Introduction to tmux

Learning tmux can be daunting with all of its options and hotkeys. Here's what you need to get up to speed quickly.

CHARLES THOMAS

PREVIOUS
Feature: The Power of Tiny initrd

NEXT
Doc Searls' EOF

tmux is one of those programs that seemed to have so many options and new commands, I thought it wasn't worth learning. Not because it wouldn't be useful, but because I assumed the learning curve would be too steep. Now that I've finally dug in to it and made it a part of my work flow though, I can't imagine going back to life without it, and I regret every time I thought about learning it and didn't. tmux does have a lot of bells and whistles, but knowing them isn't vital to its usefulness. In other words, you can do a lot with just a little tmux knowledge and a good cheatsheet.

tmux (terminal multiplexer) functions like screen: you create a session and do work inside it. At any time, you can detach the session, only to reattach later and find everything set up exactly as you left it when you detached. The feature of tmux that sets it apart from screen is the ability to split your terminal window into multiple window panes and rearrange them on the fly. Screen allows you to create multiple windows per session, but with tmux, you then can split each window vertically or horizontally (or both) and resize the panes on a whim. It's also trivial to swap panes around the window and even rotate through a pre-selected set of pane groupings—like in a tiling window manager.

A Little Background

When I first started working in operations, I found I was in and out of multiple SSH sessions to dozens of machines all day long. Pretty soon, I started looking for a terminal emulator that could handle tabs and the functionality described above, and I found Terminator for Linux.

When I moved to a QA role in a startup, I was handed a MacBook, and the first thing I did was find the OS X equivalent: iTerm2. That served me fairly well until a fateful day working out of a coffee shop while getting my car repaired. My current job involves launching anywhere between 5 and 50 cloud instances at a time to test against. The coffee shop's Wi-Fi cut out so much that I spent all day trying to launch a single set of instances and never got any actual work done.

Once my car was finished and I made it home, I dug an old Raspberry Pi out of a closet and set it up to launch instances from it. I would `ssh` in and open screen, then launch instances to my heart's content. If my laptop's Wi-Fi cut out, it didn't matter. My SSH connection to the Raspberry Pi would die, but the instances were launching in screen, so the session wasn't lost; I just couldn't see it. When the Wi-Fi came back, I'd `ssh` back in and reattach to my screen session. This gave me the added bonus of being able to issue the tear down command and sleep my laptop right away instead of waiting for the instances to be destroyed. That meant I didn't have to budget an extra 20 minutes into the end of my day for waiting on instance clean up.

This scheme worked pretty well, but eventually I realized I also could move my entire development environment to the Raspberry Pi, and it would mean less context switching and shuffling files and config around. Moving my whole setup to the Raspberry Pi was another big step in the right direction, but I pretty quickly realized I was missing the multi-paned windows of iTerm2, and screen's tabs just weren't enough. So I finally gave in and tried tmux.

Getting Started with tmux

If tmux isn't already installed by default in your distribution, it's almost certainly in your package manager (apt, yum and so on). After installation, tmux will launch your first session. You should see what looks like your normal prompt, but at the bottom of your terminal is the tmux status line. At the far right of the status line is your hostname followed by the time and date. The left side should show something like `[0] 0: bash*`. The 0 in square brackets `[0]` is the session name. Knowing the session name is important if you want to use more than one session and attach to the correct one, but that won't be covered in this introduction. The rest of the text on the left is the window list. Since you've just opened tmux for the first time, you have only the default window (number 0), and it will be displaying the current running program (bash, or your shell of choice, since it's idle). The asterisk denotes the active window. It's worth noting that `"hostname"` in quotation marks on the right side of the status bar is actually whatever the title of your terminal is; hostname is just the default. So if you run something that normally would change the name of your terminal window, that change will now be reflected on the right in the tmux status line.

Pressing Ctrl-b is the magic keystroke to access tmux command mode. This is similar to Ctrl-a in screen, with one big advantage: outside of screen, typing Ctrl-a will move your cursor to the beginning of your pending command string. Inside screen, you have to press Ctrl-a and then a again. Because tmux uses Ctrl-b for commands, Ctrl-a still works like it does outside of tmux.

