173.)

:[count]**po[!]** -or-

:[count]**pop[!]** Go back count tags. If the current buffer has been modified, this command will fail unless the force **(!)** option is present. (Same as: **:po**. See page 81.)

:[count]**pp[!]** -or-

:[count] **ppop[!]** Do a **:pop** command in the preview window. If the force option **(!)** is specified, discard any changes made on the file in the preview window. If a count is specified, pop that many tags. (Same as: **:pp**. See page 276.)

:pr -or-

:preserve Write out entire file to the swap file.This means that you can recover the edit session from just the swap file alone. (Same as: **:pr**. See pages 151 and 319.)

:[count] **prev** {[+cmd]} {[file—list]} -or-

:[count] **previous** {[+cmd]} [{file—list]} Edit the previous file in the file list. (Same as: **:prev**. See page 43.)

:[range]**print** -or-

:[range]**Print** Print the specified lines. (Same as: **:P**. See pages 100 and 308.)

:pro -or-

:promptfind Open a Find dialog box. (Same as: **:pro**. See page 340.)

:promptr -or-

:promptrepl Open a Replace dialog box. (Same as: **:promptr**. See page 340.)

:pt[!] {identifier} -or-

:ptag[!] {identifier} Open a preview window and do a **:tag**. Discard any changes in the preview window if the override **(!)** option is present. (Same as: **:pt**. See page 276.)

:ptj[!] {identifier} -or-

:ptjump[!] {identifier} Open a preview window and do a **:tjump**. Discard any changes in the preview window if the override **(!)** option is present. (Same as: **:ptj**. See page 276.)

:ptl[!] -or-

:ptlast[!] Do a **:tlast** in the preview window. Discard any changes in the preview window if the override **(!)** option is present. (Same as: **:ptlast**. See page 277.)

:[count] **ptn[!]** Open a preview window and do a :[count] **tnext**. Discard any changes in the preview window if the override **(!)** option is present. (Same as: **:ptn**. See page 276.)

:[count] **ptN[!]** Same as :[count] **ptnext!**. (Same as: **:ptNext**, **:ptprevious**. See page 277.)

:[count] **ptnext[!]** Open a preview window and do a :[count] **tnext**. Discard any changes in the preview window if the override **(!)** option is present. (Same as: **:ptn**. See page 276.)

:[count] **ptNext[!]** Same as :[count] **ptnext!**. (Same as: **:ptNext**, **:ptprevious**. See page 277.)

:[count] **ptp[!]** -or-

:[count] **ptprevious[!]** Do a **:tprevious** in the preview window. Discard any changes in the preview window if the override **(!)** option is present. (Same as: **:ptN**, **:ptNext**, **:ptp**. See page 277.)

:[count] **ptr[!]** -or-

:[count] **ptrewind[!]** Do a **:trewind** in the preview window. Discard any changes in the preview window if the override **(!)** option is present. (Same as: **:ptr**. See page 277.)

:pts[!] {identifier} -or-

:ptselect[!] {identifier} Open a preview window and do a **:tselect**. Discard any changes in the preview window if the override **(!)** option is present. (Same as: **:pts**. See page 276.)

:[line] **pu[!]! register** -or-

:[line] **put[!]! register** Put the text in the register after (before **!** is specified) the specified line. If a register is not specified, it defaults to the unnamed register. (Same as: **:pu**. See page 318.)

:pw -or-

:pwd Print current working directory. (Same as: **:pw**. See page 314.)

:[range] **py** {statement} Execute a single Python *{statement}*. (Same as: **:python**. See page 173.)

:[range] **pyf** {file} Executes the Python program contained in *{file}*. (Same as: **:pyfile**. See page 173.)

:[range] **pyfile** {file} Executes the Python program contained in *{file}*. (Same as: **:pyf**. See page 173.)

:[range] **python** {statement} Execute a single Python *{statement}*.This works only if Python support was compiled into *Vim*; it is does not work by default. (Same as: **:py**. See page 173.)

:q[!] Close a window. If this is the last window, exit *Vim*.The command fails if this is the last window for a modified file, unless the force **(!)** option is present. (Same as: **:CTRL-W CTRL-Q**, **CTRL-Wq**, **:quit**. See pages 9, 46, 144, 202, and 242-243.)

:qa[!] -or-

:qall[!] Close all windows. If the force option is present, any modifications that have not been saved will be discarded. (Same as: **:qa**. See page 242.)

:quit[!] Close a window. If this is the last window, exit *Vim*.The command fails if this is the last window for a modified file, unless the force **(!)** option is present. (Same as: **CTRL-W CTRL-Q**, **CTRL-Wq**, **:q**. See pages 9, 46, 144, 202, and 242-243.)

:r {filename} Read in the specified file and insert it after the current line. (Same as: **:read**. See pages 104, 135, 317, and 396.)

:[line] **r** {file} Read the specified file (default = current file) and insert it after the given line (default = current line). (See page 317.)

:[line] **r !**{command} Run the given command, capture the output, and insert it after the given line (default = current line). (See page 317.)

:read {filename} Read in the specified file and insert it after the current line. (Same as: **:r**. See pages 104, 135, 317, and 396.)

:[line] **read file** Read the specified file (default = current file) and insert it after the given line (default = current line). (See page 317.)

:[line] **read !**{command} Run the given command, capture the output, and insert it after the given line. (Default = current line.) (See page 104.)

:rec[!] {file} -or-

:recover[!] {file} Recover the editing session from the specified file. If no file is specified, the current file is used. If changes have been made to the file, this command will result in an error. If the force **(!)** option is present, attempting to recover a file changed in the current session will discard the changes and start recovery. (Same as: **:rec**. See page 152.)

:red Redo the last edit. (Same as: **:redo**. See page 318.)

:redi[!] {>|>>} {file} -or-

:redir[!] {>|>>} {file} Copy messages to the file as they appear on the screen. If the override option **(!)** is present, the command will overwrite an existing file.The flag “>” tells the command to write the file; the “>>” indicates append mode.To close the output file, use the command **:redir END**. (Same as: **:redi**. See page 321.)

:redo Redo the last edit. (Same as: **:red**. See page 318.)

:reg {list} -or-

:registers {list} Show the registers in list. If no list is specified, list all registers. (Same as: **:reg**. See page 222.)

:res count Change the size of the current window to count. If no count is specified, make the window as large as possible. (Same as: **CTRL-W CTRL-_**, **CTRL-W+**, **CTRL-W-**, **CTRL-W_**, **:resize**. See page 49.)

:res +count Increase the size of the current window by count. (Default = 1.) (Same as: **CTRL-W CTRL-_**, **CTRL-W+**, **CTRL-W-**, **CTRL-W_**, **:resize-**. See page 48.)

:res -count Decrease the size of the current window by count. (Default = 1.) Same as: **CTRL-W CTRL-_**, **CTRL-W+**, **CTRL-W-**, **CTRL-W_**, **:resize-**. See page 48.)

:resize count Change the size of the current window to count. If no count is specified, make the window as large as possible. (Same as **CTRL-W CTRL-_**, **CTRL-W+**, **CTRL-W-**, **CTRL-W_**, **:res**. See page 49.)

:resize +count Increase the size of the current window by count. (Default = 1.) (Same as: **CTRL-W+**, **:res +**. See page 48.)

:resize -count Decrease the size of the current window by count. (Default = 1.) (Same as: **CTRL-W-**, **:res -**. See page 48.)

:[range] **ret [!] tabstop** -or-

:[range] **retab [!] tabstop** Replace tabs at the current tab stop with tabs with the tab stops set at *{tabstop}*. If the 'expandtab' option is set, replace all tabs with space. If the force option **(!)** is present, multiple spaces will be changed into tabs where appropriate. (Same as: **:ret**. See pages 267-268.)

:retu {expression} -or-

:return {expression} Return a value from a function. (Same as: **:retu**. See page 358.)

:rew {file-list} -or-

:rewind {file-list} Edit the first file in the list. (Same as: **:rew**. See pages 44 and 170.)

:[range] **ri** {width} -or-

:[range] **right** {width} Right-justify the specified lines. If the width of a line is not specified, use the value of the 'textwidth'. (If 'textwidth' is 0, 80 is used.) (Same as: **:ri**. See page 115.)

:rv[!] {file} -or-

:rviminfo[!] {file} Read the *vim*info file specified. If the override option is present **(!)**, settings in the file override the current settings. (Same as: **:rv**. See page 233.)

:[range] **s** /{from}/{to}/{flags} Change the regular expression *{from}* to the string *{to}*. See **:substitute** for a list of flags. (Same as: **:substitute**. See pages 102, 144, 160-161, 167-168, 309, and 401.)

:[count] **sa[!]** {number} Do a **:[count]split** followed by **:argument[!]** number. (Same as: **:sargument**. See page 245.)

:[count] **sal** -or-

:[count] **sall** Open a window for all the files being edited.When a count is specified, open up to count windows. (Note that the count can be specified after the command—for example, **:all** count.) (Same as: **:all**, **:sal**. See page 243.)

:[count] **sargument[!]** {number} Do a **:[count]split** followed by :argument**[!]** number. (Same as: **:sa**. See page 245.)

:sb Same as: **:sbuffer**. (See page 51.)

:[count] **sba** -or-

:[count] **sball** Open a window for each buffer. If a count is specified, open at most count windows. (Same as: **:ba**, **:ball**, **:sba**. See page 246.)

:sb number Shorthand for **:split** and **:buffer** number. (See page 51.)

:sbl[!] -or-

:sblast[!] Shorthand for **:split** and **:blast**. (Same as: **:sbl**. See page 52.)

:sbm count[count] -or-

:sbmodified count[count] Shorthand for **:split** and **:bmodified**. (Same as: **:sbm**. See page 52.)

:[count] **sbn** Shorthand for **:split** followed by :[count] **bnext**. (Same as: **:sbnext**. See page 51.)

:[count] **sbN** Shorthand for **:split** and :[count] **bprevious**. (Same as: **:sbNext**, **:sbp**, **:sbprevious**. See page 52.)

:[count] **sbnext** Shorthand for **:split** followed by :[count] **bnext**. (Same as: **:sbn**. See page 51.)

:[count] **sbNext** Shorthand for **:split** and :[count] **bprevious**. (Same as: **:sbN**, **:sbp**, **:sbprevious**. See page 52.)

:[count] **sbp** -or-

:[count] **sbprevious** Shorthand for **:split** and :[count] **bprevious**. (Same as: **:sbN**, **:sbNext**, **:sbprevious**. See page 51.)

:sbr[!] -or-

:sbrewind[!] Shorthand for **:split** and **:brewind**. (Same as: **:sbr**. See page 52.)

:sbuffer number Shorthand for **:split** and **:buffer number**. (Same as: **:sb**. See page 51.)

:se List all options that are not set to the default. (Same as: **:set**. See pages 95, 100, 379, 382, and 394.)

:se {option} Set Boolean option. Depreciated: For all other types of options, show the value of the option. (See page 379.)

:se {option}:{value} -or-

:se {option}={value} Set an *{option}* to a *{value}*. (See page 379.)

:se {option}^={number} -or-

:se {option}**[!]** Invert a Boolean option. (See page 379.)

:se {option}& Set the option to the default value. (See page 379.)

:se {option}+={value} Add a number to a numeric option. For a string option, append the *{value}* to the string. (See page 379.)

:se {option}-={number} Subtract a number to a numeric option. For a string option, remove the *{value}* from the string. (See page 379.)

:se {option}? List the value of an option. (See page 379.)

