Extracted from:

tmux

Productive Mouse-Free Development

This PDF file contains pages extracted from *tmux*, published by the Pragmatic Bookshelf. For more information or to purchase a paperback or PDF copy, please visit http://www.pragprog.com.

Note: This extract contains some colored text (particularly in code listing). This is available only in online versions of the books. The printed versions are black and white. Pagination might vary between the online and printer versions; the content is otherwise identical.

Copyright © 2010 The Pragmatic Programmers, LLC.

All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior consent of the publisher.

The Pragmatic Bookshelf

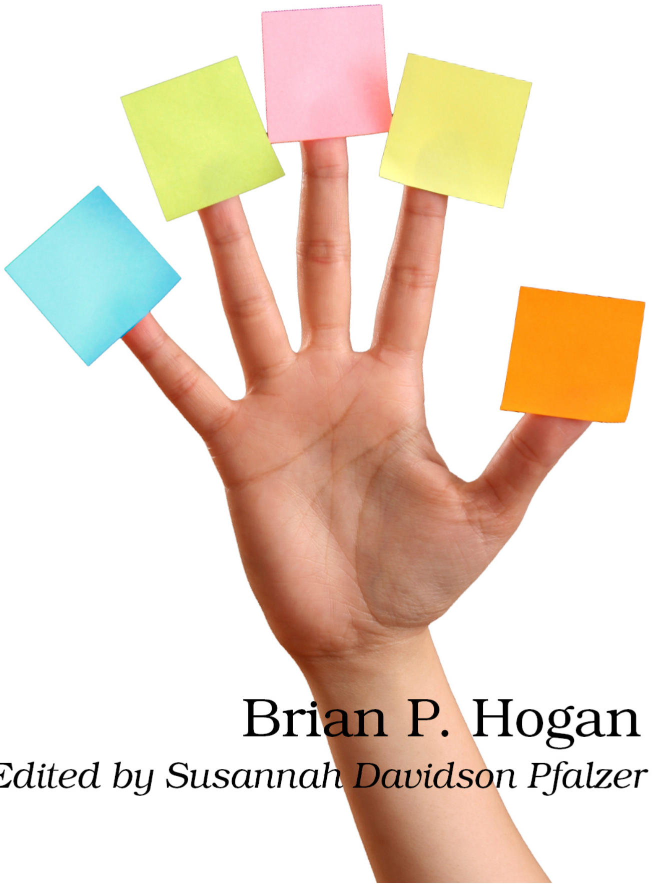
Dallas, Texas • Raleigh, North Carolina

The Pragmatic Programmers

Pragmatic exPress

tmux

Productive Mouse-Free Development



Brian P. Hogan

Edited by Susannah Davidson Pfalzer

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and The Pragmatic Programmers, LLC was aware of a trademark claim, the designations have been printed in initial capital letters or in all capitals. The Pragmatic Starter Kit, The Pragmatic Programmer, Pragmatic Programming, Pragmatic Bookshelf, PragProg and the linking *g* device are trademarks of The Pragmatic Programmers, LLC.

Every precaution was taken in the preparation of this book. However, the publisher assumes no responsibility for errors or omissions, or for damages that may result from the use of information (including program listings) contained herein.

Our Pragmatic courses, workshops, and other products can help you and your team create better software and have more fun. For more information, as well as the latest Pragmatic titles, please visit us at http://pragprog.com.

The team that produced this book includes:

Susannah Pfalzer (editor)
David J Kelly (typesetter)
Janet Furlow (producer)
Juliet Benda (rights)
Ellie Callahan (support)

Copyright © 2012 The Pragmatic Programmers, LLC.
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior consent of the publisher.

Printed in the United States of America.
ISBN-13: 978-1-934356-96-8
Encoded using the finest acid-free high-entropy binary digits.
Book version: P2.0—April 16, 2013

1.3 Detaching and Attaching Sessions

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

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

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

```
$ top
```

We now have something that looks like Figure 3, *The top command running in tmux*, on page 2 running in our terminal. We can now detach from the tmux session by pressing CTRL-b followed by d. This returns us to our regular terminal prompt. We'll learn how to get back into that session shortly, but first, let's talk about the command prefix.

