Gmail **Gal Fisher <galfisher6@gmail.com>**

VIM Jump

3 messages

Gal Fisher <galfisher6@gmail.com> Fri, Jul 7, 2017 at 11:43 AM
To: Gal Fisher <galfisher6@gmail.com>

```
8. Jumps                                        *jump-motions*

A "jump" is one of the following commands: "'", "`", "G", "/", "?", "n",
"N", "%", "(", ")", "[[", "]]", "{", "}", ":s", ":tag", "L", "M", "H" and
the commands that start editing a new file.  If you make the cursor "jump"
with one of these commands, the position of the cursor before the jump is
remembered.  You can return to that position with the "'" and "``" command,
unless the line containing that position was changed or deleted.


                                                        *CTRL-O*
CTRL-O                  Go to [count] Older cursor position in jump list
                        (not a motion command).  {not in Vi}
                        {not available without the |+jumplist| feature}


<Tab>           or                                      *CTRL-I* *<Tab>*
CTRL-I                  Go to [count] newer cursor position in jump list
                        (not a motion command).
                        In a |quickfix-window| it takes you to the position of
                        the error under the cursor.
                        {not in Vi}
                        {not available without the |+jumplist| feature}


                                                        *:ju* *:jumps*
:ju[mps]                Print the jump list (not a motion command).  {not in
                        Vi} {not available without the |+jumplist| feature}


                                                        *jumplist*
Jumps are remembered in a jump list.  With the CTRL-O and CTRL-I command you
can go to cursor positions before older jumps, and back again.  Thus you can
move up and down the list.  There is a separate jump list for each window.
The maximum number of entries is fixed at 100.
{not available without the |+jumplist| feature}

For example, after three jump commands you have this jump list:

  jump line  col file/text
    3     1    0 some text
    2    70    0 another line
    1  1154   23 end.
 >

The "file/text" column shows the file name, or the text at the jump if it is
in the current file (an indent is removed and a long line is truncated to fit
in the window).

You are currently in line 1167.  If you then use the CTRL-O command, the
cursor is put in line 1154.  This results in:

  jump line  col file/text
    2     1    0 some text
    1    70    0 another line
 >  0  1154   23 end.
    1  1167    0 foo bar
```

The pointer will be set at the last used jump position. The next CTRL-O command will use the entry above it, the next CTRL-I command will use the entry below it. If the pointer is below the last entry, this indicates that you did not use a CTRL-I or CTRL-O before. In this case the CTRL-O command will cause the cursor position to be added to the jump list, so you can get back to the position before the CTRL-O. In this case this is line 1167.

With more CTRL-O commands you will go to lines 70 and 1. If you use CTRL-I you can go back to 1154 and 1167 again. Note that the number in the "jump" column indicates the count for the CTRL-O or CTRL-I command that takes you to this position.

If you use a jump command, the current line number is inserted at the end of the jump list. If the same line was already in the jump list, it is removed. The result is that when repeating CTRL-O you will get back to old positions only once.

When the |:keepjumps| command modifier is used, jumps are not stored in the jumplist. Jumps are also not stored in other cases, e.g., in a |:global| command. You can explicitly add a jump by setting the '' mark.

After the CTRL-O command that got you into line 1154 you could give another jump command (e.g., "G"). The jump list would then become:

```
   jump line  col file/text
     4     1    0 some text
     3    70    0 another line
     2  1167    0 foo bar
     1  1154   23 end.
 >
```

The line numbers will be adjusted for deleted and inserted lines. This fails if you stop editing a file without writing, like with ":n!".

When you split a window, the jumplist will be copied to the new window.

If you have included the '' item in the 'viminfo' option the jumplist will be stored in the viminfo file and restored when starting Vim.