:se {option}^={number} Multiply a number to a numeric option. Prepend string to the beginning of the option. (See page 379.)

:se all List all options. (See page 382.)

:se all& Set all options to their default values. (See page 382.)

:se inv{option} Invert a Boolean option. (See page 380.)

:se no{option} Clear a Boolean option. (See page 380.)

:set List all options not set to the default. (Same as: **:se**. See pages 18, 95, 100, 379, 382, and 394.)

:set {option} Set Boolean option. Depreciated: For all other types of options, show the value of the option.Depreciated: show all others. (See page 382.)

:set {option}:{value} -or-

:set {option}={value} Set an option. (See page 382.)

:set {option}**[!]** Invert a Boolean option. (See page 382.)

:set {option}**&** Set the option to the default value. (See page 382.)

:set {option}+={value} Add a number to a numeric option. Append a string to a string option. (See page 382.)

:set {option}-={value} Subtract a number from a numeric option. Remove a string from a string option. (See page 382.)

:set {option}**?** List the value of an option. (See page 382.)

:set {option}^={number} Multiply a number to a numeric option. Prepend string to the beginning of the option. (See page 382.)

:set all List all options. (See page 382.)

:set all& Set all options to their default values. (See page 382.)

:set inv{option} Invert a Boolean option. (See page 380.)

:set no{option} Clear a Boolean option. (See page 380.)

:[count] **sf[!]** +command {file} -or-

:[count] **sfind[!] +command** {file} A combination of :count[count] **split** and **:find**. (Same as: **:sf**. See page 281.)

:sh -or-

:shell Suspend the editor and enter command mode (a.k.a. run a shell). (Same as: **:sh**. See pages 104, 144, 152, 319, and 346.)

:si {char} -or-

:simlat {char} Simulate the pressing of Alt-*{char}*. (See page 344.)

:sl {seconds} -or-

:sl {milliseconds)**m** Sleep the specified number of seconds or milliseconds. (Same as: **gs**, **:sleep**. See page 156.)

:sla[!]! -or-

:slast![!] **:split** followed by **:last**. If **!** is specified, modified files whose windows are closed will have their contents discarded. (Same as: **:sla**. See page 245.)

:sleep {seconds} -or-

:sleep {milliseconds)**m** Sleep the specified number of seconds or milliseconds. (Same as: **gs**, **:sl**. See page 156.)

:sm {char} Simulate the pressing of Alt-*{char}*. (Same as: **:smagic**. See page 310.)

: [range] **sm** /{from}/{to}/flags -or-

: [range] **smagic** /{from}/{to}/flags Substitute the pattern *{to}* for the pattern *{from}* for the given range assuming that the **"magic"** option is set for the duration of the command. (Same as: **:sm**. See page 310.)

:[count] **sn[!]** [file-list] **:split** followed by :[count] next If **!** is specified, discard any changes to buffers that have been modified, but not written. If file-list is specified, change the arguments to that list. (Same as: **:snext**. See page 245.)

:[count] **sN[!]** **:split** followed by :count[count] previous. If **!** is specified, discard any changes to buffers that have been modified, but not written. (Note:The count parameter can be specified after the command—for example, **:sN** count[count] .) (Same as: **:sNext**, **:spr**, **:sprevious**. See page 245.)

:[count] **snext[!] file-list** **:split** followed by :[count] next If **!** is specified, discard any changes to buffers that have been modified, but not written. If file-list is specified, change the arguments to that list. (Same as: **:sn**. See page 246.)

:[count] **sNext[!]** **:split** followed by :[count] *previous*. If **!** is specified, discard any changes to buffers that have been modified, but not written. (Note:The count parameter can be specified after the command—for example, **:sN** [count] .) (Same as: **:sN**, **:spr**, **:sprevious**. See page 245.)

:sni {command} -or-

:sniff {command} Perform a command using the interface to Sniff+. If no command is present, list out information on the current connection. Sniff+ support has to be compiled in for this to work (not on by default). (Same as: **:sni**. See page 174.)

: [range] **sno** /{from}/{to}/flags -or-

: [range] **snomagic** /{from}/{to}/flags Substitute the pattern *{to}* for the pattern *{from}* for the given range assuming that the **'nomagic'** option is set. (Same as: **:sno**. See page 310.)

:so {file} -or-

:source {file} Read in a session file. (Actually read in a whole set of commands.) (Same as: **:so**. See pages 95, 294, and 402.)

:[count] **sp** [+cmd] [file-name] -or-

:[count] **split** [+cmd] [file-name] Split the current window. If a count is specified, make the new window count lines high. If a filename is present, put that file in the new window. (Otherwise, use the current file.) (Same as: **CTRL-W CTRL-S**, **CTRL-Ws**, **CTRL-WS**, **:sp**. See pages 45, 47-48, 162, and 247.)

:[count] **spr[!]** -or-

:[count] **sprevious[!]** **:split** followed by :[count] previous. If **!** is specified, discard any changes to buffers that have been modified, but not written. (Note:The count parameter can be specified after the command—for example, **:sN** [count] .) (Same as: **:sN**, **:sNext**, **:spr**. See page 245.)

:sr[!] -or-

:srewind[!] **:split** followed by **:rewind**. If **!** is specified, modified files whose windows are closed will have their contents discarded. (Same as: **:sr**. See page 245.)

:st! Suspend the editor (UNIX terminal only). If the **!** option is not present and **'autowrite'** is set, all changed files will be saved. (Same as: **:stop**, **:sus**, **:suspend**. See page 156.)

:[count] **sta[!]** {function} -or-

:[count] **stag[!]!** {function} A combination of **:split** and **:tag**. If a count is specified it is the height of the new window. (Same as: **:sta**. See page 81.)

:star[!]! -or-

:startinsert[!] ! Begin insert mode as if a normal **i** command had been entered. If the **!** is present, the insert starts at the end of line as if an A command had been issued. (Same as: **:star**. See page 318.)

:stj[!]! {ident} -or-

:stjump[!]! {ident} Do a **:split** and a **:tjump**. (Same as: **:stj**. See page 84.)

:stop[!] ! Suspend the editor (UNIX terminal only). If the **!** option is not present and **'autowrite'** is set, all changed files will be saved. (Same as: **:st**, **:sus**, **:suspend**. See page 156.)

:sts[!]! {ident} -or-

:stselect[!]! {ident} Do a **:split** and a **:tselect**. (Same as: **:sts**. See page 84.)

:[range] **substitute** /{[from}]/{[to}]/{[flags}] Change the regular expression *{from}* to the string *{to}*. (Same as: **:s**. See pages 102, 144, 160-161, 167-168, 309, and 401.) The *{flags}* include the following:

c Confirm. Ask before making a change.

e If the search pattern fails, do not issue an error message (useful for scripts).

g Global. Change each occurrence on the line (not just the first one).

i Ignore case.

p Print each changed line.

r When *{from}* is empty, use the last search pattern rather than the last *{from}* for a **:substitute** command. (See page 100.)

:sun [count] -or-

:sunhide [count] Open a new window for all hidden buffer. Limit the number of window to count, if specified. (Same as: **:sun**, **:unh**, **:unhide**. See page 244.)

:sus[!] -or-

:suspend[!] Suspend the editor (UNIX terminal only). Actually, works in Win32 also.). If the **!** option is not present and **'autowrite'** is set, all changed files will be saved. (Same as: **:stop**, **:sus**. See page 156.)

:sv [+command] [filename] -or-

:sview [+command] [file-name] Split the window like **:split**.The only difference is that the file is opened for viewing. (Same as: **:sv**. See page 48.)

:sw -or-

:swapname List name of the current swap file. (Same as: **:sw**. See page 150.)

:sy List out all the syntax elements. (Same as: **:syntax**. See pages 96, 394, and 405-407.)

:sy case match Syntax definitions are case sensitive. In other words, the case of the letters must match. (See page 407.)

:sy case ignore Syntax definitions are case not sensitive. In other words, case differences are ignored. (See page 407.)

:sy clear Clear out any existing syntax definitions. (See page 407.)

:sy cluster {name} **contains**={groups} **add**={groups} **remove**={group} Define a cluster of syntax groups. (See page 413.)

:sy sync ccomment group-name minlines={min} **maxlines**={max} Tell *Vim* to synchronize based on C-style comments. If a group name is specified, use that group for highlighting; otherwise, use the group name Comment.The 'minlines' and 'maxlines' options tell *Vim* how much to look backward through the file for a comment. (See page 414.)

:sy include @{cluster} {file} Read in a syntax file and put all the defined groups in the specified cluster. (See page 413.)

:sy keyword {group} {keyword} **...** {keyword} options Define a set of keywords for syntax highlighting.They will be highlighted according to *{group-name}*.The options may appear anywhere within the *{keyword}* list. Options can include 'contained', 'nextgroup', 'skipwhite', 'skipnl', 'skipempty', and 'transparent'. Keywords for abbreviations can be defined like 'abbreviation'.This matches both 'abb' and 'abbreviation'. (See page 408.)

:sy list {group-name} List out the named syntax groups. (See page 414.)

:sy list @{cluster-name} List out the elements for syntax cluster. (See page 414.)

:sy match {group} **excludenl** {pattern} options Define a regular expression that matches a syntax element. Options can be 'contained', 'nextgroup', 'skipwhite', 'skipnl', 'skipempty', 'transparent', and 'contains'. (See page 408.)

:sy region options matchgroup={group} **keepend excludenl** \ **start**={pattern} **skip**={pattern} **end**={pattern} Define a syntax-matching region that starts and ends with the specified pattern. Options can be 'contained', 'nextgroup', 'skipwhite', 'skipnl', 'skipempty', 'transparent', 'contains', and 'oneline'. (See page 408.)

:sy sync clear Remove all syntax synchronization directives. (See page 414.)

:sy sync clear {sync-group-name} **sync-group-name ...** Clear all the syntax synchronization commands for the named groups. (See page 414.)

:sy sync match {sync-group-name} **grouphere** {group-name} {pattern} Define a synchronization command (in the group *{sync-group-name}*) that tells *Vim* that when it sees *{pattern}* that the group *{group-name}* follows the match. (See page 414.)

:sy sync match {sync-group-name} groupthere {group-name} {pattern} Define a synchronization command (in the group *{sync-group-name}*) that tells *Vim* that when it sees *{pattern}* that the group *{group-name}* precedes the match. (See page 414.)

:sy sync minlines={min} Define the minimum number of lines for a brute-force synchronization match. (See page 414.)

:sy sync match {match-specification} Define a match or region to be skipped during synchronization. (See page 414.)

:sync -or-

:syncbind Cause all scroll-bound windows to go to the same location. (Same as: **:sync**. See page 276.)

:syntax List out all the syntax elements. (Same as: **:sy**. See pages 96, 394, and 405-407.)

:syntax case match Syntax definitions are case sensitive. In other words, the case of the letters must match. (See page 407.)

:syntax case ignore Syntax definitions are not case sensitive. In other word, case differences are ignored. (See page 407.)

:syntax clear Clear out any existing syntax definitions. (See page 407.)

:syntax cluster {name} **contains**={groups} **add**={groups} **remove**={group} Define a cluster of syntax groups. (See page 414.)

:syntax sync ccomment group-name minlines={min} **maxlines**={max} Tell *Vim* to synchronize based on C-style comments. If a group name is specified, use that group for highlighting; otherwise, use the group name 'Comment'.The 'minlines' and 'maxlines' options tell *Vim* how much to look backward through the file for a comment. (See page 414.)

