Extracted from:

tmux 2

Productive Mouse-Free Development

The Pragmatic Bookshelf

Raleigh, North Carolina

The Pragmatic Programmers

Pragmatic exPress

tmux 2

Productive Mouse-Free Development

Brian P. Hogan

Edited by Susannah Davidson Pfalzer

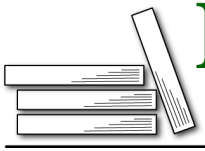
tmux 2

Productive Mouse-Free Development

Brian P. Hogan

The Pragmatic Bookshelf

Raleigh, North Carolina



Our Pragmatic books, screencasts, and audio books can help you and your team create better software and have more fun. Visit us at *https://pragprog.com*.

The team that produced this book includes:

Susannah Davidson Pfalzer (editor)
Nicole Abramowitz (copyedit)
Gilson Graphics (layout)
Janet Furlow (producer)

For sales, volume licensing, and support, please contact *support@pragprog.com*.

For international rights, please contact *rights@pragprog.com*.

Printed in the United States of America.
ISBN-13: 978-1-68050-221-3
Encoded using the finest acid-free high-entropy binary digits.
Book version: P1.0—November 2016

Detaching and Attaching Sessions

One of tmux's biggest advantages is that you can fire it up, start up programs or processes inside the tmux environment, and then leave it running in the background by "detaching" from the session.

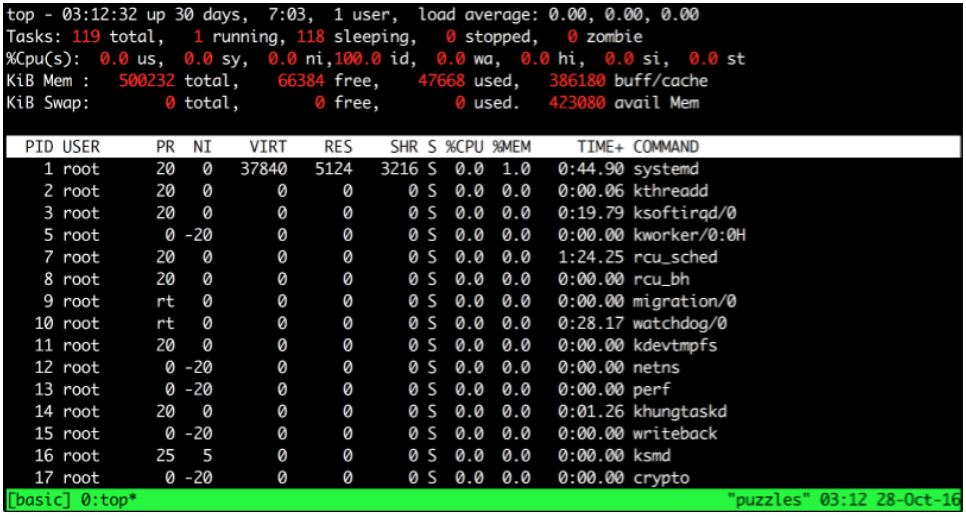
If you close a regular terminal session, all the programs you have running in that session are killed off. But when you detach from a tmux session, you're not actually closing tmux. Any programs you started up in that session will stay running. You can then "attach" to the session and pick up where you left off. To demonstrate, let's create a new named tmux session, start up a program, and detach from the session. First, create the session:

```
$ tmux new -s basic
```

Then, within the tmux session, start an application called `top`, which monitors our memory and CPU usage, like this:

```
$ top
```

You'll have something that looks like the following figure running in your terminal.



Now, detach from the tmux session by pressing PREFIX d. This returns you to your regular terminal prompt.

Now, let's look at how to get back in to that tmux session we left running. But before we do, close your terminal window.

Reattaching to Existing Sessions

We've set up a tmux session, fired up a program inside the session, detached from it, and closed our terminal session, but the tmux session is still chugging along, along with the top application we launched.

You can list existing tmux sessions using the command

```
$ tmux list-sessions
```

in a new terminal window. You can shorten the command to this:

```
$ tmux ls
```

The command shows that there's one session currently running:

```
basic: 1 windows (created Tue Aug 23 16:58:26 2016) [105x25]
```

To attach to the session, use the attach keyword. If you only have one session running, you can simply use

```
$ tmux attach
```

and you'll be attached to the session again. Things get more tricky if you have more than one session running. Detach from the basic session with PREFIX d.

Now create a new tmux session in the background using the command

```
$ tmux new -s second_session -d
```

This creates a new session, but doesn't attach to the session automatically.

Now list the sections, and you'll see two sessions running:

```
$ tmux ls
basic: 1 windows (created Tue Aug 23 16:58:26 2016) [105x25]
second_session: 1 windows (created Tue Aug 23 17:49:21 2016) [105x25]
```

You can attach to the session you want by using the -t flag, followed by the session name. Run the following command:

```
$ tmux attach -t second_session
```

This attaches you to the second_session tmux session. You can detach from this session just as you did previously, using PREFIX d, and then attach to a different session. In *Moving Between Sessions*, on page ?, you'll see some other ways to move between active sessions. But for now, let's remove the active sessions.

Killing Sessions

You can type exit within a session to destroy the session, but you can also kill off sessions with the kill-session command. It works just like tmux attach:

```
$ tmux kill-session -t basic
$ tmux kill-session -t second_session
```

This is useful for situations where a program in a session is hanging.

If you list the sessions again, you'll get this message:

```
$ tmux ls
no server running on /tmp/tmux-1002/default
```

Since there are no tmux sessions running, tmux itself isn't running, so it isn't able to handle the request.

Now that you know the basics of creating and working with sessions, let's look at how we can work with multiple programs within a single session.