LINUX JOURNAL™
Since 1994: The Original Magazine of the Linux Community

Manage Your Network with Ansible and SSH

Disaster-Planning Tips for Server Owners

AUGUST 2017 | ISSUE 280
http://www.linuxjournal.com

Filesystem Events with inotify

+

EMACS FOR SCIENCE

PREPARE FOR VACATION LIKE A SYSADMIN

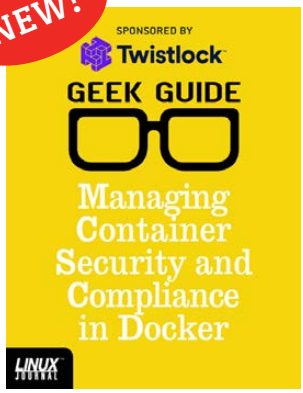
EOF: THE ACTUALLY DISTRIBUTED WEB

Create an Internet Radio Station

with Icecast and Liquidsoap

WATCH: ISSUE OVERVIEW

NEW!



Managing Container Security and Compliance in Docker

Author:
Petros Koutoupis

Sponsor:
Twistlock

Harnessing the Power of the Cloud with SUSE

Author:
Petros Koutoupis

Sponsor:
SUSE

DevOps for the Rest of Us

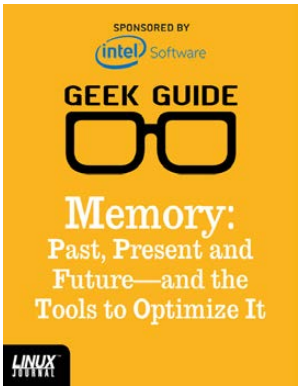
Author:
John S. Tonello

Sponsor:
Puppet

An Architect's Guide: Linux for Enterprise IT

Author:
Sol Lederman

Sponsor:
SUSE



Memory: Past, Present and Future—and the Tools to Optimize It

Author:
Petros Koutoupis

Sponsor:
Intel

Cloud-Scale Automation with Puppet

Author:
John S. Tonello

Sponsor:
Puppet

Why Innovative App Developers Love High-Speed OSDBMS

Author:
Ted Schmidt

Sponsor:
IBM

Tame the Docker Life Cycle with SUSE

Author:
John S. Tonello

Sponsor:
SUSE

CONTENTS

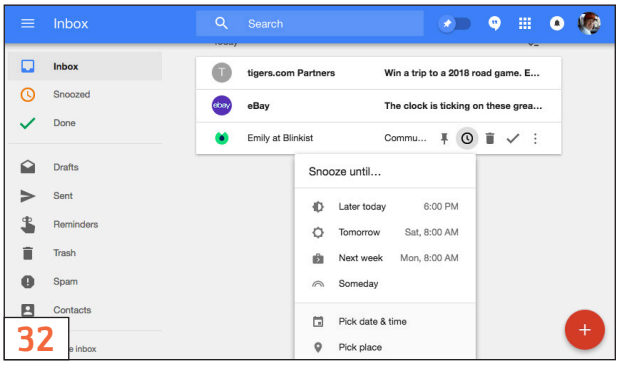
AUGUST 2017
ISSUE 280

FEATURES

Cover Image: © Can Stock Photo Inc. / dny3d

COLUMNS

22



32

IN EVERY ISSUE

ON THE COVER

- **Manage Your Network with Ansible and SSH, p. 56**
- **Disaster-Planning Tips for Server Owners, p. 34**
- **Filesystem Events with inotify, p. 88**
- **Create an Internet Radio Station with Icecast and Liquidsoap, p. 74**
- **Emacs for Science, p. 26**
- **Prepare for Vacation Like a Sysdamin, p. 50**
- **EOF: the Acutally Distributed Web, p. 114**

LINUX JOURNAL (ISSN 1075-3583) is published monthly by Belltown Media, Inc., PO Box 980985, Houston, TX 77098 USA.
Subscription rate is $29.50/year. Subscriptions start with the next issue.

LINUX JOURNAL

Executive Editor	Jill Franklin jill@linuxjournal.com
Senior Editor	Doc Searls doc@linuxjournal.com
Associate Editor	Shawn Powers shawn@linuxjournal.com
Art Director	Garrick Antikajian garrick@linuxjournal.com
Products Editor	James Gray newproducts@linuxjournal.com
Editor Emeritus	Don Marti dmarti@linuxjournal.com
Technical Editor	Michael Baxter mab@cruzio.com
Senior Columnist	Reuven Lerner reuven@lerner.co.il
Security Editor	Mick Bauer mick@visi.com
Hack Editor	Kyle Rankin lj@greenfly.net
Virtual Editor	Bill Childers bill.childers@linuxjournal.com

Contributing Editors

Ibrahim Haddad • Robert Love • Zack Brown • Dave Phillips • Marco Fioretti • Ludovic Marcotte
Paul Barry • Paul McKenney • Dave Taylor • Dirk Elmendorf • Justin Ryan • Adam Monsen

President	Carlie Fairchild publisher@linuxjournal.com
Publisher	Mark Irgang mark@linuxjournal.com
Associate Publisher	John Grogan john@linuxjournal.com
Director of Digital Experience	Katherine Druckman webmistress@linuxjournal.com
Accountant	Candy Beauchamp acct@linuxjournal.com

Linux Journal **is published by, and is a registered trade name of,**
Belltown Media, Inc.
PO Box 980985, Houston, TX 77098 USA

