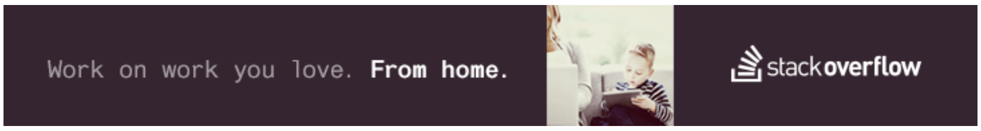
Using Vim's tabs like buffers



I have looked at the ability to use tabs in Vim (with `:tabe` , `:tabnew` , etc.) as a replacement for my current practice of having many files open in the same window in hidden buffers.

I would like every distinct file that I have open to always be in its own tab. However, there are some things that get in the way of this. How do I fix these:

1. When commands like `gf` and `^]` jump to a location in another file, the file opens in a new buffer in the current tab. Is there a way to have all of these sorts of commands open the file in a new tab, or switch to the existing tab with the file if it is already open?
2. When switching buffers I can use `:b <part of filename><tab>` and it will complete the names of files in existing buffers. `<part of filename>` can even be the middle of a filename instead of the beginning. Is there an equivalent for switching tabs?

vim editor tabs

edited Jul 2 '12 at 19:59
Keith Pinson
3,492 4 28 65

asked Sep 19 '08 at 14:42
indentation
3,193 5 14 12

9 Answers

Stop, stop, stop.

This is not how Vim's tabs are designed to be used. In fact, they're misnamed. A better name would be "viewport" or "layout", because that's what a tab is—it's a different layout of windows of *all* of your existing buffers.

Trying to beat Vim into 1 tab == 1 buffer is an exercise in futility. Vim doesn't know or care and it will not respect it on all commands—in particular, anything that uses the quickfix buffer (`:make` , `:grep` , and `:helpgrep` are the ones that spring to mind) will happily ignore tabs and there's nothing you can do to stop that.

Instead:

- `:set hidden`
  If you don't have this set already, then do so. It makes vim work like every other multiple-file editor on the planet. You can have edited buffers that aren't visible in a window somewhere.
- Use `:bn` , `:bp` , `:b #` , `:b name` , and `ctrl-6` to switch between buffers. I like `ctrl-6` myself (alone it switches to the previously used buffer, or `#ctrl-6` switches to buffer number `#`).
- Use `:ls` to list buffers, or a plugin like MiniBufExpl or BufExplorer.

edited Jul 3 '12 at 17:41
OliverUv
352 2 6

answered Sep 19 '08 at 16:44
Zathrus
7,086 2 16 22

149 To me, this is a bug, not "user error". Searching around the web for 'vim tabs' indicates that that just about everyone else disagrees with you, or is unaware of the "real" way to use Vim tabs. Also, if tabs are really "layout" views, then why are default tab titles, the current file name? If I wanted multiple views of the same file, then all my tab titles would be same name (not very helpful?). So either way you look at it, the default tab implementation in Vim is flawed. – cmcginty Jul 10 '09 at 21:10

38 What else are you going to show in the tab title? It shows the current buffer name (not "file name"). You can change what displays in the tab title anyway. – aehlke Aug 10 '10 at 11:06

37 @Casey: FWIW, when i first knew about Vim's tabs and had to wrap my head about them and what they were for, thought exactly of the definition given by Zathrus. (I'm a Vim Novice btw). So no, I don't agree it's such a broken or unintuitive feature...even less agree that "about everyone" disagrees (and think that

at least one or two refs could give more weight to the assertion). It works amazingly well for having multiple, simultaneous screen layouts. – ata Oct 2 '11 at 16:55

16 This comment is wrong, I'm not sure why it's upvoted or accepted. Robince below has the correct answer - :tab sball and :switchbuf are what you're looking for. – Roel Feb 14 '12 at 11:13

24 +1 for "Stop, stop, stop". Vim is as much a mentality as it is a tool. It's important to teach new users the mentality, so they can use the tool to its full potential. – Cody Poll Jun 10 '13 at 10:17

Bit late to the party here but surprised I didn't see the following in this list:

`:tab sball` - this opens a new tab for each open buffer.