:syntax include @{cluster} {file} Read in a syntax file and put all the defined groups in the specified cluster. (See page 414.)

:syntax keyword {group} {keyword} **...** {keyword} options Define a set of keywords for syntax highlighting.They will be highlighted according to {group-name}. The options may appear anywhere within the {keyword} list. Options can include 'contained', 'nextgroup' 'skipwhite', 'skipnl', 'skipempty', and 'transparent'. Keywords for abbreviations can be defined like 'abbreviation'. This matches both 'abb' and 'abbreviation'. (See page 408.)

:syntax list {group-name} List out the named syntax groups. (See page 414.)

:syntax list @{cluster-name} List out the elements for syntax cluster. (See page 414.)

:syntax match {group} **excludenl** {pattern} **options** Define a regular expression that matches a syntax element. Options can be 'contained', 'nextgroup', 'skipwhite', 'skipnl', 'skipempty', 'transparent', and 'contains'. (See page 408.)

:syntax region options matchgroup={group} **keepend excludenl** \ **start**={pattern} **skip**={pattern} **end**={pattern} Define a syntax-matching region that starts and ends with the specified pattern. Options can be 'contained', 'nextgroup', 'skipwhite', 'skipnl', 'skipempty', 'transparent', 'contains', and 'oneline'. (See page 408.)

:syntax sync clear Remove all syntax synchronization directives. (See page 414.)

:syntax sync clear {sync-group-name} **sync-group-name ...** Clear all the syntax synchronization commands for the named groups. (See page 414.)

:syntax sync match {sync-group-name} **grouphere** {group-name} {pattern} Define a synchronization command (in the group *{sync-group-name}*) that tells *Vim* that when it sees *{pattern}* that the group *{group-name}* follows the match. (See page 414.)

:syntax sync match {sync-group-name} **groupthere** {group-name} {pattern} Define a synchronization command (in the group *{sync-group-name}*) that tells *Vim* that when it sees *{pattern}* that the group *{group-name}* precedes the match. (See page 414.)

:syntax sync minlines={min} Define the minimum number of lines for a brute-force synchronization match. (See page 414.)

:syntax sync region {region-specification} Define a match or region to be skipped during synchronization. (See page 414.)

:[range] **t** {address} Copy the range of lines below *{address}*. (Same as: **:copy**. See page 306.)

:[count] **ta!** Go forward count tags. (Same as: **:tag**. See pages 81-82.)

:ta! /{pattern} Search for all functions that match the regular expression defined by *{pattern}* and jump to the first one. (See pages 81-82.)

:[count] **tag!** Go forward count tags. (Same as: **:ta**. See pages 81-82.)

:tag! /{pattern} Search for all functions that match the regular expression defined by *{pattern}* and jump to the first one. (See pages 81-82.)

:tags List the tags. (See page 80.)

:tc {command} -or-

:tcl {command} Execute a single Tcl *{command}*. (Same as: **:tc**. See page 174.)

:[range] **tcld** {command} -or-

:[range] **tcldo** {command} Execute a Tcl *{command}* once for each line in the range. The variable "line" is set to the contents of the line. (Same as: **:tcld**. See page 174.)

:tclf {file} -or-

:tclfile {file} Execute the Tcl script in the given *{file}*. (Same as: **:tclf**. See page 174.)

:te {name} -or-

:tearoff {name} Tear off the named menu. (Same as: **:te**. See page 339.)

:tj! ident -or-

:tjump! ident Like :tselect, but if there is only one tag, automatically pick it. (Same as: **:tj**. See page 83.)

:[count] **tl** -or-

:[count] **tlast** Go to the last tag. (Same as: **:tl**. See page 81.)

:tm {menu-item} {tip} -or-

:tmenu {menu-item} {tip} Define the "tip" text that displays when the cursor is placed over an icon in the toolbar. (Same as: **:tm**. See page 337.)

:[count] **tn** Go to the next tag. (Same as: **:tnext**. See page 83.)

:[count] **tN** Go to the previous tag. (Same as: **:tNext**, **:tp**, **:tprevious**. See page 83.)

:[count] **tnext** Go to the next tag. (Same as: **:tn**. See page 83.)

:[count] **tNext** Go to the next tag. (Same as: **:tN**, **:tp**, **:tprevious**. See page 83.)

:[count] **tp** -or-

:[count] **tprevious** Go to the previous tag. (Same as: **:tp**. See page 83.)

:[count] **tr** -or-

:[count] **trewind** Go to the first tag. (Same as: **:tr**. See page 83.)

:ts! ident -or-

:tselect! ident List all the tags that match ident. If ident is not present, use the results of the last **:tag** command. After listing the tags, give the user a chance to select one and jump to it. (Same as: **:ts**. See page 83.)

:tu {menu-item} -or-

:tunmenu {menu-item} Remove a "tip" from an menu item. (Same as: **:tu**. See page 339.)

:u Undo a change. (Same as: **:undo**. See page 318.)

:una {lhs} -or-

:unabbreviate {lhs} Remove the abbreviation. (Same as: **:una**. See page 296.)

:undo Undo a change. (Same as: **:u**. See page 318.)

:unh count[count] -or-

:unhide count[count] Write the file in all windows. (Same as: **:su**, **:sunhide**, **:unh**. See page 244.)

:unl[!] {variable} -or-

:unlet[!]! {variable} Remove the definition of the variable. If the force **(!)** option is present, do not issue an error message if the variable is not defined. (Same as: **:unl**. See page 353.)

:unm[!] {lhs} -or-

:unmap {lhs} Remove a mapping. (Same as: **:unm**. See pages 299 and 301.)

:[mode] **unme** {menu-item} -or-

:[mode] **unmenu** {menu-item} Remove the menu item named *{menu-item}*.The wildcard **"*"** will match all menu items. (Same as: **:unme**. See page 339.)

:[range] **up[!]** [file] -or-

:[range] **up[!]** >> [file] -or-

:[range] **up !**{command} -or-

:[range] **update[!]** [file] -or-

:[range] **update[!]** >> [file] -or-

:[range] **update !**{command} Acts just like the :write command if the buffer is modified. Does absolutely nothing if it's it is not. (Same as: **:up**. See page 316.)

: [range] **v** /{pattern}/ {command} Perform *{command}* on all lines that do not have *{pattern}* in them in the given range. (Same as: **:vglobal**. See page 312.)

:ve -or-

:version List version and configuration information, including the list of VIMRC files read in at startup. (Same as: **:ve**. See pages 95-96.)

: [range] **vg** /{pattern}/ {command} -or-

: [range] **vglobal** /{pattern}/ {command} Perform *{command}* on all lines that do not have *{pattern}* in them in the given range. (Same as: **:v**. See page 312.)

:vi [+cmd] {file} Close the current file and start editing the named file. If +cmd is specified, execute it as the first editing command. (Same as: **:visual**. See page 41.)

:vie [+cmd] {file} -or-

:view [+cmd] {file} Like :vi, but open the file read-only. (Same as: **:vie**. See page 41.)

:visual [+cmd] {file} Close the current file and start editing the named file. If +cmd is specified, execute it as the first editing command. (Same as: **:vi**. See page 41.)

:vm List all the mappings for visual-mode maps. (Same as: **:vmap**. See pages 297-298.)

:vm {lhs} List the visual mode mapping of *{lhs}*. (See pages 297-298.)

:vm {lhs} {rhs} Define a keyboard mapping for visual mode. (See pages 297-298.)

:vmap List all the visual-mode mappings. (Same as: **:vm**. See pages 297-298.)

:vmap {lhs} List the visual-mode mapping of *{lhs}*. (See page 297-298.)

:vmap {lhs} {rhs} Define a keyboard mapping for visual mode. (See page 297-298.)

:vmapc -or-

:vmapclear Clear all the visual-mode mappings. (Same as: **:vmapc**. See page 301.)

:[priority] **vme** {menu-item} {command-string} -or-

:[priority] **vmenu** {menu-item} {command-string} Define a menu item that is available for visual mode only.The priority determines its placement in a menu. Higher numbers come first.The name of the menu item is *{menu-item}*, and when the command is selected, the command *{command-string}* is executed. (Same as: **:vme**. See page 334.)

:vn {lhs} {rhs} -or-

:vnoremap {lhs} {rhs} Same as **:vmap**, but does not allow remapping of the *{rhs}*. (Same as: **:vn**. See page 301.)

:[priority] **vnoreme** {menu-item} {command-string} -or-

:[priority] **vnoremenu** {menu-item} {command-string} Like :vmenu, but the *{command-string}* is not remapped. (Same as: **:vnoreme**. See page 338.)

:[range] **w[!]! filename** Write out the specified file. If no filename is specified, write to the current file.The range defaults to the entire file. If the force **(!)** option is present, overwrite an existing file, or override the read-only flag. (Same as: **:write**. See pages 41-42, 104, 134, 144, 151, 168, 242, and 315-316)

:[range] **w[!]! >> file** Append the specified range to the file.This will fail if the file does not exist unless the force **(!)** option is specified. (See page 310.)

:wa -or-

:wall Write the file in all windows. (Same as: **:wa**. See page 242.)

:wh {expression} -or-

:while {expression} Start a loop. (Same as: **:wh**. See page 356.)

:wi {width} {height} Obsolete older command to set the number of rows and columns. Use :set rows and :set columns instead. (Same as: **:winsize**. See page 325.)

:winp {X} {Y} -or-

:winpos {X} {Y} Set the position of the window on the screen. (Same as: **:winp**. See page 325.)

:winsize {width} {height} Obsolete older command to set the number of rows and columns. Use :set rows and :set columns instead. (Same as: **:wi**. See page 325.)

:wn Shorthand for **:write** and **:next**. (Same as: **:wnext**. See pages 42-43.)

:[count] **wnext[!]** {+command} -or-

:[count] **wNext[!]** {+command} Shorthand for **:write** and **:[count] next**. (Same as: **:wN**, **:wp**, **:wprevious**. See page 43.)

:wp -or-

:[count] **wprevious[!]** Shorthand for :write and **:[count] previous**. (Same as: **:wN**, **:wNext**, **:wp**. See page 43.)

:[range] **wq[!] file** Write the file and exit. If a range is specified, only write the specified lines. If a file is specified, write the data to that file.When the override option **(!)** is present, attempt to overwrite existing files or read-only files. (See pages 202 and 322.)

:wqa[!] -or-

:wqall[!] Shorthand for **:wall** and **:qall**. (Same as: **:wqa**, **:xa**, **:xall**. See page 242.)

:write[!] Write out the current file. (See page 31.)

:[range] **write[!]!** {filename} Write out the specified file. If no filename is specified, write to the current file.The range defaults to the entire file. If the force **(!)** option is present, overwrite an existing file, or override the read-only flag. (Same as: **:w**. See pages 41-42, 104, 134, 144, 151, 168, 242, and 315-316.)

:[range] **write[!]!** >> {file} Append the specified range to the file.This will fail if the file does not exist unless the force **(!)** option is specified. (See page 310.)

:wv[!]! {file} -or-

:wviminfo[!]! {file} Write the *vim*info file specified. If the override option is present **(!)**, any existing file will be overwritten. (Same as: **:wv**. See page 233.)

:[range] **x[!]! file** If the file has been modified, write it.Then exit. If the override **(!)** option is present, overwrite any existing file. (Same as: **:xit**. See page 202.)

:X Prompt for an encryption key and assign the resulting value to the "key" option. (See page 143.)

