Open file under cursor

Tip 299 Printable Monobook Previous Next

created 2002 · **complexity** basic · **version** 7.0

Often a file contains the name of a second file, and you would like to open the second file. Do that by putting the cursor on the file name and typing `gf`. This tip explains the details of `gf`.

Go to file

The following commands open the file name under the cursor:

`gf`	open in the same window ("goto file")
`<c-w>f`	open in a new window (Ctrl-w `f`)
`<c-w>gf`	open in a new tab (Ctrl-w `gf`)

We usually type `gf` in normal mode in which case Vim has to determine where the file name starts and ends. It does that by taking all characters matching the `'isfname'` option. Alternatively, you can visually select the file name, then type `gf` in which case Vim will use the selected text.

With the cursor on a file name, pressing Ctrl-w then `f` (or Ctrl-f) will open the file in a new window (after a horizontal split). If you want a vertical split, you could next type Ctrl-w `L` to move the window to the right. Or, you could map a key to perform a vertical "go to file" split (`:vertical` makes what follows split vertically; `:wincmd` is a command that is equivalent to pressing Ctrl-w):

```
:map <F8> :vertical wincmd f<CR>
```

When writing a program, it is helpful to set the `'path'` option to list the directories with your include files. Then you can easily open an include file.

If there are several files in your `'path'` that match the name under the cursor, `gf` opens the first, while `2gf` opens the second, and `3gf` opens the third, etc.

You can return to the previous buffer using Ctrl-^ or Ctrl-o.

File name and line number

Instead of just a file name, you may have a name followed by a line number. For example, `global.h:123` indicates line 123 in file `global.h`. Instead of a colon, any whitespace and another delimiter character can separate the name and number; the delimiter must not be a character expected in a file name (the `'isfname'` option must not include the delimiter).

On Windows, the default `'isfname'` includes a colon. If you do not use drive letters to identify files, you could remove the colon with the command:

```
set isfname-=:
```

The following commands open the file with the cursor on the specified line number:

[illegible]	[illegible]
`<c-w>F`	open in a new window (Ctrl-w `F`)
`<c-w>gF`	open in a new tab (Ctrl-w `gF`)

When such file-name/line number pairs are the result of compiling code, the following commands are also useful:

- :help :cn
- :help :cl
- :help :cfile

The file:line plugin allows you to use combinations of file name and line number, like `global.h:123`, as an argument to Vim. When you open *file:line*, the script checks if *file* exists and *line* is a number. If so, Vim opens *file* at the correct *line* line number.

Names containing spaces

With default settings, Vim regards spaces as delimiters (not part of a file name). That means you cannot type `gf` to open a file if the name contains a space character. However, you can visually select the name (including the spaces), then type `gf` to open the file. For example, you could position the cursor on the first character of the file name, then press `v` to start visual selection, then repeatedly press `E` to move to the end of the next WORD until the whole name is selected, then type `gf`.

As an example on a Windows system, the following line may identify a file:

```
    C:\Documents and Settings\My Name\My Documents\My file.txt
```

With the default settings, if you put the cursor on the "and" word then type `gf`, Vim will try to open a file called `and`. However, you can open the required file by selecting the whole name then typing `gf`. For example, type `^vg_` to select the name, then `gf` to open the file (`^` jumps to the first nonblank character; `v` enters visual mode; `g_` jumps to the last nonblank character).

If you prefer the mouse, depending on your settings, you may be able to select the name with the mouse, then type `gf`. Before typing `gf`, check that you are in visual mode (if the status line shows `--SELECT--`, switch to visual mode by pressing Ctrl-g).

Adjusting `isfname`

To have a space (ASCII 32) considered as a valid character for a file name, add the following to your vimrc:

```
:set isfname+=32
```

On Windows, another approach is to use the 8.3 short path name to avoid spaces. For example, the following tells Vim to look for files in the `C:\Documents and Settings\My User Name\My Documents` directory:

```
:set path+=C:\\DOCUME~1\\MYUSER~1\\MYDOCU~1
```

Double backslashes are required in the `:set` command to get single backslashes in the value. Alternately, use forward slashes:

```
:set path+=C:/DOCUME~1/MYUSER~1/MYDOCU~1
```

Using an environment variable

If you commonly use names with no spaces, but the files are in a directory with spaces, you may like to use the following approach. First, you can use Vim to define an environment variable to identify the directory that you will use, for example (for Windows):

```
:let $mydir = 'C:/Documents and Settings/My Name/My Documents'
```

Suppose you have a file containing a line like the following:

```
My notes are in file $mydir/abc.txt which you can open with gf.
```

Assuming the file `abc.txt` exists in the `$mydir` directory, you can put the cursor on the name then type `gf` to open the file.