CHANGE LIST JUMPS ***changelist* *change-list-jumps* *E664***

When making a change the cursor position is remembered. One position is remembered for every change that can be undone, unless it is close to a previous change. Two commands can be used to jump to positions of changes, also those that have been undone:

g;* *E662

g; Go to [count] older position in change list.
If [count] is larger than the number of older change positions go to the oldest change.
If there is no older change an error message is given.
(not a motion command)
{not in Vi}
{not available without the |+jumplist| feature}

g,* *E663

g, Go to [count] newer cursor position in change list.
Just like |g;| but in the opposite direction.
(not a motion command)
{not in Vi}
{not available without the |+jumplist| feature}

```
When using a count you jump as far back or forward as possible.  Thus you can
use "999g;" to go to the first change for which the position is still
remembered.  The number of entries in the change list is fixed and is the same
as for the |jumplist|.

When two undo-able changes are in the same line and at a column position less
than 'textwidth' apart only the last one is remembered.  This avoids that a
sequence of small changes in a line, for example "xxxxx", adds many positions
to the change list.  When 'textwidth' is zero 'wrapmargin' is used.  When that
also isn't set a fixed number of 79 is used.  Detail: For the computations
bytes are used, not characters, to avoid a speed penalty (this only matters
for multi-byte encodings).

Note that when text has been inserted or deleted the cursor position might be
a bit different from the position of the change.  Especially when lines have
been deleted.

When the |:keepjumps| command modifier is used the position of a change is not
remembered.


                                                        *:changes*
:changes                Print the change list.  A ">" character indicates the
                        current position.  Just after a change it is below the
                        newest entry, indicating that "g;" takes you to the
                        newest entry position.  The first column indicates the
                        count needed to take you to this position.  Example:

                                change line  col text
                                    3     9    8 bla bla bla
                                    2    11   57 foo is a bar
                                    1    14   54 the latest changed line
                                >

                        The "3g;" command takes you to line 9.  Then the
                        output of ":changes is:

                                change line  col text
                                >   0     9    8 bla bla bla
                                    1    11   57 foo is a bar
                                    2    14   54 the latest changed line

                        Now you can use "g," to go to line 11 and "2g," to go
                        to line 14.

==============================================================================

9. Various motions                                      *various-motions*


                                                        *%*
%                       Find the next item in this line after or under the
                        cursor and jump to its match. |inclusive| motion.
                        Items can be:
                        ([{}])          parenthesis or (curly/square) brackets
                                        (this can be changed with the
                                        'matchpairs' option)
                        /* */           start or end of C-style comment
                        #if, #ifdef, #else, #elif, #endif
                                        C preprocessor conditionals (when the
                                        cursor is on the # or no ([{
                                        following)
                        For other items the matchit plugin can be used, see
                        |matchit-install|.  This plugin also helps to skip
                        matches in comments.

                        When 'cpoptions' contains "M" |cpo-M| backslashes
```

before parens and braces are ignored. Without "M" the number of backslashes matters: an even number doesn't match with an odd number. Thus in "(\))" and "\((\)" the first and last parenthesis match.

When the '%' character is not present in 'cpoptions' |cpo-%|, parens and braces inside double quotes are ignored, unless the number of parens/braces in a line is uneven and this line and the previous one does not end in a backslash. '(', '{', '[', ']', '}' and ')' are also ignored (parens and braces inside single quotes). Note that this works fine for C, but not for Perl, where single quotes are used for strings.

Nothing special is done for matches in comments. You can either use the matchit plugin |matchit-install| or put quotes around matches.

No count is allowed, {count}% jumps to a line {count} percentage down the file |N%|. Using '%' on #if/#else/#endif makes the movement linewise.

Gal Fisher <galfisher6@gmail.com> Fri, Jul 7, 2017 at 11:46 AM
To: Gal Fisher <galfisher6@gmail.com>

Use **:changes**
[Quoted text hidden]

Gal Fisher <galfisher6@gmail.com> Fri, Jul 7, 2017 at 11:48 AM
To: Gal Fisher <galfisher6@gmail.com>

Use **:jumps**
[Quoted text hidden]