:range ~flags count Repeat the last substitution, but the last search string as the *{from}* pattern rather than the *{from}* from the last substitution. (See page 311.)

:[range] **xit[!]** {file} If the file has been modified, write it.Then exit. If the override **(!)** option is present, overwrite any existing file. (Same as: **:x**. See page 202.)

:[range] **y** {register} -or-

:[range] **yank** {register} Yank the range (default = current line) into the register (default = the unnamed register). (Same as: **:y**. See page 318.)

:[line] **z**{code} count[count] List the given line (default = current) and a few lines after it.The code controls what section of the text is listed.The count defines what "a few" is. (See page 309.)

:map Mode Table

NVO	N	V	O	IC	I	C
:map	**:nm**	**:vm**	**:om**	**:map!**	**:im**	**:cm**
	:nmap	**:vmap**	**:omap**		**:imap**	**:cmap**
:no	**:nn**	**:vn**	**:ono**	**:no!**	**:ino**	**:cno**
:noreamp	**:nnremap**	**:vnoremap**	**:onoremap**	**:noremap!**	**:inoremap**	**:cnoremap**
:unm	**:nun**	**:vu**	**:ou**	**:unm!**	**:iu**	**:cu**
:unmap	**:numap**	**:vumap**	**:oumap**	**:unmap!**	**:iunmap**	**:cunmap**
:mapc	**:nmapc**	**:vmapc**	**:omapc**	**:mapc!**	**:imapc**	**:cmapc**
:mapclear	**:nmapclear**	**:vmapclear**	**:omapclear**	**:mapclear!**	**:imapclear**	**:cmapclear**

Modes

N Normal
V Visual
O Operator pending
I Insert
C Command line

E

Visual-Mode Commands

<Esc>	Cancel visual mode. (See page 57.)
CTRL-]	Jump to highlighted tag. (See page 60.)
CTRL-\ CTRL-N	Enter normal mode. (See page 58.)
CTRL-G	Toggle between select and visual mode. (See page 259.)
CTRL-V	Switch to visual block mode or exit block visual mode. (See page 59.)
!program	Pipe the selected text through an external program. (See page 57.)
$	Move to the end of the line. (See page 63.)
<	Shift lines to the left (different in block visual mode.) (See page 60.)
=	Indent the lines. (See page 60.)
>	Shift lines to the right (different in block visual mode.)
:command	Execute a colon-mode command on the selected lines.
~	Invert the case of the selected text.
“{register}c	Delete and enter insert mode. (See page 59.)
“{register}C	Delete the selected lines and enter insert mode. (See page 59.)
“{register}d	Delete the highlighted text. (See pages 56, 251, and 252.)

"{register}D	Delete the highlighted lines. (See page 58.)
g?	Rot13 the text. (See page 257.)
gJ	Join the selected lines with no spaces inserted between the words. (See pages 59 and 256.)
gq	Format a block. (See page 256.)
gv	Toggle between the current and previous visual-mode selection. (See pages 252-253.)
J	Join the selected lines.(See pages 59 and 256.)
K	Look up the selected word using the **man** command. (See page 60.)
o	Jump to the other end of a visual selection. (See pages 254 and 255.)
r	Delete and enter insert mode (different in block visual mode.) (See page 59.)
R	Delete the selected lines and enter insert mode. (See page 59.)
"{register}**s**	Delete and enter insert mode. (See page 59.)
"{register}**s**	Delete the selected lines and enter insert mode. (See page 59.)
u	Make the selected case all lowercase. (See page 255.)
U	Make the selected case all uppercase. (See page 255.)
V	Enter line visual mode, or exit to normal mode. (See page 59.)
"{register}x	Delete the highlighted text. (See pages 56, 251, and 252.)
"{register}X	Delete the highlighted lines. (See page 58.)
"{register}y	Yank the highlighted text into a register. (See pages 59, 162, 156.)
"{register}Y	Yank the highlighted lines into a register. (See page 59.)

Visual Block Commands

>	Move the block to the right. (See page 64.)
<	Move the block to the left. (See page 164.)
Astring**<Esc>**	Append *string* to the right side of each line. (See page 63.)
cstring**<Esc>**	Delete the selected text and then insert the *string* on each line. (See page 62.)
Cstring**<Esc>**	Delete selected text to end of line, then insert on each line. (See page 62.)
Istring**<Esc>**	Insert text on the left side of each line. (See page 60.)
O	Go to the other corner diagonally. (See page 255.)
rchar	Replace all the text with a single character. (See page 64.)

Select-Mode Commands

Starting Select Mode

gCTRL-H	Start select block mode. (See page 258.)
gh	Start select character mode. (See page 258.)
gH	Start select line mode. (See page 258.)
gV	Do not automatically reselect an area after a command has been executed. (See page 260.)

Select Mode Commands

Arrow, CTRL, Function Keys (cursor motion)	Extend selection. (See page 258.)
string**<Esc>**	Delete the selected text and replace it with *string*. (See page 258.)
<BS>	Backspace. (See page 258.)
CTRL-H	Delete the selected text. (See page 259.)
CTRL-O	Switch from select mode to visual mode for one command. (See page 259.)

F

Insert-Mode Commands

<BS>	Delete character before the cursor.
char1**<BS>**char2	Enter digraph (only when **digraph** option set).
<C-End>	Cursor past end of file. (See page 228.)
<C-Home>	Cursor to start of file. (See page 228.)
<C-Left>	Cursor one word left. (See page 228.)
<C-Right>	Cursor one word right. (See page 228.)
<CR>	Begin new line.
<Del>	Delete character under the cursor.
<Down>	Cursor one line down. (See page 228.)
<End>	Cursor past end of line. (See page 228.)
<Esc>	End insert mode (unless **'insertmode'** set).(See page 6 and 10.)
<F1>	Same as **<Help>**.
<Help>	Stop insert mode and display help window.
<Home>	Cursor to start of line. (See page 228.)
<Insert>	Toggle insert/replace mode.
<Left>	Cursor one character left. (See page 228.)
<LeftMouse>	Cursor at mouse click. (See page 109.)

<MouseDown>	Scroll three lines downward.
<MouseUp>	Scroll three lines upward.
<NL>	Same as **<CR>**.
<PageDown>	Scroll one screen forward.
<PageUp>	Scroll one screen backward.
<Right>	Cursor one character right. (See page 228.)
<S-Down>	Move one screen forward.
<S-Left>	Cursor one word left.
<S-MouseDown>	Scroll a full page downward.
<S-MouseUp>	Scroll a full page upward.
<S-Right>	Cursor one word right.
<S-Up>	Scroll one screen backward.
<Tab>	Insert a **<Tab>** character.
<Up>	Cursor one line up. (See page 228.)
CTRL-@	Insert previously inserted text and stop insert.
CTRL-[	Same as **<Esc>**.
CTRL-\ CTRL-N	Go to Normal mode.
CTRL-]	Trigger abbreviation.
CTRL-_	When **'allowrevins'** is set: change language (Hebrew, Farsi) {only works when compiled with +rightleft feature}
CTRL-A	Insert previously inserted text. (See page 229.)
CTRL-C	Quit insert mode, without checking for abbreviation, unless **'insertmode'** set. (See page 297.)
CTRL-D	Delete one shift width of indent in the current line. (See page 72 and 262.)
CTRL-E	Insert the character that is below the cursor. (See page 230.)
CTRL-F	
CTRL-H	Same as **<BS>**.
CTRL-I	Same as **<Tab>**.
CTRL-J	Same as **<CR>**.
CTRL-K {character1}{character2}	Insert digraph. (See page 200.)
CTRL-L	When insertmode is set, this command enables you to leave insert mode. (See page 179.)

CTRL-M	Same as **<CR>**.
CTRL-N	Find next match for keyword in front of the cursor. (See page 126 and 128.)
CTRL-O	Quote next character. **CTRL-V** is perferred since some terminals intercept **CTRL-Q** and interpret it as a stop code. (See page 231.)
CTRL-P	Find previous match for keyword in front of the cursor. (See page 125, 126, and 128.)
CTRL-Q	Same as **CTRL-V** (used for terminal control flow). (See page 156.)
CTRL-R register	Insert the contents of a register. (See page 230, 263, and 264.)
CTRL-R CTRL-O register	Insert the contents of a register literally and do not auto-indent. (See pages 263-264.)
CTRL-R CTRL-P register	Insert the contents of a register literally and fix indent. (See pages 263-264.)
CTRL-R CTRL-R register	Insert the contents of a register literally. (See pages 230.)
CTRL-S	Used for terminal control flow.
CTRL-T	Insert one shift width of indent in current line. (See page 263.)
CTRL-U	Delete all entered characters in the current line. (See page 228.)
CTRL-V char	Insert next non-digit literally. (See pages 85, 92, 229, and, 268.)
CTRL-V number	Insert three-digit decimal number as a single byte. (See page 229.)
CTRL-W	Delete word before the cursor. (See page 228.)
CTRL-X mode	Enter **CTRL-X** sub mode, see the following entries.
CTRL-X CTRL-]	Search for the tag that completes the word under the cursor. (See pages 128-130).
CTRL-X CTRL-D	Search for a macro definition for completion. (See pages 128 and 129.)
CTRL-X CTRL-E	Scroll up. (See pages 131.)
CTRL-X CTRL-F	Search filenames for completion. (See pages 128 and 130.)
CTRL-X CTRL-I	Search the current file and **#include** files for completion. (See page 128.)

CTRL-X CTRL-K	Complete identifiers from dictionary. (See page 128.)
CTRL-X CTRL-L	Search the line completion of the line under of the cursor. (See pages 128 and 131.)
CTRL-X CTRL-N	Search for next word that matches the word under the cursor. (See page 128.)
CTRL-X CTRL-P	Search for previous word that matches the word under the cursor. (See page 128.)
CTRL-X CTRL-Y	Scroll down. (See page 131.)
CTRL-Y	Insert the character that is above the cursor. (See pages 229 and 230.)
CTRL-Z	When in insert mode, suspend *Vim*.
^ CTRL-D	Delete all indent in the current line, and restore it in the next. (See page 262.)
0 CTRL-D	Delete all indent in the current line. (See page 262.)