To detach from your tmux session, press Ctrl-b, then press d for detach. Attaching is the one area where screen has an edge on tmux. Screen has flags that will attach an existing session if one exists and create one if it doesn't (`screen -dRR`), which means it can be easily aliased. Without extra configuration, `tmux` always will create a new session, and `tmux attach` will attach only to an existing session. There is no "attach or create". One way around this is `alias tm='tmux attach || tmux'` Running `tmux attach` will try to attach to your detached session and return 1 if it failed, because there are no sessions, so `tmux attach || tmux` will try to re-attach and create a new one if that failed.

Moving Around

After you've successfully created a tmux session, you'll want to start opening windows and splitting them into multiple panes—the good stuff. Create a new window with Ctrl-b, c. You'll see in your window list below that you now have something like `[0] 0: bash 1: bash*`. The `1` is now the active window (so the asterisk moved). You can navigate through the list of windows backward with Ctrl-b, p (for previous), forward Ctrl-b, n (for next), and you can jump directly to a numbered window with Ctrl-b followed by the number. So if you're still on window 1, pressing Ctrl-b, 0 will move you back to window 0.

If you have more than ten windows open, that method won't work. Trying to do Ctrl-b, 10 will just put you on window 1 before you can type 0. Once you have a double-digit number of windows open, you'll want to use Ctrl-b, ', which prompts you for an index. Type the number you want to go to, and when you press return, you'll go to that number, so you can type 34 without being sent to window 3.

Moving forward and backward through your window list wraps, so if you have three windows open (0, 1, 2) and you're on two, pressing Ctrl-b, n will take you to window 0. And, then Ctrl-b, p will take you from window 0 back to window 2.

If you find yourself going back and forth between two non-consecutive windows, rather than remembering their numbers or shuffling through

You can keep splitting your screen beyond that too. Each time, the current pane will be split in half leaving the rest of the panes as they were.

with n or p, use Ctrl-b, l to swap back and forth. This will take you to your previous window, kind of like how `cd -` changes back and forth between two directories on the command line.

Splitting Windows into Multiple Panes

Splitting windows into multiple panes is a similar process. To split a window pane in half horizontally, press Ctrl-b, ". To split vertically, press Ctrl-b, %. You can keep splitting your screen beyond that too. Each time, the current pane will be split in half leaving the rest of the panes as they were.

You can navigate between panes with the arrow keys (after pressing Ctrl-b, of course). Or you can cycle through all the panes with Ctrl-b, o.

Knowing which pane is "next in line" can be tricky with lots of panes in lots of different sizes. To find out how the panes are numbered (and which pane will be active next with Ctrl-b, o), use Ctrl-b, q to display an index in each of the active window's panes.

Finally, to swap back and forth between panes, Ctrl-b, ; works with panes like Ctrl-b, l does with windows as described above.

One minor disadvantage of tmux (and screen) is the inability to scroll up through your terminal history with a scroll bar. But, you still can scroll back, just press Ctrl-b, [first. Then you can navigate up the screen with the arrow keys. This is actually copy mode. tmux has its own copy/paste buffers, which are active across the session, so you can copy from one pane into another (or into another window).

Copy Mode

Unfortunately, this is where tmux gets a little funky. tmux has two different sets of keys to perform the same operations: one that emulates emacs, and the other emulates vi. Describing all of the different operations and their emacs and vi bindings is beyond the scope of this introduction; see the manual page for details. What is important is that the way you highlight text is different depending on the mode. The default mode is emacs. You can change that in a number of ways, but worth mentioning here is by your VISUAL and/or EDITOR environment variables, because it can change the bindings without you expressly making the change yourself. If either of those contain vi (for example, vi or vim), tmux will set your mode to vi.

Once you're in copy mode (by pressing Ctrl-b, [), press the spacebar, and then use the arrow keys to move your cursor around. If the text is highlighted, you are in vi mode. If not, navigate back to what you want to copy and use Ctrl-spacebar to start highlighting instead. After using the arrow keys (or emacs or vi shortcuts) to highlight the text you want to copy, use spacebar or Ctrl-spacebar again to copy the highlighted text into the buffer and exit copy mode. Now use the keys described above to navigate to the window and pane you want to paste into, and use Ctrl-b,] to paste.