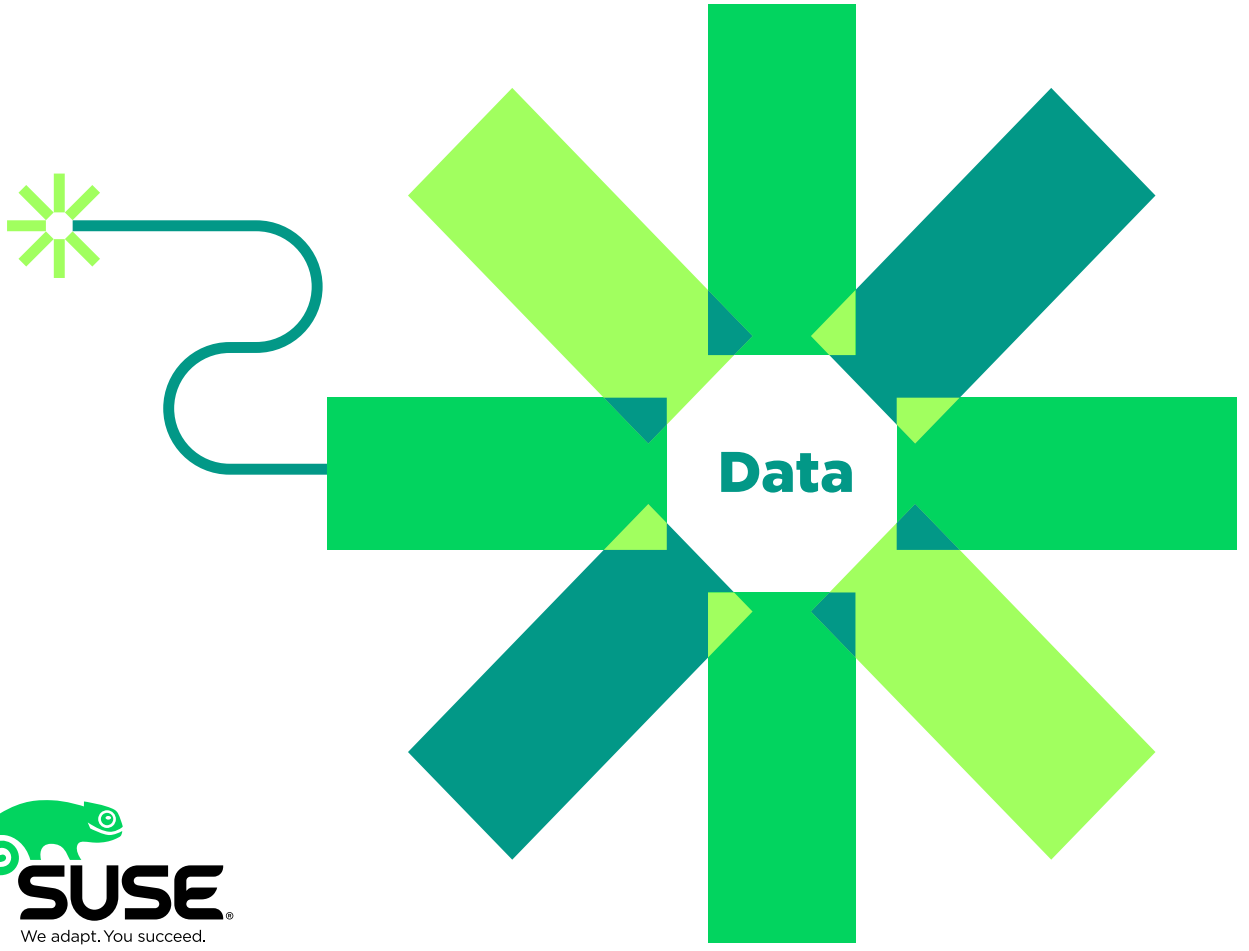
STORAGE
REDEFINED:

You cannot keep up with data explosion.

Manage data expansion with SUSE Enterprise Storage.

SUSE Enterprise Storage, the leading open source storage solution, is highly scalable and resilient, enabling high-end functionality at a fraction of the cost.

suse.com/storage



SUSE
We adapt. You succeed.

The Wacky World of Linux

SHAWN POWERS

Shawn Powers is the Associate Editor for *Linux Journal*. He's also the Gadget Guy for LinuxJournal.com, and he has an interesting collection of vintage Garfield coffee mugs. Don't let his silly hairdo fool you, he's a pretty ordinary guy and can be reached via email at shawn@linuxjournal.com. Or, swing by the #linuxjournal IRC channel on Freenode.net.

One of the nifty things about being a Linux user is how bizarre life can get. One moment you can be writing cutting-edge code, and the next you can get stuck in a nostalgia rabbit hole installing Afterstep because you used NeXTStep machines in college (guilty). This month my life got a little crazy because I had to install Microsoft Office for my daughter. The computer I had to install it on? Linux. Yet, in this wacky world we live in, it ended up working perfectly—sort of. There seems to be something new every day in the Linux world, and this month, there are lots of new things to talk about.

Reuven M. Lerner starts off this issue with some tips on disaster planning—not "evil genius" sort of planning, but rather planning for what to do when disaster inevitably happens. Yes, the obvious answer is "have a backup", but it's a bit more complicated than that, and Reuven provides sound advice.

Dave Taylor decides you deserve a bit of a break, and although you still will learn some awesome coding skills, you do it while creating a dice game. The name of the game is Bunco, which sounds like something he made up, but nonetheless, it's interesting. Be sure to check it out.

Kyle Rankin teaches how to take a vacation properly this month. If you've known Kyle for a

VIDEO: Shawn Powers runs through the latest issue.

while, you know that he's been in the awkward situation of fixing remote servers from atop a ski lift before, so his preparatory skills are worth reading.

I start a new series this issue on Ansible. I'm a big fan of DevOps tools, but so many of them have such a steep learning curve, it makes them difficult to integrate into your system. Ansible is one of my favorite configuration management platforms because it uses standard SSH for connecting to client computers. This makes it easy to start small and add more clients as you learn to take advantage of how powerful it can be.

Streaming music is the method most people use for listening nowadays, whether it's from Pandora, Spotify or any of dozens of other services. Bill Dengler shows how to create your own streaming radio station using open-source tools. If you've ever wanted to stream a live event without going through the hassle of video, it's an article you won't want to miss.

Charles Fisher finishes off the issue with instructions on using inotify to fire off events on a Linux system. I'm guilty of resorting to timed cron jobs for most things relating to filesystem changes, but with inotify, it's possible to have a filesystem change launch a process. It's a proactive way of accomplishing a task, and far, far more efficient. Charles walks through the process and helps you avoid some pitfalls along the way.

As with every issue of *Linux Journal*, this one is full of tech tips, product announcements, useful applications and reviews. Whether you need to install Microsoft Office on your Linux laptop or want to prep your data center for your trip to Hawaii, this issue has you covered. The world of Linux can be a crazy one, and we're happy to be a part of the crazy!■

RETURN TO CONTENTS

PREVIOUS
Current_Issue.tar.gz

NEXT
UpFront

Another Way

Regarding Dave Taylor's "Watermarking Images—from the Command Line" in the April 2017 issue: as usual, there is more than one way to implement a solution. Instead of this:

```
predot=$(echo $name | rev | cut --d. --f2- | rev)
postdot=$(echo $name | rev | cut --d. --f1 | rev)
newname=$(echo ${predot}--wm.$postdot)
```

I would use this:

```
predot=${name%.*}
postdot=${name##*.}
newname=${predot}-wm.$postdot
```

Or, as a one-liner:

```
newname==${name%.*}-wm.${name##*.}
```

These pattern-matching operators are available in Bash and all shells that are POSIX-compliant.