G

Option List

Option	Abbreviation	Type	Scope	Default
aleph	**al**	Number	Global	MS-DOS: 128 Others: 224
Define the first character of the Hebrew alphabet. (See page 177.)				
allowrevins	**ari**	Boolean	Global	Off
Allow the **CTRL-_** command toggle the **'revins'** option. (See page 175.)				
altkeymap	**akm**	Boolean	Global	Off
Define which keyboard map to use as the alternate one. (See pages 176-177.)				
autoindent	**ai**	Boolean	Buffer	Off
Automatically indent each line like the previous one. (See pages 70, 72, 97, 262, 396.)				
autoprint	**ap**	Boolean	Global	Off
Inoperative option put in for *Vi* compatibility. (See page 402.)				
autowrite	**aw**	Boolean	Global	Off
Automatically write files as needed. (See pages 42 and 97.)				
background	**bg**	String	Global	Depends on GUI background
Light or dark, depending on the background color. (See pages 68 and 343.)				
backspace	**bs**	String	Global	“”
Define how **<BS>** works in insert mode. (See page 94.)				

continues

Option	Abbreviation	Type	Scope	Default
backup	**bk**	Boolean	Global	Off
Produce a backup file. (See pages 146-147.)				
backupdir	**bdir**	String	Global	Amiga: **".,t:"** MS-DOS: **".,c:/tmp,c:/temp"** UNIX: **".,~/tmp,~/"**
Specify where the backup files are to go. (See page 147.)				
backupext	**bex**	String	Global	VMS: "_" Others: "~"
Define the extension to append to a backup file. (See page 146.)				
beautify	**bf**	Boolean	Global	Off
Inoperative option put in for *Vi* compatibility. (See page 402.)				
binary	**bin**	Boolean	Buffer	Off
This option enables you to edit binary files. (See pages 121 and 178.)				
bioskey	**biosk**	Boolean	Global	On
On Microsoft Windows, use the BIOS to read keys. (See page 340.)				
breakat	**brk**	String	Global	**" ^I!@*-+_;:,./?"**
Define character at which a line can be broken. (See page 236.)				
browsedir	**bsdir**	String	Global	**"last"**
Define which directory in which to start the **:browse** command. (See page 340.)				
cindent	**cin**	Boolean	Buffer	Off
Do C-style indentation. (See pages 70, 71, and 396.)				
cinkeys	**cink**	String	Buffer	**"0{,0},:,0#,!^F,o,O,e"**
Define the keys that cause a re-indent. (See page 272.)				
cinoptions	**cino**	String	Buffer	""
Define how C indentation is performed. (See pages 272-274.)				
cinwords	**cinw**	String	Buffer	**"if,else,while,do,for,switch"**
Define keywords that cause an extra indent. (See pages 71, 272, and 275.)				
clipboard	**cb**	String	Global	""
Define how the GUI interacts with the system clipboard. (See page 343.)				
cmdheight	**ch**	Number	Global	1
Define the height of the command window. (See page 386.)				
columns	**co**	Number	Global	80 or terminal width
Number of columns in a window. (See page 325.)				
comments	**com**	String	Buffer	**"s1:/*,mb:*,ex:*/,://,b:#,:%, :XCOMM,n:>,fb:-**
Define what *Vim* considers a comment. (See pages 269-270.)				
compatible	**cp**	Boolean	Global	On (but turned off when a Vimrc file is found)
Enable *Vi* compatibility. (See pages 4 and 401.)				

Option	Abbreviation	Type	Scope	Default
complete	**cpt**	String	Global	**".,w,b,u,t,I"**
Define where *Vim* searches for words to complete. (See pages 127-128.)				
confirm	**cf**	Boolean	Global	Off
Enable a confirmation dialog for some commands. (See page 386.)				
conskey	**consk**	Boolean	Global	Off
Do direct console I/O on Microsoft Windows. (See page 384.)				
cpoptions	**cpo**	String	Global	*Vim* mode: **"aABceFs"** *Vi* mode: All flags
Set compatibility options. (See page 401.)				
cscopeprg	**csprg**	String	Global	**"cscope"**
Define where the CScope program resides. (See page 172.)				
cscopetag	**cst**	Boolean	Global	Off
Use CScope for tag navigation. (See page 172.)				
cscopetagorder	**csto**	Number	Global	0
Define the search order for CScope tag commands. (See page 172.)				
cscopeverbose	**csverb**	Boolean	Global	Off
Output verbose information when using CScope. (See page 172.)				
define	**def**	String	Global	**"^#\s*define"**
Define the string that indicates a macro definition. (See page 284.)				
dictionary	**dict**	String	Global	**""**
Define the files to be searched for dictionary words. (See pages 127-128.)				
digraph	**dg**	Boolean	Global	Off
If set, allow digraphs to be entered with as ch1**<BS>**ch2. (See page 200.)				
directory	**dir**	String	Global	Amiga: **".,t:"** MS-DOS: **".,c:\tmp,c:\temp"** UNIX: **".,~/tmp,/var/tmp,/tmp"**
Define the directory where the swap files go. (See page 151.)				
display	**dy**	String	Global	**""**
If set to **"lastline"**, display partial lines as the last line of a screen. (See page 236.)				
edcompatible	**ed**	Boolean	Global	Off
Change the **g** and **c** flags for the substitute command to be **ed** like. (See page 401.)				
endofline	**eol**	Boolean	Buffer	On
Put an **<EOL>** at the end of an incomplete last line. (See page 121.)				
equalalways	**ea**	Boolean	Global	On
When creating windows, make all window sizes equal. (See page 247.)				
equalprg	**ep**	String	Global	**""**
Define the program to use for the = command. (See page 269.)				

continues

Option	Abbreviation	Type	Scope	Default
errorbells	**eb**	Boolean	Global	Off
If set, beep on error. (See page 389.)				
errorfile	**ef**	String	Global	Amiga: **"AztecC.Err"** Others: **"errors.err"**
Define the default error file for the **:clist** and related commands. (See page 89.)				
errorformat	**efm**	String	Global	A long and complex string.
Define how *Vim* parses errors coming out of **make** command. (See page 385.)				
esckeys	**ek**	Boolean	Global	On
Tell *Vim* to accept function keys that send an **<ESC>** string in insert mode. (See page 139.)				
eventignore	**ei**	String	Global	**""**
Define a set of events to ignore. (See pages 85 and 267.)				
expandtab	**et**	Boolean	Buffer	Off
If set, expand tabs to spaces on insert. (See pages 85 and 267.)				
exrc	**ex**	Boolean	Global	Off
Allow reading of initialization files in the current directory. (See page 383.)				
fileencoding	**fe**	String	Buffer	**"ansi"**
Define the encoding of the file. Used for foreign languages. (See pages 137, 176, and 178.)				
fileformat	**ff**	String	Buffer	MS-DOS, OS/2: **"dos"** UNIX: **"unix"** Macintosh: **"mac"**
The format of the current file. (See page 120.)				
fileformats	**ffs**	String	Global	MS-DOS, OS/2: **"dos,unix"** UNIX: **"unix,dos"** Macintosh: **"mac,unix,dos"** Others: **""**
Define the file formats recognized. (See page 120.)				
filetype	**ft**	String	Buffer	**""**
The type of the current file. (See pages 69, 136, and 138.)				
fkmap	**fk**	Boolean	Global	Off
Turn on Farsi keyboard mapping. (See page 176.)				
flash	**fl**	Boolean	Global	Off
Inoperative option put in for *Vi* compatibility. (See page 402.)				
formatoptions	**fo**	String	Buffer	**"tcq"**
Define how text is formatted. (See pages 97, 117-119, and 396.)				
formatprg	**fp**	String	Global	**""**
Define an external command to perform formatting. (See page 119.)				
gdefault	**gd**	Boolean	Global	Off
Make the **g** flag the default for **:substitute** commands. (See page 311.)				

Option	Abbreviation	Type	Scope	Default
graphic	**gr**	Boolean	Global	Off
Inoperative option put in for *Vi* compatibility. (See page 402.)				
grepformat	**gfm**	String	Global	**"%f:%l%m,%f %l%m"**
Defines how *Vim* interprets the output of :**grep**. (See page 289.)				
grepprg	**gp**	String	Global	**"grep -n", Win32: "findstr /n"**
Define the command to run for **:grep**. (See pages 170 and 289.)				
guicursor	**gcr**	String	Global	Complex string
Define how the cursor looks in Microsoft Windows *Vim*. (See page 345.)				
guifont	**gfn**	String	Global	**""**
Define the font to use for the GUI. (See page 344.)				
guiheadroom	**ghr**	Number	Global	50
Define the amount of space above and below the window used by the window manager. (See page 346.)				
guifontset	**gfs**	String	Global	**""**
Define fonts for English and a foreign language. (See page 175.)				
guioptions	**go**	String	Global	GTK: **"agimrtT"** UNIX: **"gmrt"**
Sets various GUI options. (See pages 234, 325, 329, and 343.)				
guipty		Boolean	Global	On
Use a **"pty"** rather than a pipe to connect to :shell commands. (See pag 346.)				
hardtabs	**ht**	Boolean	Global	Off
Inoperative option put in for *Vi* compatibility. (See page 402.)				
helpfile	**hf**	String	Global	MS-DOS: **"$VIMRUNTIME\doc\help.txt"** Others: **"$VIMRUNTIME/doc/help.txt"**
Define the location of the main help file. (See page 401.)				
helpheight	**hh**	Number	Global	Half the current screen.
The height of the help window. (See page 392.)				
hidden	**hid**	Boolean	Global	Off
Automatically hide buffers that are no longer visible. (See page 52.)				
highlight	**hl**	String	Global	Complex string
Define the highlighting for the *Vim* messages. (See pages 394-395.)				
hlsearch	**hls**	Boolean	Global	Off
Highlight search strings. (See pages 29, 97, 203, and 293.)				
history	**hi**	Number	Global	20
Define the size of the command (**:**) history. (See page 321.)				

continues

Option	Abbreviation	Type	Scope	Default
hkmap	**hk**	Boolean	Global	Off
Enable Hebrew keyboard mapping. (See page 177.)				
hkmapp	**hkp**	Boolean	Global	Off
Enable Hebrew with a phonetic keyboard. (See page 177.)				
icon		Boolean	Global	Off (but on when title can be restored)
If set, display the current filename in the icon. (See page 330.)				
iconstring		String	Global	**“”**
The string to be displayed in the icon text. (See page 330.)				
ignorecase	**ic**	Boolean	Global	Off
Ignore case differences in a search and name completion. (See pages 126 and 204.)				
include	**inc**	String	Global	**“^#\s*include”**
Define the format of an **“include”** directive. (See page 284.)				
incsearch	**is**	Boolean	Global	Off
Perform incremental searches. (See pages 30 and 97.)				
infercase	**inf**	Boolean	Buffer	Off
Figure out case of insert completion matches from the current text. (See page 127.)				
insertmode	**im**	Boolean	Global	Off
Enter the strange world where insert mode is the default. (See page 179.)				
isfname	**isf**	String	Global	Complex string
Determine which characters make up a filename. (See pages 186 and 216.)				
isident	**isi**	String	Global	Complex string
Determine which characters make up an identifier. (See page 186.)				
iskeyword	**isk**	String	Buffer	Complex string
Define what characters make up a word. (See pages 184-185 and 217.)				
isprint	**isp**	String	Global	MS-DOS: **“@,~-255”** Others: **“@,161-255”**
Determine which characters are printable. (See pages 186 and 217.)				
joinspaces	**js**	Boolean	Global	On
Put spaces between lines joined with the > command. (See page 116.)				
key		String	Buffer	**“”**
The encryption key. (See page 143.)				
keymodel	**km**	String	Global	**“”**
Define how special keys affect selection mode. (See pages 108 and 333.)				
keywordprg	**kp**	String	Global	Solaris: **“man -s”** Other UNIX: **“man”** DOS: **“”** OS/2: **“view /”** VMS: **“help”**
Define the command to run for the **K** command. (See page 79.)				