Resizing and Rearranging

Once you have your window panes all split up, you may find you want to resize or rearrange them. This process is also slightly more complicated, but it's not as confusing as copy mode's multiple key combination stuff. Resizing can be done by one- or five-line increments. To change a panel's size by one line, use Ctrl-b then Ctrl-arrow key, which will expand or contract the pane (depending on the orientation) in the direction of the arrow key. This also will have the opposite effect on the adjacent pane. tmux has no control over the size of the whole window; that is up to your window manager. If you expand one pane, the adjacent pane must shrink. To resize by five-line increments, the process is identical, except instead of Ctrl-arrow key, press Ctrl-b, Alt-arrow key.

Rearranging panes is also possible. If you have a layout you like but want some panes swapped around, use Ctrl-b, { or Ctrl-b, } to swap the current pane with the previous or next pane. (Press Ctrl-b, q to see the pane indices, so you know which pane will be swapped with the current pane.)

To change your pane layout completely, reshaping and redrawing

If you find that you have one pane you just can't fit into your current window the way you like, there's also an option to break it out.

all of the panes, use Ctrl-b, spacebar to rotate through five different layout presets:

- Even-horizontal: all panes in even columns.
- Even-vertical: all panes in even rows.
- Main-horizontal: one big row on top with the rest of the panes in columns underneath.
- Main-vertical: one big column on the left with the remaining panes in rows on the right.
- Tiled: all panes in even rectangles.

According to the manual page, you also can use Ctrl-b, Alt-[1-5] in order to rearrange the panes directly in the style described above, respectively, without cycling through the others. For whatever reason, I was unable to get that to work, but your mileage may vary.

If you find that you have one pane you just can't fit into your current window the way you like, there's also an option to break it out. Pressing Ctrl-b, ! will remove the current pane from the current window and move it to a new window of its own.

Rearranging whole windows is much simpler. Pressing Ctrl-b, . will prompt for a new index for the current window. You must provide an index that is not already in use. This will rearrange the windows so that the window you re-indexed now will be in the correct spot when cycling through windows with Ctrl-b, n or Ctrl-b, p.

On the face of it, this would suggest that you can re-index windows only at the end of the current list, but that isn't strictly true. When you close a window, the existing windows are not reordered. So if you have windows 0, 1, 2, 3, and you were to close window 1, you then would have 0, 2, 3. Windows 2 and 3 do not shift down to fill in the gap. So, pressing Ctrl-b, . would re-index any of those windows as window 1.

It's also worth noting that when you create a new window, the first missing index is used rather than adding on to the end. Using the same example, if you have windows 0, 2, 3 and you use Ctrl-b, c to create a new window, you will end up with 0, 1, 2, 3 again, not 0, 2, 3, 4.

Conclusion

This should be more than enough to get you started with tmux. There's plenty more functionality to explore in the man page, such as rebinding keys, customizing your status line, even broadcasting your keystrokes to all the visible window panes instead of just the active one. You also can make tmux config files to automate setting up your tmux environment. You can define a set of windows and panes to be opened and arranged automatically when you start a new session. You even can tell tmux to run a command inside a pane automatically when the session starts!

But, don't be intimidated by all those bells and whistles like I was. You can get a lot of benefit out of tmux, even if all you do is open a couple windows and split them into two panes on occasion. The man page has a reasonably good list of hotkeys, but they aren't in any order that I could make sense of, so I made a chart (see the tmux Cheatsheet sidebar), which should help get you started with the basics described in this article.■

Charles Thomas is a Selenium tester, Python developer and open-source contributor. He can be found all over the Web, but cha.rlesthom.as is a good place to start.

Send comments or feedback via
http://www.linuxjournal.com/contact
or to ljeditor@linuxjournal.com.

Advanced Topic—SSH and tmux

There is one advanced topic worth mentioning, since it's beyond the scope of the manual page and took some Internet sleuthing (Duck Duck Go and Stack Overflow) for me to figure out: ssh-ing directly to tmux. My primary use of tmux is on my remote development machine. I am sufficiently lazy that opening a new terminal window, ssh-ing to my dev box, then trying to figure out if I already have a tmux session going and either attaching to it or creating a new one is all just too much work. The alias I described above would take care of the "create new session" versus "attach to existing session" problem. But I can do one better: automatically create or re-attach when SSH connects.