If you copy your files to another system using different directories, you can alter the definition of `$mydir`, for example:

```
let $mydir = '/home/myname'
```

The above used forward slashes for the path delimiter; that should work on all systems. On Windows, you could use backslashes instead, and you can mix back and forward slashes (for example, using backslashes in `$mydir` and forward slashes in `$mydir/abc.txt`).

Using selection

Just select the path and then use `gf` or `<c-w>f` or `<c-w>gf`

(This may need Vim 7.2+. Older versions haven't been tested yet.)

See also

- 384 Easily switch between source and header file
- 1546 Automatically add Python paths to Vim path

References

- :help gf
- :help v_gf
- :help CTRL-W_F
- :help CTRL-W_gf
- :help CTRL-^
- :help CTRL-O
- :help 'path'   directory names used to look for the file
- :help 'isfname'   characters considered part of path/file name

Comments

TO DO

- Need brief explanation `'isfname'`.
- Need brief explanation of `'path'`: meanings of . and , , ; how to include space in path.

You can list the files in 'path' that match the name under the cursor with:

```
:echo globpath(&path, expand('<cfile>'))
```

The plugin searchInRuntime   overrides `gf` and `CTRL-W_f` to ask which file must be opened if several match. Alternatives for `:sp` and `:vsp` are also provided.

Note that if your path contains many directories or recursive searches, it can take a very long time to find all matches, and it might be better for you to just stop at the first file found.

I've made my opinion known regarding the "related tips" that John has linked to by **being bold** and just marking them for merge or moving them to "see also", but I don't know what to do with this one:

- 691 Use gf to open a file via its URL

First, it needs a little work to handle escaped characters like %20 for spaces, etc. After this is done, I'm not sure whether it is separate enough of a concept for a new tip, or whether it should be merged as well. I'm leaning toward leaving it separate, but if not, we could add another section just below the one about paths with spaces. --Fritzophrenic 16:15, January 6, 2010 (UTC)

Thanks. I'm pretty ruthless these days and I have just done a "rough merge" in from the three you chose as duplicates (226, 487, 985), and I have replaced each of the merged-in tips with a redirect to here. I do it like that because it's easy, but also because it puts the whole tips (lightly massaged, but no content change) in the history on this page which sometimes helps in the future when checking what's happened. JohnBeckett 04:04, January 7, 2010 (UTC)

I have now done some merging and have added some new bits. The pieces can be tweaked and moved to a better order in due course. JohnBeckett 09:38, January 7, 2010 (UTC)

Using forward slashes

Generally, I prefer forward slashes when writing paths, to preserve multi-platform compatibility. So on a Windows machine which insists on back slashes, I use this conversion command often:

```
" Substitute back slash to forward SLASH.
command! -range Sslash <line1>,<line2>s;\\;/;g
```

Thus :Sslash will work on a single line, or a specified range, e.g. :7,14Sslash for lines 7 through 14. Or better yet, just visualize an area, and then execute the command via ":" which brings up the implied range :'<,'> -- which can be automated by the following visual mapping,

```
" Visualize the desired area, then hit ",s".
vmap ,s :Sslash<CR>
```

The above is fine but it just clutters this tip. It might be somewhere else with a see also from here. JohnBeckett 09:38, January 7, 2010 (UTC)

Following relative links in HTML source

We might want to use gF in an HTML file which we have opened directly from the web with netrw. For example, on the following src attribute:

```
      <script src="../resources/testharnessreport.js">
```

To follow such a link we may need to make use of the protocol and path in netrw_choice or %. This seems like a badly needed feature. I see this is a work in progress in the aforementioned VimTip691. Once complete I think it should feature prominently on this page.

Categories: VimTip | File Handling | Todo | Add category