Option	Abbreviation	Type	Scope	Default
langmap	**lmap**	String	Global	**“”**
Define a set of mappings for a foreign keyboard. (See page 175.)				
laststatus	**ls**	Number	Global	1
Determine which windows get a status line. (See pages 246-247.)				
lazyredraw	**lz**	Boolean	Global	Off
Do not redraw the screen during macro execution. (See page 400.)				
linebreak	**lbr**	Boolean	Window	Off
Break long lines at nice places. (See page 236.)				
lines		Number	Global	Screen height
Set the number of lines in the edit window. (See page 325.)				
lisp		Boolean	Buffer	Off
Set options that make editing Lisp programs easier. (See pages 396 and 401.)				
list		Boolean	Window	Off
Make invisible characters visible. (See pages 84 and 392.)				
listchars	**lcs**	String	Global	**“eol:$”**
Define how list mode appears. (See pages 85, 392, and 393.)				
magic		Boolean	Global	On
Turn on or off the magic properties of some search characters. (See pages 214 and 309.)				
makeef	**mef**	String	Global	Amiga: **“t:vim##.Err”** UNIX: **“/tmp/vim##.err”** Others: **“vim##.err”**
Define the name of the file to use for **:make** and **:grep** output. (See page 284.)				
makeprg	**mp**	String	Global	**“make”**
Define the program to run for the **:make** command. (See pages 87 and 285.)				
matchpairs	**mps**	String	Buffer	**“(:),{:},[:]”**
Define characters to match for **“%”** and show match. (See page 278.)				
matchtime	**mat**	Number	Global	5
Define the time (in 1/10 seconds) for **'showmatch'** to work. (See page 278.)				
maxfuncdepth	**mfd**	Number	Global	100
Define how deeply function calls may be nested. (See page 400.)				
maxmapdepth	**mmd**	Number	Global	1000
Define the maximum number of nested key mappings. (See page 400.)				
maxmem	**mm**	Number	Global	512
Define the maximum amount of memory for a single buffer. (See page 399.)				
maxmemtot	**mmt**	**number**	**global**	2048, or half the amount of memory available
Define the maximum amount of memory total. (See page 400.)				

continues

Option	Abbreviation	Type	Scope	Default
mesg		Boolean	Global	Off
Inoperative option put in for *Vi* compatibility. (See page 402.)				
modeline	**ml**	Boolean	Buffer	**modeline**
If set, look for modelines in the file. (See page 382.)				
modelines	**mls**	Number	Global	5
The number of lines at the top and bottom to look for modelines. (See page 152.)				
modified	**mod**	Boolean	Buffer	Off
Set to true if the buffer has been modified. (See page 152.)				
more		Boolean	Global	On
When displaying more output than a screen of data, page things through **more**. (See page 395.)				
mouse		String	Global	GUI, MS-DOS: **"a"** Others: **""**
Define which modes enables you to use a mouse. (See page 332.)				
mousefocus	**mousef**	Boolean	Global	Off
If set, the window focus follows the mouse. (See page 331.)				
mousehide	**mh**	Boolean	Global	Off
Hide the mouse while typing in character. (See page 333.)				
mousemodel	**mousem**	String	Global	MS-DOS: **popup:** Others: **extend**
Define how the mouse is to be used. (See pages 108 and 331.)				
mousetime	**mouset**	Number	Global	500
Time between mouse clicks for a double-click. (See page 332.)				
novice		Boolean	Global	Off
Inoperative option put in for *Vi* compatibility. (See page 402.)				
nrformats	**nf**	String	Buffer	**"octal,hex"**
Define which formats are recognized for **CTRL-A** and **CTRL-X**. (See pages 198 and 395.)				
number	**nu**	Boolean	Window	Off
Display line numbers. (See pages 18 and 100.)				
open		Boolean	Global	Off
Inoperative option put in for *Vi* compatibility. (See pages 402.)				
optimize		Boolean	Global	Off
Inoperative option put in for *Vi* compatibility. (See pages 402.)				
osfiletype	**oft**	String	Buffer	RISC OS: **"Text"** Others: **""**
The file type as determined by the OS. (See page 179.)				
paragraphs	**para**	String	Global	**"IPLPPPQPP LIpplpipbp"**
Define the *troff* macros that begin a paragraph. (See pages 122-123.)				

Option	Abbreviation	Type	Scope	Default
paste		Boolean	Global	Off
Define how *Vim* interacts with the system clipboard. (See page 396.)				
pastetoggle	**pt**	String	Global	""
Define a key to switch between **paste** and **nopaste**. (See page 396.)				
patchmode	**pm**	String	Global	""
Turn on **patchmode**, which saves the original file once. (See page 147.)				
path	**pa**	String	Global	UNIX: **".,/usr/include,,"** OS/2: **".,/emx/include,,"** Others: **".,,"**
Set the path to be used for locating **#include** and **:find** files. (See pages 127 and 281.)				
previewheight	**pvh**	Number	Global	12
Define the height of the preview window. (See page 392.)				
prompt		Boolean	Global	Off
Inoperative option put in for *Vi* compatibility. (See page 402.)				
readonly	**ro**	Boolean	Buffer	Off
Set to indicate that the buffer is read-only. (See pages 152-153.)				
redraw		Boolean	Global	Off
Inoperative option put in for *Vi* compatibility. (See page 402.)				
remap		Boolean	Global	On
Allow recursive remapping. (See page 300.)				
report		Number	Global	2
Set the number of lines that must be changed before a message is issued. (See page 392.)				
restorescreen	**rs**	Boolean	Global	On
Try and restore the screen after editing. (See page 395.)				
revins	**ri**	Boolean	Global	Off
Insert character in reverse (for foreign language). (See pages 175 and 396.)				
rightleft	**rl**	Boolean	Window	Off
If set, indicates that the file contains right-to-left encoding. (See page 174.)				
ruler	**ru**	Boolean	Global	Off
Display the status ruler. (See pages 392 and 396.)				
rulerformat	**ruf**	String	Global	Empty
Define the format of a ruler. (See page 392.)				
scroll	**scr**	Number	Window	Half the screen height
Define the number of lines to scroll for **CTRL-D** and **CTRL-U**. (See page 190.)				
scrollbind	**scb**	Boolean	Window	Off
The window that scrolls with other scroll-bound windows. (See page 275.)				

continues

Option	Abbreviation	Type	Scope	Default
scrolljump	**sj**	Number	Global	1
Define the number of lines to scroll at one time. (See page 193.)				
scrolloff	**so**	Number	Global	0
Number of lines to keep above or below the cursor. (See page 193.)				
scrollopt	**sbo**	String	Global	**"ver,jump"**
Define how synchronized scrolling works. (See page 276.)				
sections	**sect**	String	Global	**"SHNHH HUnhsh"**
Define the *troff* macros that define a section boundary. (See page 123.)				
secure		Boolean	Global	Off
Enable secure mode, which prevents the execution of some risky commands. (See page 383.)				
selection	**sel**	String	Global	**"inclusive"**
Define how a selection behaves. (See pages 108 and 344.)				
selectmode	**slm**	String	Global	**""**
Defines the events that can begin select mode. (See pages 108, 258, and 333.)				
sessionoptions	**ssop**	String	Global	**"buffers,winsize,options, help,blank"**
Defines what is saved by a **:mksession** command. (See page 248.)				
shell	**sh**	String	Global	MS-DOS: **command** OS/2: **cmd** UNIX: **$SHELL** or **"sh"**
The name of the command parser. (See page 319.)				
shellcmdflag	**shcf**	String	Global	MS-DOS: **"/c"** or **"-c"** depending on the value of **shell** others: **"-c"**
The flag that tells the shell that a command follows. (See page 319.)				
shellpipe	**sp**	String	Global	**">"**, **"| tee"**, **"|& tee"**, or **"2>&1| tee"**, depending on the value of **"shell"**
The string to pipe the output of the command into something else. (See page 319.)				
shellquote	**shq**	String	Global	MS-DOS: **""** or **"/"**,depending on the value of **'shell"** Others: ""
The quote character to put around the command name. (See page 319.)				
shellredir	**srr**	String	Global	**">"**, **">&"**, or **">%s 2>&1"**
String to redirect the shell output. (See page 319.)				
shellslash	**ssl**	Boolean	Global	Off
If set, always use **"/"** in filenames, even under MS-DOS. (See page 319.)				
shelltype	**st**	Number	Global	0
Define the Amiga shell type. (See page 179.)				

Option	Abbreviation	Type	Scope	Default
shellxquote	**sxq**	String	Global	MS-DOS: "" or "/", depending on the value of shell. UNIX: "\"
The shell quoting characters for commands and redirection. (See page 319.)				
shiftround	**sr**	Boolean	Global	Off
Adjust all shifts to a **shiftwidth** boundary. (See page 269.)				
shiftwidth	**sw**	Number	Buffer	8
Define the width of a shift for the << and >> commands. (See pages 60, 69, 263, 266, and 269.)				
shortmess	**shm**	String	Global	**filnxtToO**
Shorten some messages. (See page 387.)				
shortname	**sn**	Boolean	Buffer	Off
If set, use short filenames for swap filenames. (See page 152.)				
showbreak	**sbr**	String	Global	""
String to display at the beginning of the second part of broken lines. (See page 236.)				
showcmd	**sc**	Boolean	Global	UNIX: Off Others: On
showfulltag	**sft**	Boolean	Global	Off
Show the complete tag when doing a tag search. (See page 130.)				
showmatch	**sm**	Boolean	Global	Off
Show matching brackets in insert mode. (See pages 345 and 396.)				
showmode	**smd**	Boolean	Global	On
Display the current mode on the status line. (See pages 386 and 394.)				
sidescroll	**ss**	Number	Global	0
Define the distance that each horizontal scroll moves. (See page 193.)				
slowopen	**slow**	Boolean	Global	Off
Inoperative option put in for *Vi* compatibility. (See page 402.)				
smartcase	**scs**	Boolean	Global	Off
When **'ignorecase'** is set, assume search strings that are all uppercase want the case to be matched. (See page 204.)				
smartindent	**si**	Boolean	Buffer	Off
Indent like **'autoindent'**, only smarter. (See pages 70-71 and 396.)				
smarttab	**sta**	Boolean	Global	Off
Insert indents at the beginning of a line, normal tabs elsewhere. (See page 266.)				
softtabstop	**sts**	Number	Buffer	0
Define what **tabstop** is to be simulated when Tab is pressed. (See pages 265-266 and 396.)				
sourceany		Boolean	Global	Off
Inoperative option put in for *Vi* compatibility. (See page 402.)				

continues

Option	Abbreviation	Type	Scope	Default
splitbelow	**sb**	Boolean	Global	Off
Make **:split** open window at the bottom rather than the top. (See page 247.)				
startofline	**sol**	Boolean	Global	On
Allow some commands to go past the start or end of a line. (See page 369.)				
statusline	**stl**	String	Global	Empty
Define the format of the status line. (See pages 293 and 389.)				
suffixes	**su**	String	Global	**".bak,~,.o,.h,.info,.swp,.obj"**
List of file suffixes to ignore when searching for files that match a wildcard pattern. (See page 397.)				
swapfile	**swf**	Boolean	Buffer	On
Turn on or off the use of a swap file. (See page 150.)				
swapsync	**sws**	String	Global	**"fsync"**
Tell the operating system to write the swap file to disk. (See page 151.)				
switchbuf	**swb**	String	Global	**""**
Define how the editor behaves when switching buffers. (See page 288.)				
syntax	**syn**	String	Buffer	Empty
The current language used for syntax highlighting. (See pages 136 and 294.)				
tabstop	**ts**	Number	Buffer	8
Define how big a tab is. (See pages 263, 267, and 411.)				
tagbsearch	**tbs**	Boolean	Global	On
Do a binary search of a sorted tag file. (See page 289.)				
taglength	**tl**	Number	Global	0
Define the number of significant characters in a tag. (See page 290.)				
tagrelative	**tr**	Boolean	Global	On
Tags are relative to the directory containing the tag files. (See page 290.)				
tags	**tag**	String	Global	**"./tags,tags"**
Define the list of tag files. (See page 290.)				
tagstack	**tgst**	Boolean	Global	On
Maintain a tag stack. (See page 290.)				
term		String	Global	$TERM, or operating system–dependent value.
Define the name of the terminal. (See pages 137 and 400.)				
terse		Boolean	Global	Off
Makes some error messages a little shorter. (See page 388.)				
textauto	**ta**	Boolean	Global	On
Obsolete. Use the fileformats option instead. (See page 402.)				
textmode	**tx**	Boolean	Buffer	MS-DOS, OS/2: On Others: Off
Obsolete, use fileformats instead. (See page 402.)				

