WeGet letters

by Michael W Lucas

Dear Most Useless Advice Columnist in Technology (or Anywhere),

In all of open source, kernel developers are the elite. They get to implement the really cool stuff and invent nifty new features, like ZFS and buffer caches and memory protections. Any advice on how I could become one of them?

I've read your column several times, and honesty demands that I inform you that I'm going to maximize my chances of success by listening carefully to everything you suggest, then doing the exact opposite.

—Novice But Not Naïve

Dear NBNN̈,

"Working in computing isn't enough for me. I want my failures to be truly inexplicable!"

Very well.

Many people fantasize that kernel developers are programming elite. John Baldwin, of repeated FreeBSD Core Team fame as well as the editorial board chair of this very Journal, went from writing documentation straight into kernel development. John has been unfortunate enough to know me for decades so I can confidently assure you that not only is he not an elite but the remarkable, incriminating, and noteworthy things about him have absolutely nothing to do with programming. Kernel developers must achieve a minimum competence, yes, but beyond a couple rules, there's nothing special about kernel code. Imagining that kernel programmers are an elite will sabotage you before you start, so I strongly encourage it.

Many people fantasize that kernel developers are programming elite.

If you insist on proceeding, though, if you demand you be allowed to weave yourself a chrysalis and transform into a kernel developer like a panic-prone memory-dumping file-corrupting butterfly, immediately separate your dreams from your goals. A goal is something actionable that is completely within your con-

trol to achieve. Accomplishing a dream requires other people to intervene on your behalf. Going out for a dinner date with that attractive person? Totally a dream. Asking that attractive person out for a dinner date, and when they remind you that you are inherently unlovable and should leave them the heck alone instead of stalking them like the creepy hero of a so-called "romance?" An absolutely achievable goal!

You cannot control other people. Work on goals. Never on dreams.

What goals can you set that would guide you become a kernel developer?

Start by reading the documentation.

There are books like *The Design and Implementation of the FreeBSD Operating System, FreeBSD Device Drivers*, and *Designing BSD Rootkits*, which add an interesting twist to learning how the kernel works. *The FreeBSD Developers Handbook* is freely available. Fill your brain. Do the exercises. If something is beyond you, well, people have written articles and books discussing it.

Most transformations end hard right here because humans cherish cozy comfy non-actionable dreams and aren't as fond of ugly hard goals.

Note that I didn't say "ask other people how to start learning about the kernel." If you haunt the mailing lists, the forums, or the Internet's sketchier discussion boards you'll occasionally see people asking for help in learning to program the kernel. You might think that these people are looking for the list above, but the answer I give here appears on the most cursory search and comes across as *please hold my hand*. Do caterpillars ask for help weaving their chrysalis? No! They sweat and struggle so they can slither into their cramped cocoons and simmer into transcendence. You must do the work. Most transformations end hard right here because humans cherish cozy comfy non-actionable dreams and aren't as fond of ugly hard goals.

As with any other part of contributing to an open source project, you need to find a tiny piece to work on. Start with bugs. Problem reports are a gold mine for the aspiring kernel developer. As you look through possible projects, you must again separate goals from dreams. "Solve several panic bugs and get my fixes committed" is a dream. It requires that established kernel developers notice your fixes and choose to incorporate them. "Solve one reported kernel panic this month" isn't exactly a goal, because you can't guarantee that you will be able to solve it. "Spend ten hours this month working on a reported kernel panic, without taking breaks every three minutes to gripe on social media, in the work chat, or to my pet who has to put up with me even though I'm inherently unlovable." There—*that's* a goal! Complete enough of those goals and you'll develop the skill of kernel programming.

The hard part of working in the kernel, though?

Other people.

Suppose you develop patches to fix reported problems and attach them to the bug. You can't make a project member notice your work. If they notice your work, you can't make them take your patch as-is. A project member might use your patch as inspiration or a proof-of-concept and create a wholly different patch for reasons you hadn't even thought of. Making people notice you is a dream. Making yourself dang hard to ignore by submitting a whole series of quality patches is absolutely a goal.

An interesting thing about how caterpillars become butterflies. They don't. We see the caterpillar crawl into its cocoon and the butterfly emerge, so we assume that there's been a transformation when the harsh reality is, the caterpillar's chrysalis? It's a coffin. The caterpillar crawls in and melts to goo surrounding a tiny lump that's basically a self-assembling butterfly kit. The butterfly's first meal is 100% Grade A caterpillar sludge. When you submit your twentieth patch and still it feels like nobody cares, be grateful that you haven't transformed yourself into literal physical muck. Mental muck is less noticeable.

Suppose your patches get picked up? What then?

Again, it's people.

In that glorious aeon when the Sacred and Penultimately Blessed Computer Science Research Group distributed primordial BSD, a single person could achieve a good understanding of Unix. A complete install took only a few megabytes. And yes, that included the compiler and source code, what part of "complete install" was unclear? College students were expected to read and understand the code.

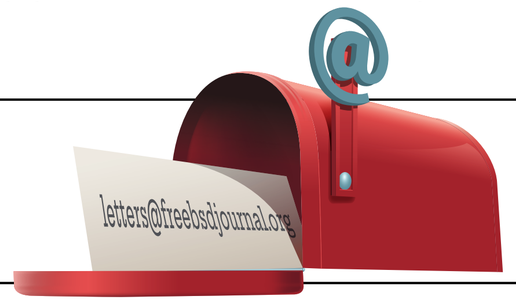
Today? By the time you finish reading the base system source code, it's changed and you get to start over. Becoming a "kernel developer" is almost impossible. You might, at best, become a trusted developer responsible for one tiny slice of the kernel. Performing maintenance will require interacting with other parts of the kernel, which means discussing your changes with the people responsible for those parts. Working in the kernel is no different than programming in userland, except people believe you've achieved a certain minimal competence.

If you achieve your dream and become a full-on kernel developer, you'll discover that people are not a problem.

If you achieve your dream and become a full-on kernel developer, you'll discover that people are not a problem. They are the problem. Every change you make will upset someone. Users and non-kernel programmers will have this weird idea that you're the elite, that you know what you're doing, that you are less baffled than them.

As you've declared an intent to not merely ignore but reverse my thoughts, let me summarize: becoming a kernel programmer is the one true path to happiness and I wish you well. Dream on!

Have a question for Michael?
Send it to letters@freebsdjournal.org



MICHAEL W LUCAS Unlike esteemed *FreeBSD Journal* Editorial Chair and elite kernel developer John Baldwin, Michael W. Lucas remained in documentation. His latest book is *Run Your Own Mail Server*, which uses FreeBSD as a reference platform. Learn more at https://mwl.io.