`:he switchbuf` - this controls buffer switching behaviour, try `:se switchbuf=usetab,newtab`. This should mean switching to the existing tab if the buffer is open, or creating a new one if not.

Note that `:he` is short for `:help` and `:se` is short for `:set`.

edited Mar 14 at 11:45

answered Aug 13 '10 at 11:36
robince
8,155 1 21 37

1 +1 for `switchbuf - set switchbuf=useopen` is great to avoid the annoying behavior of `:make`, `:vimgrep` and similar commands that changes the buffers in the window layout even when the buffer was already displayed. – mMontu Feb 2 '12 at 10:36

24 When writing an answer like this, could you please type full command names? I didn't realize that he -> help until I tried it. It's going to be read more times that written, so please? – Theo Belaire Jan 27 '14 at 22:38

4 @robince I knew about the ability to truncate commands, but I didn't know if there was some other command that `:he` could expand to. I had to go start vim to know what your answer was talking about, which was an annoyance that could have been avoided with 3 extra characters in your answer. Imagine if the instructions for building a project where "run `./con<tab> && make a<tab> && make in<tab>`". Sure, you could figure it out, but it isn't as clear to read. – Theo Belaire May 30 '14 at 16:43

7 Exactly. A little verbosity can help save lives (and sometimes files), which is why Vim help shows full command names and how much to type to safely abbreviate them. Lest someone who is used to typing `:ptb` to get a traceback from his debugger accidentally launches a nuclear strike because Vim at the control station at the missile silo he's at has this command secretly mapped to run `:PushTheButton` (without those pesky launch codes and such)... Just joking, but I guess you know where I'm getting at. – ack Jun 24 '14 at 16:07

6 @Tyr: On the other hand, you can always use vim help to find out what the abbreviation means: `:he he`. Hehe. – Tony Aug 31 '14 at 0:59

Vim help explains the confusion "tabs vs buffers" pretty well.

> A buffer is the in-memory text of a file.
> A window is a viewport on a buffer.
> A tab page is a collection of windows.

Opening multiple files is achieved in vim with **buffers**. In other editors (e.g. notepad++) this is done with tabs, so the name tab in vim maybe misleading. **Windows** are for the purpose of splitting the workspace and displaying multiple files (buffers) together on one screen. In other editors this could be achieved by opening multiple GUI windows and rearranging them on the desktop. Finally in this analogy vim's **tabs** would correspond to multiple desktops, that is different rearrangements of windows.

As vim help explains a tab can be used, when one wants to temporary edit a file, but does not want to change anything in the current layout of windows and buffers. In such a case another tab can be used just for the purpose of editing that particular file.

Of course you have to remember that displaying the same file in many tabs or windows would result in displaying the same working copy (buffer).

answered Jan 23 '12 at 15:05
crenate
1,935 1 10 12

Contrary to some of the other answers here, I say that you can use tabs however you want. vim was designed to be versatile and customizable, rather than forcing you to work according to predefined parameters. We all know how us programmers love to impose our "ethics" on everyone else, so this achievement is certainly a primary feature.

`<C-w>gf` is the tab equivalent of buffers' `gf` command. `<C-PageUp>` and `<C-PageDown>` will switch between tabs. (In Byobu, these two commands never work for me, but they work outside of Byobu/tmux. Alternatives are `gt` and `gT`.) `<C-w>T` will move the current window to a new tab page.

If you'd prefer that vim use an existing tab if possible, rather than creating a duplicate tab, add `:set switchbuf=usetab` to your .vimrc file. You can add `newtab` to the list (`:set switchbuf=usetab,newtab`) to force QuickFix commands that display compile errors to open in separate tabs. I prefer `split` instead, which opens the compile errors in a split window.

If you have mouse support enabled with `:set mouse=a`, you can interact with the tabs by clicking on them. There's also a `+` button by default that will create a new tab.

For the documentation on tabs, type `:help tab-page` in normal mode. (After you do that, you can practice moving a window to a tab using `<C-w>T`.) There's a long list of commands. Some of the window commands have to do with tabs, so you might want to look at that documentation as well via `:help windows`.