Option	Abbreviation	Type	Scope	Default
textwidth	**tw**	Number	Buffer	0
Set the width of a line. (See pages 97, 114-115, 118-119, and 396.)				
tildeop	**top**	Boolean	Global	Off
Define how the ~ operator works. (See pages 200 and 401.)				
timeout	**to**	Boolean	Global	On
Enable time outs for keyboard input for mapping. (See page 385.)				
timeoutlen	**tm**	Number	Global	1000
Set the amount of time to wait after a key arrives to see whether it is the start of a function key. (See page 385.)				
title		Boolean	Global	Off (but on when title can be restored)
If set, display the current filename in the title bar. (See page 329.)				
titlelen		Number	Global	85
Maximum percentage size of the title bar to be used for a title. (See page 330.)				
titleold		String	Global	"Thanks for flying Vim"
The fallback old title, which is restored if the real one cannot be restored. (See page 330.)				
titlestring		String	Global	""
Define the string to be displayed on the title bar rather than the current filename. (See page 330.)				
toolbar	**tb**	String	Global	"icons,tooltips"
Define how the toolbar appears. (See page 329.)				
ttimeout		Boolean	Global	Off
Enable time outs for keyboard input. (See page 385.)				
ttimeoutlen	**ttm**	Number	Global	-1
Set time out for waiting for mapped keyboard input. (See page 385.)				
ttybuiltin	**tbi**	Boolean	Global	On
Search built-in termcap before using the system one. (See page 400.)				
ttyfast	**tf**	Boolean	Global	System and terminal dependent
Connection to the terminal is fast. (See page 329.)				
ttymouse	**ttym**	String	Global	Depends on the term option
Define how the terminal mouse works. (See page 400.)				
ttyscroll	**tsl**	Number	Global	999
Number of lines to scroll. More than this jumps. (See page 401.)				
ttytype	**tty**	String	Global	Alias for **"term"**
See the **term** option. (See page 137.)				
undolevels	**ul**	Number	Global	UNIX, OS/2: 1000 Others: 100
Define the number of changes remembered for **undo**. (See page 202.)				

continues

Option	Abbreviation	Type	Scope	Default
updatecount	**uc**	Number	Global	200
Specify the character typed before data is saved in the swap file. (See page 150.)				
updatetime	**ut**	Number	Global	4000
Specify the amount of time (in milliseconds) to wait after typing stops before writing the data to the swap file. (See pages 136 and 150.)				
verbose	**vbs**	Number	Global	0
Turn on verbose messages. (See page 402.)				
viminfo	**vi**	String	Global	**“”**
Define a file in which to save information between edits. (See pages 231-233.)				
visualbell	**vb**	Boolean	Global	Off
Beep by flashing the screen. (See apge 389.)				
warn		Boolean	Global	On
Turn on some warning messages. (See page 388.)				
weirdinvert	**wiv**	Boolean	Global	Off
Old compatibility option for some weird terminals. (See page 402.)				
whichwrap	**ww**	String	Global	**“b,s”**
Define what type of commands can wrap past the beginning or end of a line. (See page 189.)				
wildchar	**wc**	Number	Global	**<Tab>**
Define which character starts wildcard completion. (See pages 397-398.)				
wildcharm	**wcm**	Number	Global	None (0)
Define the character that starts wildcard completion in mappings. (See page 397.)				
wildignore	**wig**	String	Global	**“”**
Pattern of filenames to ignore during wildcard completion. (See page 398.)				
wildmenu	**wmnu**	Boolean	Global	Off
When completing wildcards, display a menu of possible files. (See pages 293 and 398.)				
wildmode	**wim**	String	Global	**“full”**
Define how *Vim* handles matches. (See pages 398-399.)				
winaltkeys	**wak**	String	Global	**“menu”**
Define how Vim uses the <Alt> key in Microsoft Windows. (See page 384.)				
window	**wi**	Numeric	Global	Off
Inoperative option put in for *Vi* compatibility. (See page 402.)				
winheight	**wh**	Number	Global	1
Define the minimum size of the current window. (See page 247.)				
winminheight	**wmh**	Number	Global	1
Define the minimum size of the windows that are not current. (See page 247.)				

Option	Abbreviation	Type	Scope	Default
wrap		Boolean	Window	On
Wrap long lines so that they can be seen on the screen. (See pages 233, 236, and 393.)				
wrapmargin	**wm**	Number	Buffer	0
Define the margin at which to start text wrapping. (See page 114 and 396.)				
wrapscan	**ws**	Boolean	Global	On
Define which commands wrap past the beginning or end of a line. (See page 205.)				
write		Boolean	Global	On
Allows the writing of files. (See page 144 and 399.)				
writeany	**wa**	Boolean	Global	Off
Automatically write files without the aide of overrides (**!**). (See page 399.)				
writebackup	**wb**	Boolean	Global	On
Write backup file over the existing one. (See pages 147-148.)				
writedelay	**wd**	Number	Global	0
Delay between output characters for debugging. (See page 402.)				
w300		Numeric	Global	Off
Inoperative option put in for *Vi* compatibility. (				
w1200		Numeric	Global	Off
Inoperative option put in for *Vi* compatibility.				
w9600		Numeric	Global	Off
Inoperative option put in for *Vi* compatibility.				

H

Vim License Agreement

By Bram Moolenaar

Summary

The *Vim* editor is Charityware.You can use and copy it as much as you like, but you are seriously encouraged to make a donation to orphans in Uganda. See the section on “Kibaale Children’s Centre” later in this appendix.

Details

There are no restrictions on distributing an unmodified copy of *Vim*. Parts of *Vim* may also be distributed, but this text must always be included.You are allowed to include executables that you made from the unmodified *Vim* sources, your own usage examples, and *Vim* scripts.

If you distribute a modified version of *Vim*, you are encouraged to send the maintainer a copy, including the source code. Or make it available to the maintainer through FTP; let him know where he can find it. If the number of changes is small (for example, a modified Makefile) emailing the changes will do.When the maintainer asks for it (in any way), you must make your changes, including source code, available to him.

The maintainer reserves the right to include any changes in the official version of *Vim*.This is negotiable.You are not allowed to distribute a modified version of *Vim* when you are not willing to make the source code available to the maintainer.

The current maintainer is Bram Moolenaar (Bram@vim.org). If this changes, it will be announced in appropriate places (most likely www.vim.org and comp.editors). When it is completely impossible to contact the maintainer, the obligation to send him modified source code ceases.

It is not allowed to remove these restrictions from the distribution of the *Vim* sources or parts of it.These restrictions may also be used for previous *Vim* releases rather than the text that was included with it.

If you are happy with *Vim*, please express that by reading the rest of this appendix. You can also have a look at http://www.vim.org/iccf/.

Kibaale Children's Centre

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 the Rakai District. Farmers comprise the bulk of the population. Although people are poor, there is enough food. But this district is suffering from more cases of AIDS per capita than any other part of the world. Some say that it originated in this region. (Estimates are that 10% to 30% of Ugandans are infected with HIV.) Parents are dying, leaving many orphans. In this district, about 60,000 children have lost one or both parents (out of a total population of about 350,000).The deaths are continuing daily.

The children need a lot of help.The KCC works 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.The regime of President Idi Amin and the following civil wars have negatively impacted on education in the area. 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 live in an extended family. An uncle or older sister takes care of them. Because these families are big and the income (if any) is low, a child is lucky if he or she receives healthy food. Clothes, medical care, and schooling are beyond most children's reach.To help these children in crisis, a sponsorship program was put into place. A child can be financially adopted. For a few dollars per month, KCC sees to it that the financially adopted child gets indispensable items, is healthy, and goes to school; KCC takes care of anything else that needs to be done for the child and the family who 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 training is given. Health workers are being trained, hygiene education is carried out, and households are encouraged to build a proper latrine. (I helped setting up a production site for cement slabs.These are used to build a good latrine and are sold below cost.)

There is a small clinic at the project, which provides children and their families with medical help.When needed, transport to a hospital is offered. Immunization programs are carried out, and help is provided when an epidemic threatens (measles and cholera, for example).

From the summer of 1994 to the summer of 1995, I spent a whole year at the centre, working as a volunteer. I helped to expand the centre and worked in the area of water and sanitation. I learned that the help that the KCC provides really makes an impact. Now that I am back in Holland, I want to continue supporting KCC.To do this, I am raising funds and organizing the sponsorship program. Please consider one of these possibilities:

1. Sponsor a child: $15 a month (Holland: fl 27,50)
2. Sponsor a child and the improvement of its environment: $25 a month (Holland: fl 45)
3. Sponsor the health team: Any amount a month or quarter
4. A one-time donation

Compared with other organizations that provide child sponsorship, these amounts are very low.This is because the money goes directly to the centre. Less than 5% is used for administration.This remains possible because this is a small organization that works with volunteers. If you would like to sponsor a child, you should intend 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 working at the centre. I know them personally. The centre is visited at least once a year to check its progress (at our own cost). I have been back to visit the centre myself in 1996, 1998, and 2000.

If you have any further questions, contact the centre directly or send the *Vim* maintainer your queries by email at Bram@vim.org.

The address of the centre is as follows:
Kibaale Children's Centre
P.O. Box 1658
Masaka, Uganda
East Africa

Note

These are Year 2000 prices. Please check the Web page at http://www.vim.org/iccf for the current prices.

Sending Money

United States and Canada

Contact the *Kibaale Children's Fund* (KCF) in Surrey, Canada.You can send them a one-time donation or your sponsorship money directly. Please send me a note so that I know what has been donated because of *Vim*. KCF can also provide more information about sponsorship.

Kibaale Children's Fund
c/o Pacific Academy
10238-168 Street
Surrey, B.C.V4N 1Z4
Canada
Phone: 604-581-5353

Holland

Transfer to the account of Stichting ICCF Holland in Venlo.You might be eligible for a tax deduction based on your contribution(s) to KCC. For more information about this possibility, check with your tax preparer.

Postbank, nr. 4548774

Europe

To avoid banking costs, you should send Bram Moolenaar a Eurocheque, written out to Bram Moolenaar in Dutch Guilders (DFL). But any other method should work. Ask for information about sponsorship.

Stichting ICCF Holland
Bram Moolenaar
Clematisstraat 30
5925 BE Venlo
The Netherlands

Others

Transfer to one of these accounts if possible:

Postbank, nr. 4548774
Swift code: INGB NL 2A, IBAN: NL47 PSTB 0004 5487 74
under the name Stichting ICCF Holland,Venlo
If that does not work: Rabobank Venlo, nr. 3765.05.117
Swift code: RABO NL 2U
under the name Bram Moolenaar,Venlo

Otherwise, send a cheque in U.S. dollars to the address in the preceding section. The minimal amount is $70. (My bank does not accept smaller amounts for foreign cheques, sorry.)

An alternative is to send a postal money order.That should be possible from any country. Use this name: Abraham Moolenaar (which is how it appears on my passport).

Author's Note

By Steve Oualline

The people behind *Vim* have spent a lot of time and effort to make one of the best editors in the world.Yet they do not ask anything for themselves; instead, they ask that you help some of the poorest and most needy people in Africa. Please send them a donation.