—G. Allard

Dave Taylor replies: *Thanks for sharing the fancy way to do those data field chops in Bash. The reason I don't use those sorts of notations is simply because I believe it makes the script considerably harder to understand and edit if you ever go back to it weeks, months or years later. I actually have two goals I'm*

trying to attain simultaneously with my column: write interesting code and make sure it's as understandable as possible for the widest range of readers.

Beetle

Regarding the request from a reader in the May issue's Letters section to see a photo of Shawn's truck: I'd rather see a picture of his VW. My daughter and I are just about finished with a '78 Type 1303. We had to rebuild the engine, brakes, CV joints, and on and on.

I have enjoyed the magazine a lot over the years. Keep it up.

—Steve Witt

Shawn Powers replies: *I actually have two Beetles.*

1) 1973 Super Beetle Convertible: it's my daily driver in the summer. (Winters are hard on Beetles here in Michigan. The salt eats them, so I only drive them in the summer.) It's orange with a black top, and it is in

Figure 1. 1973 Super Beetle Convertible

the perfect shape for driving. The paint isn't perfect, and the top isn't perfect, but they're presentable. Mechanically, it's solid, without any rust. It's not a show car, but because it's a bug, it turns heads!

2) This was my first bug, a couple years ago. It's a yellow 1975 Basic Beetle, model 110. It's a very unique model, because it was made only in 1975, and it is the "worst" outfitted Beetle ever made! In 1975, inflation was rising like crazy, but VW wanted to offer a Beetle for less than $3,000, so the company stripped down the Standard Beetle as much as it could and sent one to every dealership with a price of $2,999.

Figure 2. 1975 Basic Beetle

The car came with all black trim, no ventilation blower, no radio, a partial inside header cover, stripped down handles on the door, no sound-proofing in the engine compartment, and every other shortcut the VW folks could think of. Mine has a few modifications from its original model

110 status. There's an aftermarket sunroof, some chrome trim (although the original flat black was in the car, and I hope to put it back on), a radio and a conversion to carburetor. I love having such a unique piece of Beetle history, but having a super stripped-down car is an odd bragging point!

Here are some details about the model 110: https://www.thesamba.com/vw/forum/viewtopic.php?t=220215&highlight=basic+model+110.

Anyway, good luck with your daughter's car. It sounds like a wonderful project! I just wish I had a garage, because working in the driveway is no fun!

Write *LJ* a Letter

We love hearing from our readers. Please send us your comments and feedback via http://www.linuxjournal.com/contact.

PHOTOS

Send your Linux-related photos to ljeditor@linuxjournal.com, and we'll publish the best ones here.

RETURN TO CONTENTS

PREVIOUS
Letters

NEXT
Editors' Choice

diff -u

What's New in Kernel Development

Arnd Bergmann has recommended that the minimum supported **GCC** version be raised to 4.3 and the recommended version to 4.9. However, he plans to document the fact that folks like **Geert Uytterhoeven** and others were still using GCC 4.1 to compile recent Linux kernels with success.

According to Arnd's analysis, testing older and older GCC versions initially would produce more and more unnecessary warnings, making it more and more difficult to spot legitimate bugs. And beyond a certain GCC version, linker errors and build failures would begin to appear for certain architectures, notably **ARM**.

He posted a series of kernel patches that grew increasingly ugly as they had to accommodate older and older GCC versions. Some of these, he acknowledged, probably would be too ugly to make it into the kernel.

Protecting the kernel's ability to compile with ancient GCC versions is valuable work. A lot of enterprise systems can't really be upgraded without risking massive breakage, and so they just sit there unchanged for years, chugging along, bringing in revenue. But, the business has no choice but to upgrade its kernel in order to keep the

system secure. As long as the company can still use all its old tools to do that, it won't have to worry about the large expenditures needed to retool the entire system and possibly discover that it can't be done within the available budget.

The oldest supported version of **GNU Make** is being raised from 3.80 to 3.81. **Masahiro Yamada** inadvertently broke Linux support for Make 3.80 in 2014 with a cleanup patch that made it into the kernel without anyone noticing the compatibility issue.

In fact, in the intervening three years, no one else has noticed the issue either. Instead, Masahiro himself discovered what had happened and recently suggested updating the documentation to list Make version 3.81 as the official minimum version.

There were no objections of any kind. Being broken for three years with no one noticing is a fairly good argument in favor of dropping support for a given version of a tool. Even **Linus Torvalds**, in his comment to the thread, seemed to indicate that this time period was

enough to claim there were no active users of a given version.

As in the case of GCC, if even a single active user can be found for an old version of a tool, the Linux folks will make a big effort to continue to support that version. But as shown in the case of Make, with no such users standing forth, no such effort will be made.

The filesystem mounting API has been needing an update to help distinguish between a variety of actions that user code might take. But, it's also been needing new features to support modern ideas like containers for virtualization. Recently, **David Howells** addressed the latter, posting some patches to implement mount contexts. These would hold various binary data, along with the mount options, for use by the code performing the mount or by the filesystem itself.

This is valuable because it allows more detailed error reporting, which is useful for something like filesystem mounting that can fail in many different ways. But as Miklos Szeredi put it, David's code didn't go far enough to clean up the overall mount API.

Miklos wanted the mount API to have sharp distinctions between creating a filesystem instance, attaching a filesystem within an existing mounted directory, editing the superblock and adjusting the mount properties. David's work did this a little, but not enough, said Miklos.

But, David wasn't convinced of the necessity of having such extremely sharp distinctions between actions that were, after all, very closely related.

