

The
Pragmatic
Programmers

tmux 3

Productive
Mouse-Free
Development



Brian P. Hogan
edited by Tammy Coron

This extract shows the online version of this title, and may contain features (such as hyperlinks and colors) that are not available in the print version.

For more information, or to purchase a paperback or ebook copy, please visit <https://www.pragprog.com>.

Copyright © The Pragmatic Programmers, LLC.

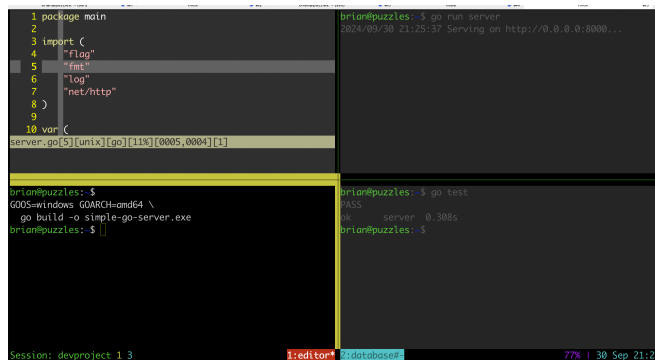
Preface

Your mouse is slowing you down.

When it was first introduced, the mouse created a new way for people to interact with computers. You can click, double-click, and even triple-click to open documents, switch windows, and select text. And thanks to trackpads, you can even swipe and use gestures to interact with your applications. The mouse, along with graphical interfaces, made computers just a little easier for the average person to use. But, there's a downside to the mouse, especially for programmers.

As you build software, you work with multiple programs throughout the course of your day. If you're a full-stack developer, you might have a database console, a local web server, and a text editor running at the same time. And you'll surely have to run your test suite and other commands. Switching between terminal windows or tabs with the mouse can slow you down. It may not seem like much, but moving your hand off of the keyboard's home row, placing it on the mouse, locating the pointer, and performing the task can eat up time and break your focus. It can also induce strain on your wrist, arm, or shoulder. That repetitive movement can cause some serious discomfort if you're not careful about how you hold that mouse.

Using tmux, you can create an environment like this right in a single terminal window, managed entirely without a mouse:



```
1 package main
2
3 import C
4     "flag"
5     "fmt"
6     "log"
7     "net/http"
8
9
10 var C
server.go[5][unix][go][11x](0005,0004)[1]

brian@puzzles: $ go run server
2024/09/30 21:25:37 Serving on http://0.0.0.0:8080...

brian@puzzles: $ go test
PASS
ok      server  0.308s
brian@puzzles: $

Session: devproject 1 3  1:editor  2:database  77% 1 30 Sep 21:25
```

Using tmux's windows, you can easily manage a text editor and a database console, and run local web servers and tests within a single environment. You can split tmux windows into sections, so multiple apps can run side-by-side. This means you can run a text-based browser, a text-based chat client, or your automated tests in the same window as your main editor.

Best of all, you can quickly move between these windows and panes using only the keyboard. Over time, the keystrokes you use to manage your environment will become second nature to you, which will greatly increase both your concentration and your productivity.

In this book, you'll configure, use, and customize tmux. You'll manage multiple programs simultaneously, write scripts to create custom environments, and use tmux to work remotely with others. With tmux, you can create a work environment that keeps almost everything you need at your fingertips.

What Is tmux?

tmux is a *terminal multiplexer*. It lets you use a single environment to launch multiple terminals, or windows, each running its own process or program. For example, you can launch tmux and load up the Vim text editor. You can then create a new window, load up a database console, and switch back and forth between these programs, all within a single session.

If you use a modern operating system and a terminal that has tabs, this doesn't sound like anything new. But running multiple programs simultaneously is only one of tmux's features. You can divide your terminal windows into horizontal or vertical panes, which means you can run two or more programs on the same screen side by side. You can ensure these new windows or panes always open in the directory you want. You can move them from window to window and change their layout. And you can do it all without using the mouse.

You can also *detach* from a session, meaning you can leave your environment running in the background. If you've used GNU-Screen before, you're familiar with this feature. In many ways, tmux is like GNU-Screen with a lot of extra features and a more approachable configuration system. Since tmux uses a client-server model, you can control windows and panes from a central location or even jump between multiple sessions from a single terminal window. This client-server model also lets you create scripts and interact with tmux from other windows or applications.

Over the course of this book, you'll use all of these features and more.

Who Should Read This Book

Whether you're a system administrator or a software developer who spends a good part of your time using the terminal and command-line tools, this book aims to help you work faster.

If you're a software developer, you'll use tmux to build a development environment that can make working with multiple terminal sessions a breeze. And if you're already comfortable using Vim or Emacs, you'll see how tmux can accelerate your workflow even more.

If you're a system administrator or a developer who spends time working with remote servers, you'll be interested in how you can leverage tmux to create a persistent dashboard for managing or monitoring servers.

What's in This Book

This book will help you incorporate tmux into your work by taking you through its basic features and showing you how you can apply them to everyday situations.