If you work for a medium-size or large company, please take the time to tell your boss how much using *Vim* has helped you and encourage your company to make a substantial donation.

The people behind *Vim* are good people. Please help them out.

I

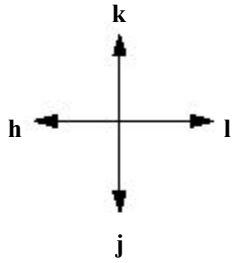
Quick Reference

Karanjit S. Siyan, Ph.D

The following pages contain the maximum amount of useful information in the minimum space, and thus provide a quick reference. Not all commands are covered, but we have tried to include every command you will encounter in day-to-day editing.

Basic Commands

Movement



Undo/Redo

u	Undo the last change
CTRL-R	Redo the last change that was undone
U	Undo all changes on the line.

Getting Out

ZZ	Write file and exit
:q!	Discard changes and exit

Searching

[*count*]**f**{*char*}	Forward search for character on a line
[*count*]**F**{*char*}	Backward search for character on a line
/{*string*}	Search forward for string.
?{*string*}	Search backward for string
{*count*}**n**	Repeat last search
{*count*}**N**	Repeat last search in opposite direction
/	Repeat last search forward
?	Repeat last search backward

Inserting Text

[*count*]**i***text***<Esc>**	Insert text (before cursor)
[*count*]**a***text***<Esc>**	Insert text (after cursor)
o	Open line below cursor
O	Open line above cursor
[*count*]**cw**	Change word
[*count*]**cc**	Change entire line
[*count*]**c**{*motion*}	Change from cursor to {*motion*}.

Deleting Text

[*count*]**dd**	Delete lines.
[*count*]**dw**	Delete words.
[*count*]**x**	Delete characters
[*count*]**d**{*motion*}	Delete from cursor to {*motion*}.

Help

:help {*topic*}	Display help on the given topic.
CTRL-]	Follow help "hyper-link."
CTRL-T	Return to location before last **CTRL-]**.
ZZ	Exit help

Additional Movement

m{*letter*}	Place mark
'{*letter*}	Go to mark
G	Last line
gg	First line
[*count*]**G**	Go to [*count*] line
CTRL-U	Up 1/2 screen
CTRL-D	Down 1/2 screen
z<Enter>	Make this line the top one on the screen
zz	Center window around the current line.

Additional Editing Commands

Editing

.	Repeat last change
[*count*][*"register*]**d**{*motion*}	Delete from cursor to {*motion*}
[*count*][*"register*]**dd**	Delete lines
[*count*][*"register*]**c**{*motion*}	Change from cursor to {*motion*}
[*count*][*"register*]**cc**	Change lines.
[*count*][*"register*]**y**{*motion*}	Yank from cursor to {*motion*}
[*count*][*"register*]**yy**	Yank lines
[*count*][*"register*]**p**	Put after cursor
[*count*][*"register*]**P**	Put before cursor
xp	Twiddle characters
[*count*]**r**{*char*}	Replace one character
[*count*]**R**{*string*}**<ESC>**	Replace characters until stopped
[*count*]**J**	Join lines

Windows

:split {*file*}	Split window
:q	Close current window
CTRL-Wj	Go up a window
CTRL-Wk	Go down a window
CTRL-Wo	Make the current window the only one.

Keyboard Macros

q{*register*}	Record commands into a {*register*}.
q	Stop recording
[*count*]**@**{*register*}	Execute the keyboard macro in {*register*}

Text Formatting Commands

[*count*]**gq**{*motion*}	Format selected text
[*count*]**gqq**	Format lines
[*count*]**gq}**	Format paragraphs
:set textwidth={*width*}	Set the width of the text line (turn auto wrapping on)
:[*range*]**center**	Center the text
:[*range*]**left**	Left-align the text
:[*range*]**right**	Right-align the text
:set formatoptions={*characters*}	Set options which control formatting. Options include:

t	Automatically wrap text
2	When formatting text, use the second line of the paragraph to determine what indent to use.
q	Allow formatting of comments with **gq** command.

Commands for Programmers

General

:syntax on	Turn on syntax highlighting
%	Go to matching brace, comment or **#ifdef**
:set autoindent	Indent each new line the same as the previous one
:set cindent	Use C style indentation
CTRL-D	In insert mode, unindent one level of auto indent.
[*count*]>>	Shift lines right
[*count*]>{*motion*}	Shift right
[*count*]<<	Shift lines left
[*count*]<{*motion*}	Shift left
[*count*]={*motion*}	Indent code from cursor to {motion}
CTRL-]	Jump to tag (function definition).
[*count*]**CTRL-T**	Go to location before last tag jump.
[*count*]**CTRL-W CTRL-]**	Split the window and jump to tag under the cursor.
K	Run man on the word under the cursor

Making the program

:make	Run *make* and capture the output
:cc	Jump to current error
:cn	Jump to next error
:cp	Jump to previous error

Grep

:grep '{*string*}' {*files*}
Run *grep* and capture the output.
Treat matches like **:make** errors.

Include File Searches

[*count*] **]CTRL-D**	Find the definition of the macro under the cursor
[*count*] **]CTRL-I**	Search for the word under the cursor
[*count*] **]d**	List first macro definition for the macro the cursor is on.
[*count*] **]D**	List all macro definitions

Commands for directory searches

:set path={*directory*},....	Tell *Vim* where to search for files
:checkpath	Make sure that *Vim* can find all the files that are referenced in **#include** directives.
:find {*file*}	Edit the {*file*}. If the file is not in the current directory, search through the **'path'** to find it.
gf	Edit the file who's name is under the cursor. Search through the **'path'** if the file is not in the current directory.

Visual Mode Commands

Basic Commands

v	Enter character visual~mode
V	Enter line visual mode
CTRL-V	Enter block visual mode
$	Move the right side of the block to the end of each line.

Selection Commands

[*count*]**a(**	Select () block
[*count*]**a[**	Select [] block
[*count*]**a{**	Select {} block
[*count*]**aw**	Select word and space after
[*count*]**iw**	Select word only.
[*count*]**as**	Select sentence and following space
[*count*]**is**	Select sentence only.

Editing Commands

[*"register*]**d**	Delete the selected text.
[*"register*]**y**	Yank the text into register
>	Shift text right
<	Shift text left
=	Run the lines through the indent program
J	Join the highlighted lines
gq{*motion*}	Format text to **'textwidth'**
U	Convert text to all UPPER CASE
u	Convert text to all lower case.
~	Invert the case of the text.

Insert Mode Commands

Insert Mode Commands

CTRL-V{*char*}	Insert character literally
CTRL-D	Unindent one level
0 CTRL-D	Remove all automatic indentation
CTRL-T	Insert one **'shiftwidth'**
CTRL-Y	Copy the character above the cursor
CTRL-E	Copy the character below the cursor
CTRL-N	Find next match for word before the cursor
CTRL-P	Find previous match for word before the cursor
CTRL-R{*register*}	Insert contents of a register
CTRL-K{*char1*}{*char2*}	Insert digraphs
CTRL-W	Delete word before cursor

Abbreviations

:abbreviate {*abbr*} {*expansion*}
Define abbreviation.

Ex Mode Commands

Basic Commands

Q	Enter Ex mode
:vi	Enter normal mode
:[*range*]**s/**{*old*}/{*new*}/{*flags*}	Substitute {*new*} for {*old*}. If a flag of "g" is present substitute all occurrences on a line. Otherwise just change the first one.
:[*range*]**write** [*file*]	Write text to a file
:[*line*]**read** [*file*]	Read text from another file into the current file.

File Selection Commands

:rewind	Edit the first file
:next	Edit next file
:prev	Edit previous file
:last	Edit the last file in the list
:args	List the files in the edit list.

Editing

:[*range*]**delete**	Delete the specified lines
:[*range*]**copy**[*line*]	Copy the specified lines to [*line*] (default = after current line)
:[*range*]**move**[*line*]	Like**:copy** but delete the lines as well.

Miscellaneous Commands

:[*range*]**print**	Print the specified lines
:[*range*]**number**	Print the specified lines with line numbers
:[*range*]**list**	Print the specified lines with **'list'** option on.
:dlist {*name*}	List definitions for {*name*}
:[*range*]**retab[!]** {*tabstop*}	Change the tabbing from the existing setting to the new {*tabstop*}.
:exit[!]	Close the current file. If it's the last one, exit the editor
:suspend	Suspend the editor
:source {*file*}	Read commands from the specified file.
:redir >{*file*}	Record the output of the commands in the specified file.

Options

Setting	
:set {*option*}= {*value*}	Set an option
:set {*option*}**?**	Display the value of an option
:set	Display the value of options that are set to something other than the default.
:set all	Display the value of all options
:set {*option*}**&**	Set an option to the default
:browse set	Set options using a full screen based dialog.
:set {*option*}+= {*value*}	Add a value to a list of options (string option). Add a number to a numeric option.
:set {*option*}-= {*value*}	Remove a value from a list of options (string option). Subtract a number from a numeric option.
:set {*option*}	Turn on a boolean option
:set no {*option*}	Turn off a boolean option
:set inv{*option*}	Invert a boolean option
Indent / Tabbing	
:set cindent	Turn on C style indentation
:set autoindent	Indent each line the same as the previous one
:set expandtabs	Turn all tabs into space
:set softtabstop	Set the amount of space that the **<Tab>** key uses. (Note: This is not the same as the number of spaces for a Tab.)
Listing Options	
:set list	Turn on list mode where everything is visible
:set number	Display line numbers
Searching Options	
:set hlsearch	Highlight matching search strings
:set incsearch	Do incremental searches
:set wrapscan	If a search reaches the bottom of the file wrap to the top and keep searching. (Also will wrap from the top to the bottom for reverse searches)
:set ignorecase	Make searches case insensitive
:set autowrite	Automatically write files as needed to preserve data.

Regular Expressions

Simple Atoms	
x	The literal character "x"
^	Start of line
$	End of line.
.	A single character
\<	Start of a word.
\>	End of word.
Range Atoms	
[abc]	Match either "a", "b", or "c".
[^abc]	Match anything except "a", "b", or "c".
[a-z]	Match all characters from "a" through "z".
[a-zA-Z]	Match all characters from "a" through "z" and "A" through "Z".
Sub Patterns	
\(*pattern*\)	Mark the pattern for later use. The first set of \(.\) marks a pattern as \1, the second \2 and so on.
\1	Matches the same string that was matched by the first sub-expression in \ (and \). Example: "\([a-z]\).\1" matches "ata", "ehe", "tot", etc.
\2	Like "\1", but uses second sub-expression,
\9	Like "\1", but uses ninth sub-expression.
Modifiers	
*	Match the previous atom 0 or more times. As much as possible.
\+	Match the previous atom 1 or more times. As much as possible.
\=	Match the previous atom 0 or 1 times.
\{}	Match the previous atom 0 or more times. (Same as the "*" modifier.)
\{*n*}	Match the previous atom *n* times.
\{*n*,*m*}	Match the previous atom *n* to *m* times.
\{*n*,}	Match the previous atom *n* or more times.
\{,*m*}	Match the previous atom from 0 to *m* times.
\{-*n*,m}	Match the previous atom *n* to *m* times. Match as little as possible.
\{-*n*,}	Match the previous atom at least *n* times. Match as little as possible.
\{-,*m*}	Match the previous atom up to *m* times. Match as little as possible.
\{-}	Match the previous atom 0 or more times. Match as little as possible.
str1|*str2*	Match *str1* or *str2*.