WeGet letters

by Michael W Lucas

Greetings, Letters Columnist!

Before you treat my letter like those IRS notices and fling it into /dev/null, please note that — if I timed this correctly — a delivery driver will be knocking on your door any minute now with a gallon of handmade gelato, all in the hopes of evoking a useful answer from you.

What does your average sysadmin need to know about embedded systems?

—Mostly Expecting Derision, Sadly

Dear BAD,

A valiant effort. Your email was delayed by greylisting, however, so it arrived after the delivery truck. I have no idea where you found pickled salmon and durian gelato, but I commend the effort you put into achieving a new height of appalling. Well done. You're a natural sysadmin.

What does a sysadmin need to know about embedded systems? An embedded system is just a computer. What makes embedded systems different from regular hosts? Absolutely nothing, except you must do everything correctly.

You can tune a regular host. When /var/log/ on your web server consumes an inordinate amount of disk, you can adjust the log rotation conditions or tweak what you log. System tuning in ossified enterprises with rigorous change control processes enforced by an authoritarian goon whose soul was crushed by discovering he was born too late to offer his services as an unpaid intern to Gaius Julius Caesar Germanicus is lax compared to changing proper embedded systems. Changing an embedded system requires a system update, and do you know how often these devices get reflashed? That's right, *never*. Maybe you let your streaming media device reboot when it downloads the newest firmware, but when was the last time you updated your fridge? I avoid this class of problem by walking into the appliance store and saying, "You can sell me anything so long as it includes a complete lack of Internet connectivity." They promptly lead me to

What does a sysadmin need to know about embedded systems? An embedded system is just a computer.

the 20th Century Room. I leave with a botnet-resistant dishwasher. Expensive, but worth it. If you're even asking this question, you have chosen poorly.

An embedded system configured less than perfectly will stop working. Maybe that's "stop working correctly," maybe it starts again once you restart it, maybe it full-on bricks itself. Whatever. It's fixable, of course. Many embedded systems are designed for repair by replacement. A few have terminals where you can check for full disks or wedged processes but hide it. Disk drives all have firmware, and many manufacturers offer arcane tricks to access the terminal. Run a few wires from a DB9 to select pins on a disk drive, type the secret code, and poof — you have hard-wired access directly into the drive's feeble little brain. There's even a menu system. Once you exclude the treachery, the menu consists primarily of lies, but it exists.

Everything is embedded systems. All the way down.

Your RAID controller? It runs an embedded operating system that lets it emulate a wholly different RAID controller from the 1980s because sysadmins insist that they be allowed to set a "stripe size" that doesn't map to anything inside the actual storage medium. Hit F1 or DELETE during boot, and you'll get into the mainboard's embedded system. The "BIOS Log" is full, by the way. It's been full for years. Nothing useful has been logged since your predecessor was hired. Everything in your computer is a separate embedded system supporting your non-embedded system. Your "bare metal" server isn't bare metal, because "bare metal" does not exist.

Everything is embedded systems. All the way down.

I've griped about virtualization many times, but everything is virtual and the whole world merits that same scorn.

Why put an operating system in your hard drive, mainboard, or USB chipset? Hardware changes are expensive and require talking to the manufacturer. Software changes are inexpensive and can be jammed into existing hardware. We weasel around hardware bugs with differently buggy software. Yes, releasing correct hardware would be better, but the people in charge assure us that's not possible, so reality once again demonstrates that what seemed like a good idea was actually a horrible idea. The embedded system saves the manufacturer money but increases pain.

Don't give me that look. All computers increase pain.

The purpose of a system is what it does, and a computer does pain.

We cope with the pain by adding more computers.

The first Rule of System Administration declares "computers were a mistake," and examining the mental health impact of the teetering cataclysm of systems inside your computer proves it, let alone the systems in your car. Don't get me started on your car. Requiring everyone in the United States to borrow, purchase, or otherwise "acquire" a multi-ton kinetic energy weapon was bad enough, but add in separate computers for the brakes and the accelerator and the climate control, and congratulations! You've built a Rolling Debacle. When your drive-by-wire steering catches a SIGABRT the only question is who

gets debacled, you or bystanders. I would say "innocent bystanders," but nowadays everybody uses a computer, so nobody is innocent. Even folks who want to remain unsoiled are compelled to use a website or an app because the guilty make themselves feel better by dragging everyone down to their sewer. ("Come to my party! It'll be great! Never mind the floaters!")

Can you even find a computer without a gratuitous embedded system somewhere? Certainly. It's called an abacus. My abacus has a series of five-bit bytes and a separate register with two-bit bytes. Quite sophisticated. The operating system is implemented in hand position. Best of all, nobody has written a TCP/IP stack for the abacus.

The bad news is that embedded systems have become more accessible than ever. Inexpensive, low-power systems are available from your least abhorred retailer, enabling you to build tiny systems to control your whole home if you wish. If you want a custom remote control for the big door on the shed where you store your multi-ton kinetic energy weapon, you can do so for the low cost of a hundred bucks of hardware, several man-months of effort, and your will to live.

So, what should you know about embedded systems? If you're stuck with one and it breaks, pretend you're an idiot and call tech support. Otherwise, you'll delve into the system and discover just what atrocities the designer committed in the name of system stability. That won't go anywhere pleasant.

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

MICHAEL W LUCAS is the author of *Absolute FreeBSD*, *Dear Abyss*, *SSH Mastery*, and more. The new edition of *Networking for System Administrators* has recently escaped.
https://mwl.io