In [Chapter 1, Learning the Basics, on page ?](#), you'll use tmux's basic features. You'll create sessions, panes, and windows and get comfortable with basic navigation.

In [Chapter 2, Configuring tmux, on page ?](#), you'll redefine many of the default keybindings and customize how tmux looks as you build your own configuration file from scratch.

In [Chapter 3, Scripting Customized tmux Environments, on page ?](#), you'll script a custom development environment using the command-line interface, configuration files, and the tmuxinator program.

After that, you'll work with text in [Chapter 4, Working With Text and Buffers, on page ?](#). You'll use the keyboard to move backward through the buffer, select and copy text, and work with multiple paste buffers. You'll also integrate tmux with your system clipboard.

Next, in [Chapter 5, Pair Programming with tmux, on page ?](#), you'll set up tmux so that you and a coworker can work together on the same codebase from different computers.

Finally, [Chapter 6, Workflows, on page ?](#) introduces more advanced ways to make you more productive. You'll manage windows, panes, and sessions; create popup windows; build custom menus; and explore tmux plugins.

Changes in the Third Edition

This new edition has some notable changes from the second edition. `tmux` 3 introduced several backward-incompatible changes to `tmux`'s configuration syntax that this edition addresses. It also introduced some new features. Here's what's changed in the third edition:

- All examples in this edition require at least `tmux` 3.4. Previous versions use different configuration syntax for some options or have different behavior.
- This book covers Windows 10 and 11 installation using the much more streamlined Windows Subsystem for Linux.
- [Chapter 2, Configuring `tmux`, on page ?](#) now reflects the new configuration syntax introduced in `tmux` 3.
- [Chapter 5, Pair Programming with `tmux`, on page ?](#) has an updated method for remote pairing, as the previous method is no longer compatible with `tmux` 3. You'll also find updated instructions for sharing sockets between users.
- [Chapter 6, Workflows, on page ?](#) contains updates to existing workflows and introduces hooks, popup windows, and configurable menus.

In addition, you'll find smaller changes throughout the book to clarify explanations and add more context, all based on previous reader feedback.

What You'll Need

To use `tmux`, you'll need a computer that runs macOS, Windows 10 or 11 with WSL and Bash support or a flavor of Unix or Linux. Unfortunately, `tmux` doesn't run under the regular Windows Command Prompt or Powershell, but it will run great under WSL. You can also explore `tmux` on a virtual machine, VPS, cloud, or shared hosting environment running Linux or FreeBSD.

You should also have a good grasp of using command-line tools on a macOS, Linux, or Unix-like system. You'll use the Bash shell in this book, and being comfortable with creating directories and text files, as well as some basic scripting will help you move more quickly through the examples.

While not required, some experience with text editors such as Vim or Emacs might be helpful. `tmux` works in much the same way, and it has some predefined keyboard shortcuts that you may find familiar if you use one of these text editors.

Conventions

You drive tmux with the keyboard; you'll encounter many keyboard shortcuts throughout the book. Since tmux supports both lowercase and uppercase keyboard shortcuts, it may sometimes be unclear which key the book is referencing.

These are the conventions you'll find throughout the book:

- **CTRL-b** means "press the **CTRL** and **b** keys simultaneously."
- **CTRL-R** means you'll press the **CTRL** and **r** keys simultaneously, but you'll need to use the **SHIFT** key to produce the capital "R." I won't explicitly show the **SHIFT** key in any of these keystrokes.
- **CTRL-b d** means "press the **CTRL** and **b** keys simultaneously, then release, and then press **d**." In [Chapter 1, Learning the Basics, on page ?](#), you'll learn about the *command prefix*, which will use this notation, but shortened to **PREFIX d**.
- You'll find many terminal commands throughout the book, like this:

```
$ tmux new-session
```

The dollar sign represents the prompt from the Bash shell session. You won't need to type it when you type the command. It just denotes that this is a command you should type.

- Finally, as you'll see in [Chapter 2, Configuring tmux, on page ?](#), you can configure tmux with a configuration file in your home directory called `.tmux.conf`. Filenames starting with a period don't show up in directory listings on most systems or text editors by default. Code listings in this book have a header that points to the file in the book's source code download, like this:

```
config/tmux.conf
# Set the prefix from C-b to C-a
set -g prefix C-a
```

To make it easier for you to find the file in the source code download, I've named the example file `tmux.conf`, without the leading period. The headers above the code listing reference that file.

Online Resources

The book's website¹ has links to submit errata for the book as well as the source code for the configuration files and scripts you'll use in this book. If you're reading the electronic copy of this book, you can click the filename above code excerpts to view the file's full source code. If you're reading the electronic copy of this book, you can click the filename above code excerpts to view the file's full source code. I also host a companion website for the book.²

Working with tmux has made me much more productive over the years, and the last two editions of this book helped thousands of developers put tmux to use for them. I'm excited to share this third edition with you. Let's get started by installing tmux and working with its basic features.

-
1. <https://pragprog.com/titles/bhtmux3/tmux-3/>
 2. <https://tmuxbook.com>