The Pragmatic Programmers

tmux 3

Productive Mouse-Free Development

Brian P. Hogan

edited by Tammy Coron

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 everything 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.

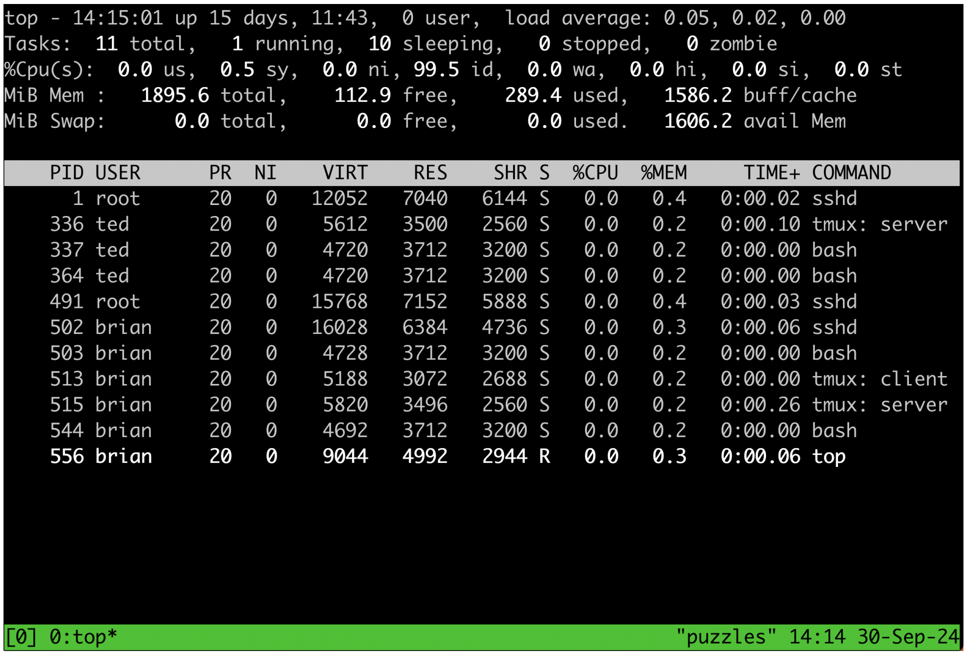
To demonstrate this, create a new named tmux session, start up a program, and detach from the session. First, create the session:

```
$ 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.



```
top - 14:15:01 up 15 days, 11:43,  0 user,  load average: 0.05, 0.02, 0.00
Tasks:  11 total,   1 running,  10 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.0 us,  0.5 sy,  0.0 ni, 99.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   1895.6 total,    112.9 free,    289.4 used,   1586.2 buff/cache
MiB Swap:      0.0 total,      0.0 free,      0.0 used.   1606.2 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
      1 root      20   0   12052   7040   6144 S   0.0   0.4   0:00.02 sshd
    336 ted       20   0    5612   3500   2560 S   0.0   0.2   0:00.10 tmux: server
    337 ted       20   0    4720   3712   3200 S   0.0   0.2   0:00.00 bash
    364 ted       20   0    4720   3712   3200 S   0.0   0.2   0:00.00 bash
    491 root      20   0   15768   7152   5888 S   0.0   0.4   0:00.03 sshd
    502 brian     20   0   16028   6384   4736 S   0.0   0.3   0:00.06 sshd
    503 brian     20   0    4728   3712   3200 S   0.0   0.2   0:00.00 bash
    513 brian     20   0    5188   3072   2688 S   0.0   0.2   0:00.00 tmux: client
    515 brian     20   0    5820   3496   2560 S   0.0   0.2   0:00.26 tmux: server
    544 brian     20   0    4692   3712   3200 S   0.0   0.2   0:00.00 bash
    556 brian     20   0    9044   4992   2944 R   0.0   0.3   0:00.06 top

[0] 0:top*                                              "puzzles" 14:14 30-Sep-24
```

Now, detach from the tmux session by pressing PREFIX d. This returns you to your regular terminal prompt. The tmux session is still running, and the top program is running inside of that session.

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

Reattaching to Existing Sessions

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

You can list existing tmux sessions using the following command:

```
$ tmux list-sessions
```

You can shorten the command to this:

```
$ tmux ls
```

Open a new terminal window and list the sessions. The command's output shows that there's one session currently running:

```
basic: 1 windows (created Fri Jun 14 06:34:45 2024)
```

The output of the command shows the tmux session name you provided when you created the session, and it shows the time you created it.

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

```
$ tmux attach
```

Execute that and you'll be attached to the session again.

You can have more than one tmux session running, and you can switch between them. Detach from the basic session with PREFIX d.

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

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

This command creates a new session, but the -d switch tells tmux not to attach to the session automatically.

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

```
$ tmux ls
basic: 1 windows (created Fri Jun 14 06:34:45 2024)
```

```
second_session: 1 windows (created Fri Jun 14 06:38:37 2024)
```

You can attach to the session you want by using tmux attach with 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.

Now let's remove the active sessions.

Killing Sessions

There are two ways to end a tmux session. First, you can attach to the session, stop all of the programs within the session, and then type exit within a session. You can also kill off sessions with the kill-session command. It works just like tmux attach.

Run the following commands to end the basic and second_session sessions you created:

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

If you list the sessions again, you'll get a message telling you tmux isn't running:

```
$ 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.

The tmux kill-session command is the equivalent of closing a terminal. All of the processes inside are killed. This command is incredibly useful for situations where a program in a session becomes unresponsive.

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