WeGet letters

by Michael W Lucas

Dear Worst Columnist in This Journal,

My company has rack upon rack of storage servers. When I started as a sysadmin, nine-gigabyte drives were common. Now, each drive is multiple terabytes, and we're building arrays that aren't just petabytes but exabytes. We're building a data center for multiple zettabytes. What can any company be doing with all this storage?

—It's not bootleg movies, I checked

Dearest Bootleg,

That really is the question, isn't it? We have vast amounts of data storage capacity, and yet a measurable fraction of the world's manufacturing capacity is dedicated to producing more. We have entire container ships full of SSDs adrift in the Pacific Ocean, eagerly awaiting that glorious moment when they finally get to dock and offload all that blank storage. Organizations like yours order disks by the pallet. What can anyone do that generates so much data that they need yawning chasms of storage?

Unless you're working in exciting big data fields like bioinformatics or ripping holes in the universe at the Large Hadron Collider in the hope that your favorite incarnation of The Doctor will show up and tell you to stop, most of those petabytes are either data that you shouldn't have, obsolete data, or data that nobody will take responsibility for throwing away.

Organizations have a horrible habit of keeping every scrap of data that they get, even when possession of that data poses an appalling risk to the organization's health or existence. How many data breaches have you seen where a company leaked, say, Social Security numbers or credit card numbers or biological analyses of nose hair samples, and you immediately asked yourself why the company had that information in the first place? It's like a disease. Perhaps a C-level officer made the decision to gather this data, or maybe it was an unsupervised web designer infuriated with his manager who decided that the database could handle one more column. The decision to collect that kind of data comes easily but getting rid of it demands meeting after meeting. Given the choice between calling that meeting and playing NetHack, most of us cuddle our keyboards. After all, if the data gets stolen, you probably won't be the employee chosen for sacrifice at the Temple of Mass Media—and if you are, you can use that symbolic execution as a point on your resume demonstrating that you are experienced and land a better job.

Then there's the old data. Last year's expense reports. 1993's expense reports. Spreadsheets containing estimates of expenses before replacing the leaky roof on the building that the previous CEO moved the company out of. A folder labeled "blackmail photos," and while they're certainly incriminating, especially the one with the chocolate fountain and the barbeque tongs, nobody currently employed recognizes anyone in any of the photographs. These documents are an archive of the organization's history. When the time comes that your friendly little real estate firm serendipitously discovers a cure for cancer and the CEO decides to hire a ghostwriter to chronicle the organization's amazing history, some poor bastard is going to have to dig through all those fossilized layers searching for evidence that can be misconstrued to demonstrate brilliance.

All this data could conceivably be used—one day—if a bizarre, never-to-be-repeated series of coincidences should strike that makes the long-dreaded astrological alignment of Jupiter, Pluto, and Halley's Comet with Polaris seem commonplace. It won't happen, but it could. The most pernicious data, though, is cruft that can never possibly be used, but nobody will take the responsibility to discard. Old database backups that might, possibly, be necessary. Old databases that can never be useful under any circumstances, because the software to read those backups runs only on SCO UNIX and even NetBSD has dropped *that* binary compatibility layer. Realistically, even though you have the skills to crack open what is almost certainly a bunch of comma separated values with a weird file extension, if anyone asked, you'd be much more likely to laugh and say there is no way to read that data than actually break out file(1) and strings(1) and pipe the whole mess into Perl and produce a handy Excel-compatible spreadsheet. Images of laptop hard drives from employees who fled in 2001, because their manager declared that the next person to fill that role would need that employee's files—and then refused to release those files to said replacement. Test spreadsheets that were discarded as failures. Accounting files that were eradicated for excessive honesty and replaced with IRS-friendly versions. As your organization ages it will acquire more and more of this detritus, filling drive after drive, until nobody is willing to either look at the data or take responsibility for discarding it.

We want our systems to be clean!

We want our storage tidy and elegant.

Any reasonable sysadmin finds this offensive. We want our systems to be clean! We want our storage tidy and elegant. Lugging around petabytes of the wreckage—or worse, backing up said petabytes—violates our proprieties. Many of us itch to attack this debris, discarding what is unneeded and organizing the rest. I'm forced to call out System Administration Rule #18 here: *it is cheaper for the organization to buy more storage than to pay you to clean out existing files*. Think back on those old 9-GB hard drives. Remember how many thousands or millions of files they could hold? Opening each file, assessing the contents, and deciding if it merited survival or should be cast into the outer darkness was an overwhelming task. Those drives were minuscule by today's standards. This isn't a modern problem; my first hard drive was 20 MB, and it contained more files than I could cope with. Worse, many of those files still exist. Every system I get has more hard drive capacity than the last. I'm never quite sure what files I will need, so I copy everything from the old hard drive into an archive folder on the new system. The only thing I don't have is the code for the Sinclair ZX80 maze game that Young Lucas enjoyed playing, and I'm sure that's available somewhere on the Internet. Destroying these files is a high-risk, low-gain game for any manager. If successful, the organization can avoid spending a few hundred bucks on storage. If unsuccessful, some of those antediluvian files turn out to be of vital importance and the manager's career is over. Even options like archiving to tape pose risks. While every true sysadmin archives everything in an open source format like tar, many organizations insist on using "Enterprise Backup Systems" with an appalling habit of obsoleting support for old formats.

With ample opportunity for self-humiliation and minimal potential reward, nobody is going to tackle this morass.

You cannot solve this problem.

You *can* avoid contributing to it.

Consider the data you, personally, are responsible for. Are you following your organization's data retention policy? If your organization has no data retention policy, establish one yourself. It can be as simple as telling your team, "Hey, I want to discard all logs on these systems after 60 days. Does anyone have a problem with that?" Perhaps you'll need some data longer, and other data you can throw away after a week. A good data retention policy can even keep you out of court — logs that do not exist cannot be subpoenaed. You don't want to go to court. Court is not fun, and neither lawyers nor judges understand sysadmin humor.

Or you can buy even more storage and stop worrying.

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

MICHAEL W LUCAS is the author of *Absolute FreeBSD*, *TLS Mastery*, and *$ git sync murder*. His *DNSSEC Mastery* and *Domesticate Your Badgers* should be out in early 2022, despite earnest requests from the Humane Society. For a complete list of everything he's done, query his SNMP table. Submit your questions to letters@freebsdjournal.org.