Ultimately, Miklos agreed that "everything that works now should work the same way on the old as well as the new interfaces." This way at least, there would be no breakage, and other developers could pick up where David's work left off.—Zack Brown

SAVE
10%
REGISTER TODAY
USE CODE: WILLJ

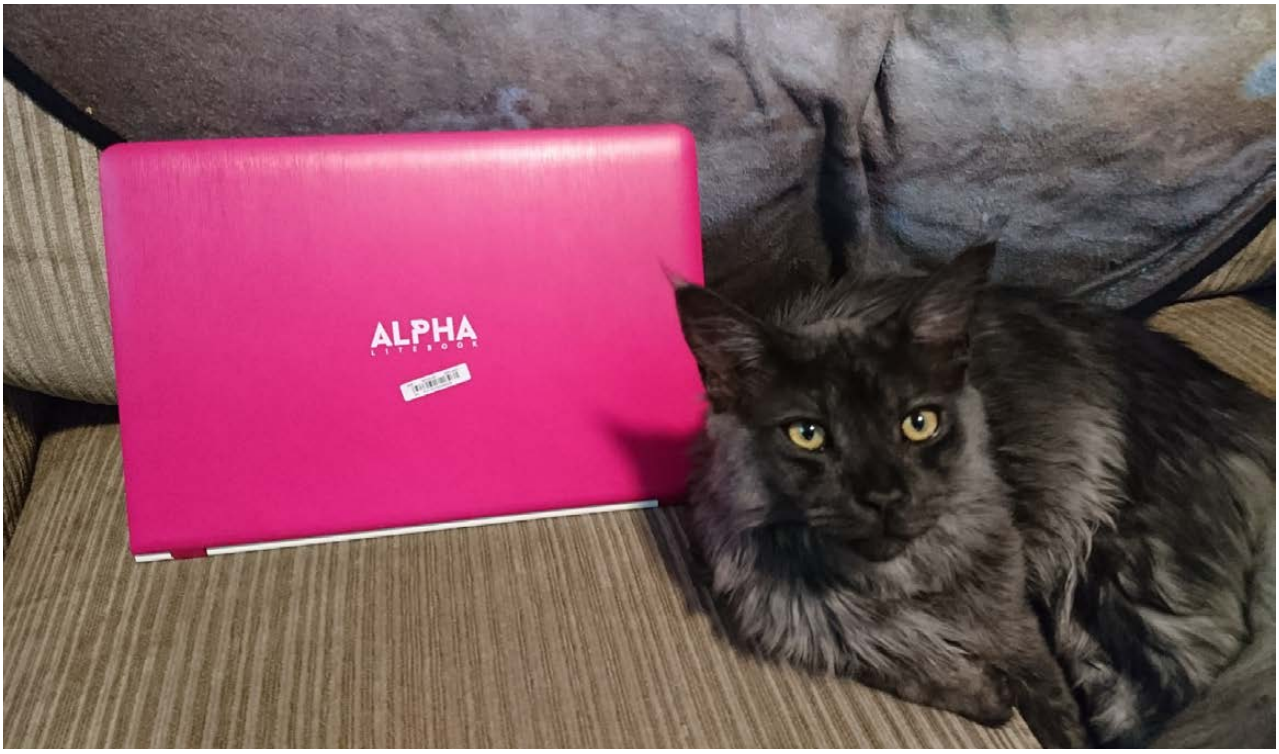
INFRASTRUCTURE
AUTOMATION
SECURITY
CLOUD
DEVOPS
SCALABILITY
IOT

WOMEN IN LINUX SUMMIT 2017

AUGUST 19, 2017

VIRTUAL CONFERENCE

WOMENINLINUX.COM



Litebook

Linux Journal reader Kevin Bruce dropped me an email about a too-good-to-be-true laptop from the folks at Alpha Store (https://alpha.store). The laptop is a 14", quad core, 1080p laptop with a 512GB hard drive for $249. My old Dell D420 has finally given up, so I was in the market for a fairly inexpensive laptop. I decided to order one. I opted for the $269 model with a hybrid drive, but everything else is the same as the base model. Rather than a full review, here are my quick takeaways:

- The screen is amazing. Really. It's bright, vibrant and has incredible colors. The screen makes my D420 look like garbage, and upon first boot, I was giddy to see how amazing it looked.

- The keyboard is a little springy, but tolerable. The plastic keys feel a little cheap, but the action is fine. It reminds me of typing on a

Macbook Pro, but with looser, cheaper plastic for the actual keys.

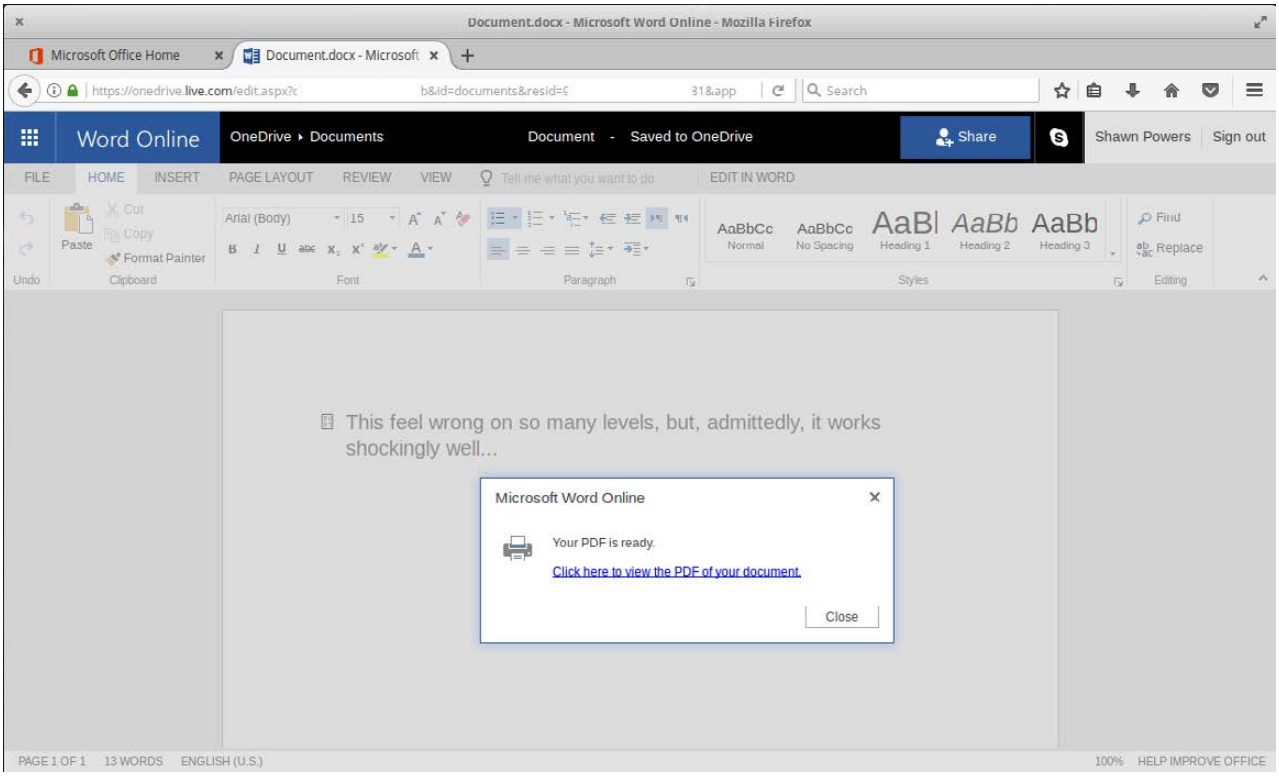
- The trackpad is horrible—and not just the sort of horrible that is annoying, but the sort of horrible that makes you unwilling to use the computer for anything other than typing or watching movies. It has an unpredictable scrolling feature, poor response and a cheap feel, and its complementing buttons are just as bad. The trackpad makes the entire experience unpleasant, and unless you're okay using an external mouse, I'd abandon ship.