There actually are several ways to approach this, each with pros and cons. The most obvious to me was to set tmux as my shell, using chsh or editing /etc/passwd. Any relatively up-to-date distribution and tmux version should Just Work. tmux does have a flag for running correctly as a shell in older versions (see the man page). The problem with this approach is that without a custom tmux config file, you are connected to a new session. Pressing Ctrl-b, s will show you a menu of sessions and allow you to use arrow keys and enter to select the old session you detached from, but then you have to deal with cleaning up your new session(s), which involves entering tmux commands after using Ctrl-b, : to access the tmux shell (again, see the man page).

You also could invoke tmux inside your .bashrc (or the equivalent in your shell of choice), but you have to add checks to make sure you aren't already in tmux, or you could end up in a loop. Your shell launches tmux, which launches a shell, which runs your .bashrc again launching another tmux, and so on. This option seemed sufficiently dangerous to me that I didn't even try it, although I was able to find information about other people doing it.

In the end, the option I chose was to edit my SSH authorized_keys file. This is probably considered a bit of a hack. The authorized_keys file has an option for restricting access to certain commands by adding `command=...` before the start of the key. I think this is how GitHub allows SSH access for git but not shell access, for example.

This has a downside as well. Because it's a restriction, if you have more than one SSH key in the authorized_keys, and you're coming from a machine with more than one of those keys, SSH is going to use the one that is the least restricted.

In my case, I generated a new key to connect from my work laptop to my Raspberry Pi dev box. I left the original key in place on the Raspberry Pi and added the new one with `command="tmux attach || tmux"` prepended. When I tried to `ssh` in, even when I specified the identity file that was restricted, SSH would use the original, unrestricted key (without `command=`, etc., in it), and I would get a normal shell, not a tmux session. I resolved this by removing all other authorized keys from the machine. I suppose I also could have added a different user with only that new and restricted authorized key or removed the other private key from my work laptop, but removing the other key seemed like the easiest solution at the time.

tmux Cheatsheet

Starting, Attaching and Detaching:

- `tmux` — start a new tmux session.
- `tmux attach` — attach to an existing session.
- `tmux attach -t mysession` — attach to a session named mysession.
- `tmux attach || tmux` — attach to a session if one exists; if not create a new one.
- Ctrl-b, d — detach from the active session.
- Ctrl-b, : — open the tmux command shell.
- Ctrl-b, ? — display tmux commands and key bindings.
- Ctrl-b, s — prompt to change sessions, selecting from a list.
- Ctrl-b, $ — rename the current session.

Adding and Navigating Windows:

- Ctrl-b, c — create a new window.
- Ctrl-b, n — activate the next window in the window list.
- Ctrl-b, p — activate the previous window in the window list.
- Ctrl-b, [0-9] — activate window number 0–9.
- Ctrl-b, ’ — prompt for a window number (for windows higher than 9).
- Ctrl-b, l — activate previously active window.

Adding and Navigating Panes:

- Ctrl-b, " — split current pane horizontally.
- Ctrl-b, % — split current pane vertically.
- Ctrl-b, arrows — move active pane to the adjacent pane in the direction of the arrow.
- Ctrl-b, o — activate the next pane in order.
- Ctrl-b, q — display each pane's index.
- Ctrl-b, ; — activate previously active pane.
- Ctrl-b, . — re-index the current window.

Resizing and Rearranging Panes:

- Ctrl-b, Ctrl-arrow — resize current pane one line by shrinking/expanding in the direction of the arrow.
- Ctrl-b, Alt-arrow — resize current pane five lines by shrinking/expanding in the direction of the arrow.
- Ctrl-b, { — swap the current pane with the previous pane.
- Ctrl-b, } — swap the current pane with the next pane.
- Ctrl-b, spacebar — rearrange all panes, cycling through even-horizontal, even-vertical, main-horizontal, main-vertical and tiled.
- Ctrl-b, Alt-[1-5] — rearrange all panes, using the numbered layout from the list above.
- Ctrl-b, ! — break the current pane out into a new window.

RETURN TO CONTENTS