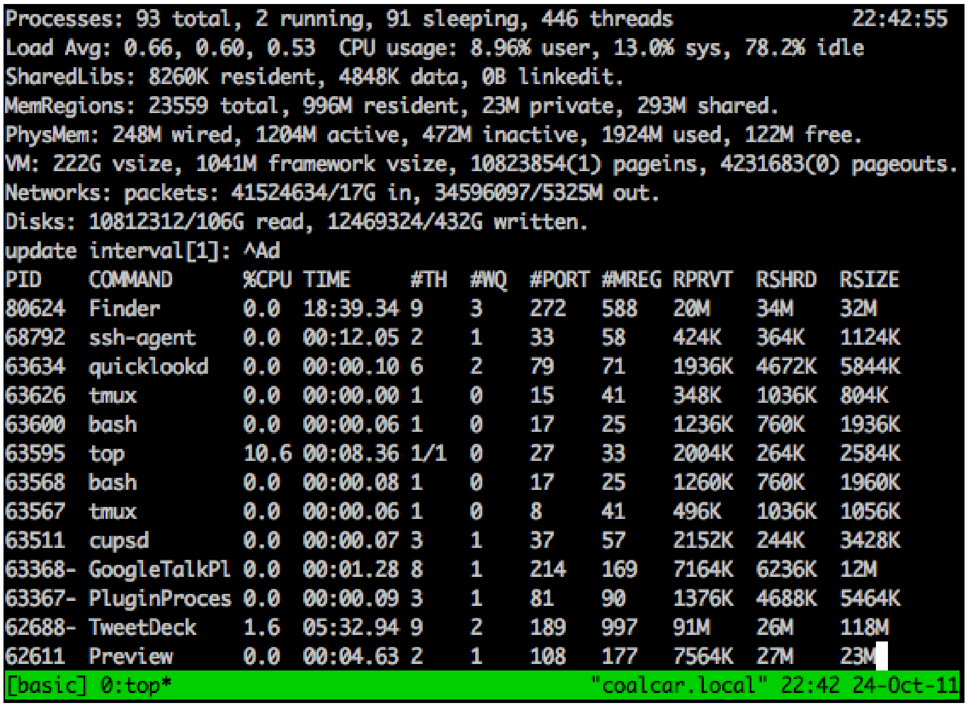
The Command Prefix

Since our programs are running inside tmux, we need a way to tell tmux that the command we're typing is for tmux and not for the underlying application. The CTRL-b combination does just that.

When we wanted to detach from our tmux session, we pressed CTRL-b, followed by d for "detach." We have to prefix each tmux command with this key combination, and it's important to note that we don't hold all these keys down together, but we instead first press CTRL-b simultaneously, release those keys, and then immediately press the key for the command we want to send to tmux.

Throughout the rest of this book, we'll simply use the notation PREFIX, followed by the shortcut key for our commands, like PREFIX d for detaching from a session. In Chapter 2, *Configuring tmux*, on page ?, we'll remap the prefix to an easier combination, but until then, we'll use the defaults.

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



```
Processes: 93 total, 2 running, 91 sleeping, 446 threads                22:42:55
Load Avg: 0.66, 0.60, 0.53  CPU usage: 8.96% user, 13.0% sys, 78.2% idle
SharedLibs: 8260K resident, 4848K data, 0B linkedit.
MemRegions: 23559 total, 996M resident, 23M private, 293M shared.
PhysMem: 248M wired, 1204M active, 472M inactive, 1924M used, 122M free.
VM: 222G vsize, 1041M framework vsize, 10823854(1) pageins, 4231683(0) pageouts.
Networks: packets: 41524634/17G in, 34596097/5325M out.
Disks: 10812312/106G read, 12469324/432G written.
update interval[1]: ^Ad
PID    COMMAND      %CPU TIME     #TH  #WQ  #PORT #MREG RPRVT  RSHRD  RSIZE
80624  Finder       0.0  18:39.34 9    3    272   588   20M    34M    32M
68792  ssh-agent    0.0  00:12.05 2    1    33    58    424K   364K   1124K
63634  quicklookd   0.0  00:00.10 6    2    79    71    1936K  4672K  5844K
63626  tmux         0.0  00:00.00 1    0    15    41    348K   1036K  804K
63600  bash         0.0  00:00.06 1    0    17    25    1236K  760K   1936K
63595  top          10.6 00:08.36 1/1  0    27    33    2004K  264K   2584K
63568  bash         0.0  00:00.08 1    0    17    25    1260K  760K   1960K
63567  tmux         0.0  00:00.06 1    0    8     41    496K   1036K  1056K
63511  cupsd        0.0  00:00.07 3    1    37    57    2152K  244K   3428K
63368- GoogleTalkPl 0.0  00:01.28 8    1    214   169   7164K  6236K  12M
63367- PluginProces 0.0  00:00.09 3    1    81    90    1376K  4688K  5464K
62688- TweetDeck    1.6  05:32.94 9    2    189   997   91M    26M    118M
62611  Preview      0.0  00:04.63 2    1    108   177   7564K  27M    23M
[basic] 0:top*                                     "coalcar.local" 22:42 24-Oct-11
```

Figure 3—The top command running in tmux

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.

We can list existing tmux sessions using the command

```
$ tmux list-sessions
```

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

```
$ tmux ls
```

The command shows that we have one session currently running:

```
basic: 1 windows (created Mon Jan 30 16:58:26 2012) [105x25]
```

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

```
$ tmux attach
```

and we'll be attached to the session again. Things get a little more tricky if we have more than one session running. Let's detach from the basic session with PREFIX d.

Now, if we create a new tmux instance in the background using the command

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

and list the sessions again, we'll see this:

```
$ tmux ls
basic: 1 windows (created Mon Jan 30 16:58:26 2012) [105x25]
second_session: 1 windows (created Mon Jan 30 17:49:21 2012) [105x25]
```

We can attach to the session we want by using the -t flag:

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

This puts us in the second_session tmux session. We can detach from this session just as we did previously, 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

We can type exit within a session to destroy the session, but we can also kill off sessions with the kill-session command.

```
$ 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 we list the sessions again, we'll get this message:

```
$ tmux ls
failed to connect to server
```

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.