There are also a few strange quirks, but none of them would have been a showstopper for me. The included Wi-Fi driver has weird issues with sleeping, so I found myself rebooting occasionally when the laptop lost connectivity. This isn't uncommon when you're trying to run Linux on a laptop that obviously was made for Windows, and I'm fairly certain finding the right driver would help.

The hybrid hard drive was really just a 32GB SSD device with the entire Elementary OS installed and a 512GB device that had four NTFS partitions on it. I could delete the partitions and reclaim the space, but it was a strange "out of the box" experience.

Finally, you can see two things in the photo. One, I have a really awesome cat. Two, the "red" model I bought is clearly pink. Not just pink, but hot pink. The white label even says it's pink, but the Alpha Store insists I bought a red laptop. The forums show other people with the same issue, so apparently to the Alpha folks, pink and red are the same thing. Honestly, I think the pink looks cooler than red anyway, but it was odd.

My advice to Kevin? Unless you want to use an external mouse, avoid this computer. The display is absolutely gorgeous, but the trackpad makes it almost unusable. The whole computer feels cheap, but that's not surprising for such an inexpensive model. If you're looking for a cheap laptop computer, I think the best option is to get a used one like the Lenovo Yoga 11e or something. They're nice machines, have better hardware (apart from maybe the screen), and you can get them for about the same price.—Shawn Powers



Microsoft Office on a Linux Machine?!

My middle child is headed to college this fall, and although the college doesn't require a specific type of computer, it does require students to have Microsoft Office. Not Microsoft Office-compatible, but specifically Microsoft Office. That bums me out, but I figured Office 365 might be just the answer for a daughter who doesn't want to get a Windows laptop.

The coolest part about Office 365 is that college students can get it free if they have a college email account. The free subscription doesn't include a downloadable, installable version of Office, but for Linux users, that doesn't matter. The online version is all we'd be able to use anyway, so for a college student, Microsoft Office doesn't actually include a Microsoft tax.

The surprising part? Office 365 works great on Linux. No really.

I opened a few native Microsoft files and created a few of my own. The OneDrive storage worked wonderfully from the web browser, and even printing worked well. You can see in the screenshot that Microsoft converts the document into a PDF file so it can be printed natively on whatever system you're using.

I'm surprised to report that Office 365 works so well on Linux, but it honestly does. For a college student, the online offering might be enough to meet all the requirements most colleges have for software. Even if you aren't a fan of Microsoft and its products, I recommend at least trying Office 365 if you're in a position that requires Microsoft Office, because it sure beats installing Windows 10! —Shawn Powers

Android Candy: Clip-on Lenses

My Sony Xperia phone takes fairly nice photos. You can see in Figure 1 that it's pretty clear, with nice color. The thing is, I want to livestream my daughters' volleyball and basketball games this year, and my phone just doesn't have a wide enough angle to capture the entire court. So, I figured I'd try some of the clip-on wide angle lenses for phones. But honestly, the results are a bit disappointing.

The first lens I tried (the larger one on the left of Figure 2) is the $10 Zomei 0.6x lens (http://a.co/3b3AdGC). This lens feels solid and heavy with good-looking glass and a metal lens case. The "clippy" part attaches with plastic threads, but still, it feels solid. Unfortunately, it didn't provide very much more coverage, and the wideness it did give made for a pretty ugly image (Figure 3).

The other lens I tried was a $12 Amir, 140° wide angle lens that

Figure 1. This is my backyard taken with my Sony Xperia.

Figure 2. Oddly, the cheaper lens felt nicer but performed worse, and the more expensive lens felt cheaper but performed better.

Figure 3. This lens felt so solid, I expected much better results.

Figure 4. This has a wider angle and a clearer picture, but both were disappointments.

does 0.4x (http://a.co/2Ozkc5d). This is a much smaller lens, and it feels much cheaper. The coverage and photo are actually fairly similar to the Zomei, but it's actually a bit nicer. Plus, the smaller size makes the phone less unwieldy when taking a photo. As you can tell in Figure 4, it's still not a great picture, but it is wider than the phone can do natively.

My advice? If you have to take a wide angle shot with a phone, step back a bit to get a wider angle the old-fashioned way. Clip-on lenses are a great idea, and they do technically work, but if you're expecting the same quality photo your phone can take natively, you'll be sadly disappointed.—Shawn Powers

Emacs for Science

I typically cover software packages that do actual calculations to advance scientific knowledge. But for this article, I am covering a slightly stranger tool in the arsenal of scientific computation.

Emacs is a text editor that has almost all the functionality of an operating system. A collection of enhancements and configuration settings are available bundled under the name of scimax. Being an Emacs user myself, I was surprised I'd never heard of it before now. This project has been in development for some time, but it recently has started to find wider attention.

Before installing it, however, you need to install the latest version of Emacs to get all of the needed functionality. As with most of my articles, I am assuming that you are using a Debian-based distribution. You can install Emacs by using the daily snapshot package, available at the official launchpad archive. Install it with the following commands:

```
sudo add-apt-repository ppa:ubuntu-elisp/ppa
sudo update
sudo install emacs-snapshot
```

This will ensure that you have the latest and greatest version.

Once this is installed, go ahead and install the scimax code itself. You can get it from the main GitHub repository with the command:

```
git clone https://github.com/jkitchin/scimax.git
```

Start it with the command:

```
emacs-snapshot -q -l path/to/scimax/init.el
```

The first time you do this, there will be a lot of activity while Emacs downloads and installs the full suite of extra packages you need in order for the scimax code to have all of the required dependencies.

