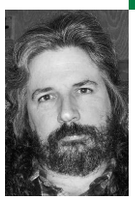
ROBERT G. FERRELL

/dev/random



Robert G. Ferrell is a chronically underemployed information security geek who enjoys surfing (the Internet), sashimi (it makes great bait), and long walks (to the coffee machine and back).

rgferrell@gmail.com

OVER TOO MUCH FINE BELGIAN ALE one night deep in my largely fictitious past, I decided to create my own file system. It wouldn't be anything fancy, I thought, just create a good, basic, reliable workhorse for the OS I would never get around to writing, either. I wanted to call it (the operating system) OreOS because I had the munchies at the time. Copyright issues seemed inevitable, however, so, to avoid legal entanglements and strike a blow for truth in advertising, I next named it ZenOS. An operating system that doesn't really operate anything is, after all, something of a paradox. The logical name for this new file system would therefore have been ZFS, but, sadly, Sun had nabbed that one out from under me. I finally settled on RGFS, on the supposition that no one else would want to name a file system using my initials.

Shortly thereafter I discovered that the German company Actum had a product called *Zenos*, although any durn fool can see this isn't the same thing as ZenOS. Their site was also in German, unfortunately, and since all the Deutsche I know I learned from *Hogan's Heroes*, I wasn't able to make heads or tails of what Zenos was supposed to do. In the interest of clarity and because I could already feel the hot, weaselly breath of intellectual property lawyers wilting my 70% post-consumer fiber shirt collar, I decided that discretion was the better, or at least less litigious, part of valor.

Most vendors solve this branding conundrum by just tacking "nix" onto the first syllable of some company-related word in the established evolutionary tradition of organisms mined from the Mother Code. My OS wasn't likely to resemble UNIX in any way, however, owing to the fact that I really don't understand how operating systems work, so I decided to eschew this nomenclatural methodology for something more novel and fitting. Besides, who wants to run something called "Robnix" (which sounds like a bad Web comic or a direct-to-DVD family action movie, an oxymoron if ever I've seen one)?

"What," I therefore inquired of myself, "would one reasonably call an operating system like mine—one still in the, um, *formative* stages?" The answer came out of the blue, as epiphanies are

wont to, like unto the arrival of a sinus headache shortly after stumbling into a field of blooming goldenrod: EmbryOS! As a name this had it all: It was descriptive, topical, vaguely biological, and relatively easy to print diagonally across a box in large, colorful letters. I had obviously missed my Madison Avenue calling long ago, at an early stage (probably between pupa and larva). Happily I sat back and envisioned the Web 3.0 HD 1080i 64-bit multimedia extravaganza (I figured 2.0 would be passé by the time I get around to an actual release), the brochures, the full-page ads in slick magazines, the IPO, and my eventual six-bedroom cabaña in the Bahamas, complete with fire-engine-red Porsche Carrera GT in the garage. Must . . . not . . . hyperventilate.

I put at least two full beers' worth of thought into the design of RGFS, and I came to the conclusion that journaling is just *so* late '90s. It was time to move file system architecture into the twenty-first century, I decided, so RGFS should keep track of changes not in a journal, but in a blog. That way, the system itself can add comments and attach still images, Flash presentations (what do you call Flash contained in flash memory? HyperFlash?) and even YouTube movies or streaming multimedia to the usual boring ol' file information. I might need to up the block size to, say, a terabyte or so to accommodate this new architecture, and that could perhaps slow down seek times a wee bit, but progress requires sacrifice, right?

File Blogging, or Flogging, might well be the file system wave of the future. It will render slack space obsolete, since every last bit will now be at a premium. Entire drives will need to be dedicated, not to the actual primary data itself, but to the metadata attached to it. We might even do away with the primary data altogether. RAID protocols will have to be adjusted accordingly, as well. Heck, let's just drop the RAID nonsense and ensure data redundancy using BitTorrent. Every storage device in the network neighborhood can host a stripe or mirror partition of every other one. SAN will now stand for *Symbiotic Area Network*.

Once computers are free—nay, required—to keep their own flogs, who knows what interesting and useful intel we can gain from plumbing their innermost ruminations? Will the dual-core processors consider themselves superior to their monolithic brethren? Will those employing LDAP have DNS envy? Will they list their favorite bands and complain endlessly about how we don't understand them?

While I'm on a roll, innovation-wise, I may as well implement some other ideas I've been kicking around, such as spanning the superblock across several volumes and encrypting the magic number. (Speaking of which, I'm thinking of incorporating a separate hardware data path just for passing around file system labels. I'll call it the Magic Bus.) The whole inode thing is hopelessly old-fashioned, too. I'm updating mine to iN0d3z and giving them their own MySpace page. Any machine that wants to add content to a file system can download them there.

EmbryOS/RGFS will of course be Open Source. In fact, I'm going to take this one step further and declare it to be Pro-Am Invitational Source, which means that I invite any programmer, be he or she professional or hobbyist, actually to create the source *for* me in the first place. You'll receive full credit for your input, naturally, in the End User License Agreement just below "EXCLUSION OF INCIDENTAL, CONSEQUENTIAL AND CERTAIN OTHER DAMAGES."

I ask for help not because I'm lazy and incompetent, but merely to protect the consumer. If I have to do the coding myself it may end up a little wonky. Inline-embedding GWBasic in Perl is just so exacting . . .

Riddle me this: Why is coding an operating system like being the parent of a young child on Christmas Eve?

Some assembly required.