Addition: 2013-12-19

To open multiple files in vim with each file in a separate tab, use `vim -p file1 file2 ...`. If you're like me and always forget to add `-p`, you can add it at the end, as vim follows the normal command line option parsing rules. Alternatively, you can add a bash alias mapping `vim` to `vim -p`.

edited Jul 14 '14 at 18:58

answered Nov 11 '12 at 5:27
Zenexer
7,580 4 35 51

1 +1 for `vim -p` however be aware that only up to 10 tabs will be display by default. Extra files are loaded into buffers, but not displayed in their own tab. – IanB Oct 22 '14 at 23:37

I ran into the same problem. I wanted tabs to work like buffers and I never quite manage to get them to. The solution that I finally settled on was to make buffers behave like tabs!

Check out the plugin called Mini Buffer Explorer, once installed and configured, you'll be able to work with buffers virtaully the same way as tabs without losing any functionality.

answered Sep 19 '08 at 15:12
Dominic Dos Santos
1,148 8 14

Looking at :help tabs it doesn't look like vim wants to work the way you do...

Buffers are shared across tabs, so it doesn't seem possible to lock a given buffer to appear only on a certain tab.

It's a good idea, though.

You could probably get the effect you want by using a terminal that supports tabs, like multi-gnome-terminal, then running vim instances in each terminal tab. Not perfect, though...

answered Sep 19 '08 at 14:50
Mike G.
1,550 9 15

I tried that, the problem with that, besides making it difficult to navigate between tabs, is that yanked text can't be shared across terminal tabs, as they are essentially two different terminals/shells – puk Jan 6 '12 at 10:10

@puk : Try the YankRing vim plugin. It saves yanked text in a file, so it can be shared accross vim instances. – pixelastic Feb 26 '12 at 22:14

@Pixelastic I found a suitable solution here for yanking to system clipboard. Also, Vim works best in one instance. I stopped making it do what I wanted it, and learned how to properly use vim :-) stackoverflow.com/a/8757876/654789 – puk Feb 26 '12 at 22:32

@puk : Thanks, it didn't occurred to me that using the system clipboard means sharing it across vim instances as well :) – pixelastic Apr 13 '12 at 12:53

- You can map commands that normally manipulate buffers to manipulate tabs, as I've done with gf in my .vimrc:

  ```
  map gf :tabe <cfile><CR>
  ```

  I'm sure you can do the same with [^

- I don't think vim supports this for tabs (yet). I use gt and gT to move to the next and previous tabs, respectively. You can also use Ngt, where N is the tab number. One peeve I have is that, by default, the tab number is not displayed in the tab line. To fix this, I put a couple functions at the end of my .vimrc file (I didn't paste here because it's long and didn't format correctly).

answered Sep 19 '08 at 14:53
Lucas Oman
11.2k 1 36 43

`<c-w>gf` will open the filename under the cursor in a new tab – Peter Rincker Apr 24 '11 at 21:08

1 @Lucas Oman: Your link doesn't work any more. – kynan Dec 6 '11 at 13:47

I use buffers like tabs, using the BufExplorer plugin and a few macros:

```
" CTRL+b opens the buffer list
map <C-b> <esc>:BufExplorer<cr>

" gz in command mode closes the current buffer
map gz :bdelete<cr>

" g[bB] in command mode switch to the next/prev. buffer
map gb :bnext<cr>
map gB :bprev<cr>
```

With BufExplorer you don't have a tab bar at the top, but on the other hand it saves space on your screen, plus you can have an infinite number of files/buffers open and the buffer list is searchable...

answered Sep 19 '08 at 22:46
jkramer
9,430 4 32 47

If you want buffer to work like tab, check out the tabline plugin - http://vim.sourceforge.net/scripts/script.php?script_id=1507

That uses a single window, and add a line on the top to simulate the tab (just showing the list of buffers). This came out a long time ago when tab was only supported on the GUI vim but not on the command line vim. Since it is really operate with buffers, everything integrate well with the rest of vim.

answered Jun 27 '13 at 21:12
thien.vuong
21 2