When you finally have everything installed and start scimax, you will

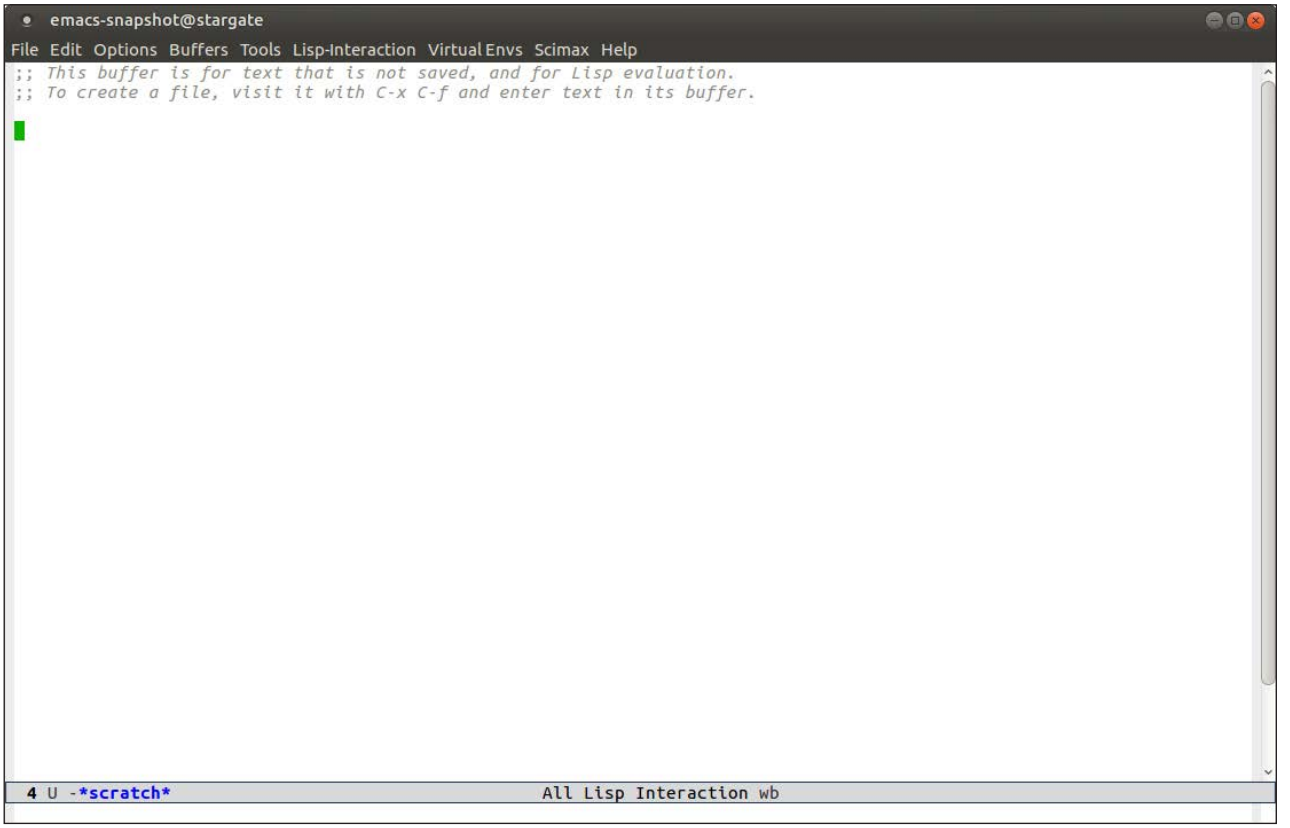


Figure 1. You will see several new menu item entries at the top of your Emacs window.

see several new menu items in your Emacs session.

The real driving need behind all of the work that has gone into scimax is to make research more easily reproducible and to handle documentation of your research with minimal extra overhead.

Since you want to work with organized documents within Emacs, the base of scimax is built on top of org-mode. Org-mode, by itself, is probably something you will want to look into as a potent tool. Scimax takes this powerful core and makes it easier to use with a series of shortcuts.

Because of org-mode's power, it would be time well spent if you learned at least the basics of how to use the main parts of this package. Org-mode has a markup syntax of its own, and scimax adds a layer of shortcuts that make it easier to write.

Along with the regular org-mode markup syntax, scimax makes it easier to include LaTeX sections for more advanced textual layout. Many people in scientific fields are familiar with LaTeX. For those who

aren't, LaTeX is document layout program where you write your text and include layout instructions for the LaTeX engine. The idea is that you separate out the text from the formatting of that text.

If you have graphical images as part of your research, scimax has added some extra functionality to make image rescaling and presentation better than the org-mode defaults by using external programs from the imagemagick package.

Because org-mode is designed to be a document structuring package for Emacs, it allows for exporting your text into a great many other formats. Also, since it separates out the formatting from the actual text, it can be applied to many different document structures, such as articles, books or web pages.

Scimax uses the ox-manuscript Emacs package to handle exporting to high-quality document formats. This is useful when you need to produce final versions of your scientific reports or articles in a format like PDF.

Bibliographic references within your document are handled through bibtex entries.

So far, I've covered a quick overview of the document management, organization and formatting tools that are provided through scimax, but Emacs and org-mode have much more functionality. You can interact with the outside world in a few different ways. The first is through email. You can grab selections of your text, or an entire buffer, and issue an `org-mime` command within Emacs to tell it to send an HTML-based email. Depending on your system, you may need additional configuration in order for this to work as expected.

The other way to interact with the outside world is through Google searches. As someone who writes a fair bit myself, I cannot understate the need for a Google window to be open to be able to verify some fact or statement as I am writing. With scimax, the `google-this` Emacs package gets installed and is available as you are working. This allows you to fire up Google searches based on either specific text selections or the contents of entire buffers immediately from the document that you are working on.

Along with communicating with the outside world, the other powerful interaction with external tools is through org-mode's

ability to run external programs and have their output inserted into sections of your document. This one piece of functionality makes the dream of reproducible research a real possibility. You do need to be diligent and put it into practice, but you no longer have the excuse of saying that it isn't possible. The idea is that, from within your org-mode document, you can define a block of code that makes some calculation or generates some graph. You then can have org-mode fire this block so that it can be run through an external engine and have the results pulled back in and inserted as the displayed text in the original location.

The default engine configured in scimax is Python, which is definitely a good starting point. With more configuration, you can add support for several other languages. The powerful idea here is that you always can go back to the original code that generated some result or some graph and re-create it. More and more scientific journals are demanding this level of reproducibility, so having it as part of your article contents directly means you never will lose track of it.

The last thing I want to cover is how to organize all of the work that scimax is helping you do. The highest level of organization is the ability to set up projects. A project is essentially a directory with all of the associated files for that given project. These projects are handled by the Emacs projectile package. This package allows you to move between projects, find files within projects or do full searches through a given project.

