/dev/random
Layers

ROBERT G. FERRELL

Robert G. Ferrell, author of *The Tol Chronicles*, spends most of his time writing humor, fantasy, and science fiction. rgferrell@gmail.com

Once upon a time, there was only one layer: the operating system. It was your first and best means of exchanging ones and zeroes with the processor, the mystical heart of your computer. You wrote code, compiled it or fed it to an interpreter, and something interesting usually happened. Well, my code *always* made something interesting happen, but my threshold for interesting includes power cycling and printers with a pronounced tendency to spit out page after page of nonsense. I also evinced a preternatural knack for triggering crash dumps that literally caused the machine, not to mention any clued-in onlookers, to shudder.

As I've said (too) many times, my digital heyday came during a previous geological era as reckoned in the accelerated chronology of computing. The code I hammered out looks primitive and ragged now, much like my wardrobe and finances. Those were the days when shareware came on floppies obtained at music-turned-discount-software stores in the mall, before every new computer game release required the latest supercharged video card to run at better than two frames per second. Those were the days when hacking was, at worst, criminal mischief—not supervillainy.

The first incursion of layers was the graphic user interface, intended to make navigating the operating system a little easier for people who hadn't the patience to commit dozens of program options and arguments to memory. This was in no way mandatory: those of us who liked the cryptic nature of the command line beast could still accomplish whatever we needed without getting our fingers GUI. But then, gradually, virtualization and emulation and compartmentalization began to creep into our systems like vampires seeking refuge from a clear summer's afternoon. After a while it was no longer at all apparent what floor of the computational skyscraper you were working on.

I retired some years ago from looking over the shoulders of system administrators to verify that they'd implemented at least the minimal security measures mandated by government standards. Toward the end of that intellect-numbing career, virtualization was already complicating lives. Did certain security settings apply only to the underlying operating system, for example, or did they need to be duplicated for every virtual machine instance? In those situations where one operating system had significantly different mandated security parameters from another, but both were instantiated in virtual machines on the same box, which one's security settings took precedence?

The virtualization rabbit hole now goes much deeper from those comparatively halcyon days. Today the concept of a single operating system directly supporting applications in the enterprise seems as quaint and whimsical as a racoon coat in a rumble seat. Not all of these layers are distinct operating system images; it's true. Some of them, like Docker layers, are just topological metaphors for processes being run as a suite within a lightweight container. I think I just got a charley horse in my frontal lobe from typing "topological metaphors." Ow.

Anyway, this layering mania got me to wondering: just what are these people running from? What is it about the base operating system that makes them so uncomfortable? Are they embarrassed by the belief that people regard them as unsophisticated because they only have a couple of layers going? Or is it just that they were exposed to the OSI model during their formative years and now feel that multiple layers are necessary for things to work?

Speaking of network models, it seems to me that we'll need to reinforce the TCP/IP stack in order to bear the weight of all these new layers. Maybe stick some rebar in there or something. Come to think of it, perhaps it's also time to establish an entirely new nomenclature that reflects today's puff pastry networking reality. After all, continual change for change's sake is what technical advancement is really all about, right? No novelty, no progress.

We'll start at the bottom, because that's where I'm most at home. The current lowest level is the Physical layer (OSI Layer 1), so-called because it deals with wires and adapters and those little cylindrical doodads on some cables that you don't know what they do—physical objects, in other words. I propose we rename this the *Fiddly Bits* layer, since one out of one columnist surveyed declared this to be a lot more descriptive and accurate.

Next up is the Data link layer (OSI Layer 2). Data link sounds like some rural ISP that set up shop using old satellite dishes and routers they dug out of the dumpster behind Fry's. I think a better name for something that connects data paths is the *Drawbridge* layer. Above the Drawbridge we come to the Network layer (OSI Layer 3). Here the bits really hit the fan, what with packets and frames buzzing around like flies over garbage. For that reason, I think of it as the *Landfill* layer.

The Transport layer (OSI Layer 4) is where those bits get packaged and shipped off to market, so I call it the *Loading Dock* layer. The Session layer (OSI Layer 5) is mostly concerned with keeping lines of communication open, so we'll think of it as the *Switchboard* layer. Layer 6, the OSI Presentation layer, is where one format gets converted to another; I'll call this the *Thesaurus* layer. Finally, there is the Application layer (OSI Layer 7). This is sort of a catchall area for everything else that needs to happen to make software and user care about one another, so to me it is the *Kitchen Drawer* layer.

There you have the layers of the RGF model: Fiddly Bits, Drawbridge, Landfill, Loading Dock, Switchboard, Thesaurus, and Kitchen Drawer. The old "All People Seem to Need Data Processing" mnemonic doesn't work any longer, admittedly, but at least these are layer names that evoke actual mental images, not those sterile engineering labels your brain has to massage into real language before they mean anything to you. I doubt my terminology will make it into an RFC, unless it's an April Fool's submission, but that's not my concern. I'm just the idea guy.

"Layer," incidentally, can also refer to a hen that actively produces eggs. Eggs, like operating systems, have shells which both protect and provide access to the underlying contents. Computer science and animal husbandry: working hand in, um, talon for a better tomorrow.

Cluck().