Projectile assumes that these project directories are under some kind of version control system, such as Git. Luckily, scimax includes the magit Emacs package, which adds lots of extra functions that allow you to interact with the Git repository that the current file belongs to directly from Emacs. You can create or clone repositories, stage and commit changes, manage diffs between versions, and even handle pushes to and pulls from remote repositories. Along with the explicit control over a Git repository, scimax includes extensions to org-mode to handle track changes, as well as to insert edit marks within your org-mode document.

Putting all of this organizational work together, scimax provides the ability to create and use scientific notebooks. A series of commands

starting with `nb-` allow you to wrap all of the organizational functionality to create, manage and archive these notebooks. Now, you have no reason not to start documenting all of your scientific research in a reproducible way—except maybe the learning curve. But, as the old saying goes, nothing worth doing is easy, and I think this is definitely worth doing, at least for some people.—Joey Bernard

Resources

- Scimax: https://github.com/jkitchin/scimax
- Emacs: https://www.gnu.org/software/emacs
- org-mode: http://orgmode.org

RETURN TO CONTENTS

THEY SAID IT

The shoe that fits one person pinches another; there is no recipe for living that suits all cases.
—Carl Jung

Courage is the ladder on which all the other virtues mount.
—Clare Booth Luce

The thing women have got to learn is that nobody gives you power. You just take it.
—Roseanne Barr

One man practicing sportsmanship is better than a hundred teaching it.
—Knute Rockne

The man who is swimming against the stream knows the strength of it.
—Woodrow Wilson

WISTEM WOMEN IN STEM 2017

THE GLOBAL WOMEN IN STEM CONFERENCE & AWARDS.

100 SPEAKERS - HUNDREDS OF WOMEN IN SCIENCE AND TECHNOLOGY

Women are increasingly becoming the engine driving global economic growth and innovation. Join us as we celebrate the women who are making this possible in spite of all the odds. With over a hundred speakers and hundreds of attendees from all across the world, WISTEM is possibly the biggest Women in STEM conference in the World.

SEPTEMBER
10th-12th
SAN FRANCISCO

EXPERIENCE THE THREE AMAZING DAYS THAT FLY BY, BUT STAY WITH YOU FOREVER.

SESSIONS & TRACKS

START-UP PITCH

AWARDS

WWW.WOMENINSTEMCONFERENCE.COM

An initiative of
MKF, A Public Non Profit
Unlocking Potential. Igniting Ideas. Empowering Actions.

Affiliate Partner Program of

GSMA MOBILE WORLD CONGRESS

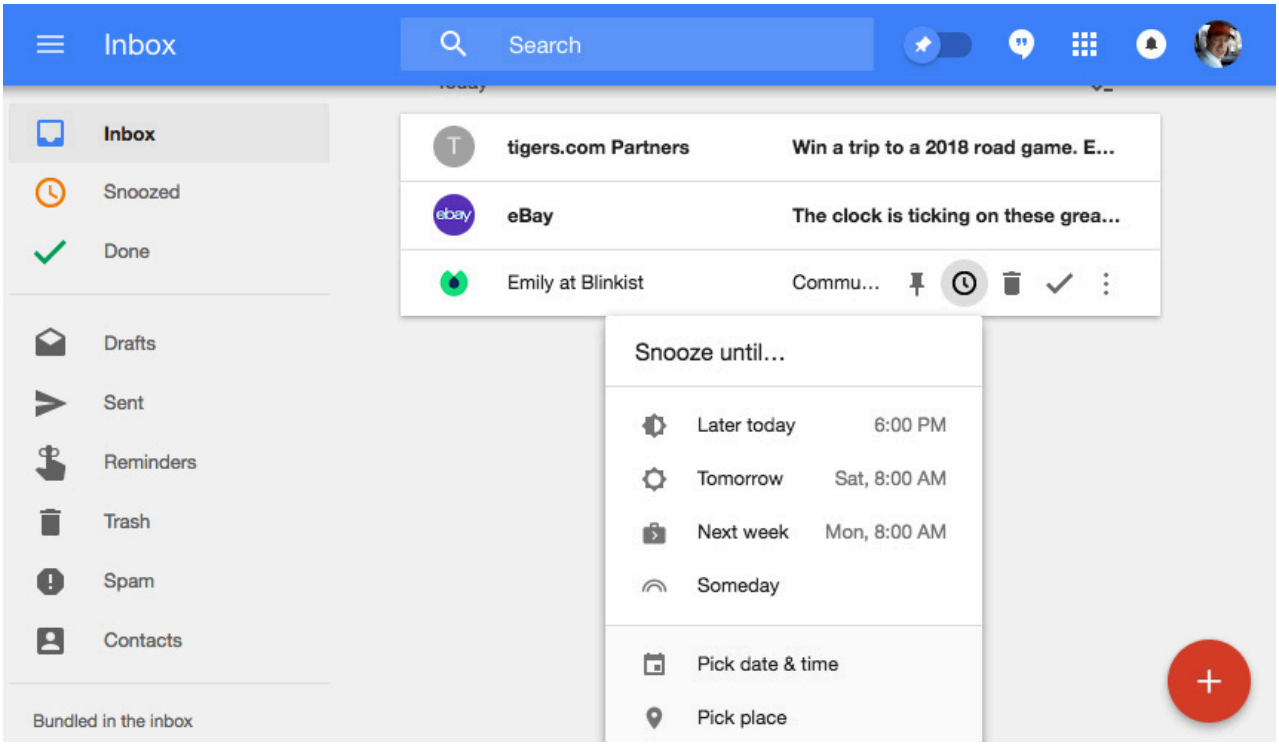
ctia Everything Wireless

PREVIOUS
UpFront

NEXT
Reuven M. Lerner's
At the Forge

My Quest for Inbox Zero

I've never been able to accomplish "inbox zero" for more than a couple hours. Part of it is due to poor organization on my part, and part of it is due to being too busy. The problem isn't that I get more email than I can handle (although that's probably true, I get a couple hundred messages a day), it's that most messages require a followup that I can't accomplish immediately. I've



tried to move items I need to handle later quickly into a to-do list like Wunderlist, but it's an extra step that takes more time, so I just let things pile up in my inbox.

For some reason, I recently decided to try Google Inbox. It's not a new offering from Google, but I think it's matured since I originally tried it years ago. See, Google Inbox allows me to keep inbox zero with minimal effort and zero guilt. Here's the premise:

1. When an email comes in, a quick swipe to the right will archive it. It's not even a click—just a swipe and it's gone.

2. If I need to open it, I can tap it, then click a check box to archive it if I don't need to follow up.

3. Here's the beauty. If I need to follow up on a message, even if it's just something interesting I want to read later at my leisure, I can snooze it. In the screenshot, you can see the options for snoozing, and they work perfectly.

The idea of snoozing email isn't new. I remember using "boomerang" for such a thing a few years ago. But Google Inbox has integrated the snooze and archive features so well that I can clear my inbox *completely* in moments and not worry that I've forgotten something. In fact, the snooze feature works much better than just leaving stuff in my inbox, because it actually comes back to my attention after the snooze period instead of falling off the end of my inbox.

Thanks to a fast, minimalist take on achieving inbox zero, I'm giving Google Inbox this month's Editors' Choice award. It's not a new product, but it has revolutionized my life. You can check it out without messing up your regular Gmail inbox by logging in at http://inbox.google.com in a web browser or by downloading the mobile app. You won't regret it!

—Shawn Powers

RETURN TO CONTENTS

Avoiding Disaster

Worried that your server will go down? You should be. Here are some disaster-planning tips for server owners.

REUVEN M. LERNER

Reuven M. Lerner, a longtime Web developer, offers training and consulting services in Python, Git, PostgreSQL and data science. He has written two programming ebooks (*Practice Makes Python* and *Practice Makes Regexp*) and publishes a free weekly newsletter for programmers, at http://lerner.co.il/newsletter. Reuven tweets at @reuvenmlerner and lives in Modi'in, Israel, with his wife and three children.

PREVIOUS
Editors' Choice

NEXT
Dave Taylor's
Work the Shell

IF YOU OWN A CAR OR A HOUSE, YOU ALMOST CERTAINLY HAVE INSURANCE. Insurance seems like a huge waste of money. You pay it every year and make sure that you get the best possible price for the best possible coverage, and then you hope you never need to use the insurance. Insurance seems like a really bad deal—until you have a disaster and realize that had it not been for the insurance, you might have been in financial ruin.

Unfortunately, disasters and mishaps are a fact of life in the computer industry. And so, just as you pay insurance and hope never to have to use it, you also need to take time to ensure the safety and reliability of your systems—not because you want disasters to happen, or even expect them to occur,

but rather because you have to.

If your website is an online brochure for your company and then goes down for a few hours or even days, it'll be embarrassing and annoying, but not financially painful. But, if your website is your business, when your site goes down, you're losing money. If that's the case, it's crucial to ensure that your server and software are not only unlikely to go down, but also easily recoverable if and when that happens.

Why am I writing about this subject? Well, let's just say that this particular problem hit close to home for me, just before I started to write this article. After years of helping clients around the world to ensure the reliability of their systems, I made the mistake of not being as thorough with my own. ("The shoemaker's children go barefoot", as the saying goes.) This means that just after launching my new online product for Python developers, a seemingly trivial upgrade turned into a disaster. The precautions I put in place, it turns out, weren't quite enough—and as I write this, I'm still putting my web server together. I'll survive, as will my server and business, but this has been a painful and important lesson—one that I'll do almost anything to avoid repeating in the future.

So in this article, I describe a number of techniques I've used to keep servers safe and sound through the years, and to reduce the chances of a complete meltdown. You can think of these techniques as insurance for your server, so that even if something does go wrong, you'll be able to recover fairly quickly.

I should note that most of the advice here assumes no redundancy in your architecture—that is, a single web server and (at most) a single database server. If you can afford to have a bunch of servers of each type, these sorts of problems tend to be much less frequent. However, that doesn't mean they go away entirely. Besides, although people like to talk about heavy-duty web applications that require massive iron in order to run, the fact is that many businesses run on small, one- and two-computer servers. Moreover, those businesses don't need more than that; the ROI (return on investment) they'll get from additional servers cannot be justified. However, the ROI from a good backup and recovery plan is huge, and thus worth the investment.

Indeed, I prefer to keep my sites in Git, backed up on a commercial hosting service, such as GitHub or Bitbucket, and then deployed using a system like Capistrano.

The Parts of a Web Application

Before I can talk about disaster preparation and recovery, it's important to consider the different parts of a web application and what those various parts mean for your planning.

For many years, my website was trivially small and simple. Even if it contained some simple programs, those generally were used for sending email or for dynamically displaying different assets to visitors. The entire site consisted of some static HTML, images, JavaScript and CSS. No database or other excitement was necessary.

At the other end of the spectrum, many people have full-blown web applications, sitting on multiple servers, with one or more databases and caches, as well as HTTP servers with extensively edited configuration files.

But even when considering those two extremes, you can see that a web application consists of only a few parts:

- The application software itself.
- Static assets for that application.
- Configuration file(s) for the HTTP server(s).
- Database configuration files.
- Database schema and contents.

Assuming that you're using a high-level language, such as Python, Ruby or JavaScript, everything in this list either is a file or can be turned into

one. (All databases make it possible to "dump" their contents onto disk, into a format that then can be loaded back into the database server.)

Consider a site containing only application software, static assets and configuration files. (In other words, no database is involved.) In many cases, such a site can be backed up reliably in Git. Indeed, I prefer to keep my sites in Git, backed up on a commercial hosting service, such as GitHub or Bitbucket, and then deployed using a system like Capistrano.

In other words, you develop the site on your own development machine. Whenever you are happy with a change that you've made, you commit the change to Git (on your local machine) and then do a `git push` to your central repository. In order to deploy your application, you then use Capistrano to do a `cap deploy`, which reads the data from the central repository, puts it into the appropriate place on the server's filesystem, and you're good to go.

This system keeps you safe in a few different ways. The code itself is located in at least three locations: your development machine, the server and the repository. And those central repositories tend to be fairly reliable, if only because it's in the financial interest of the hosting company to ensure that things are reliable.

I should add that in such a case, you also should include the HTTP server's configuration files in your Git repository. Those files aren't likely to change very often, but I can tell you from experience, if you're recovering from a crisis, the last thing you want to think about is how your Apache configuration files should look. Copying those files into your Git repository will work just fine.

Backing Up Databases

You could argue that the difference between a "website" and a "web application" is a database. Databases long have powered the back ends of many web applications and for good reason—they allow you to store and retrieve data reliably and flexibly. The power that modern open-source databases provides was unthinkable just a decade or two ago, and there's no reason to think that they'll be any less reliable in the future.

And yet, just because your database is pretty reliable doesn't mean that it won't have problems. This means you're going to want to keep a snapshot ("dump") of the database's contents around, in case the

database server corrupts information, and you need to roll back to a previous version.

My favorite solution for such a problem is to dump the database on a regular basis, preferably hourly. Here's a shell script I've used, in one form or another, for creating such regular database dumps:

```
#!/bin/sh

BACKUP_ROOT="/home/database-backups/"
YEAR=`/bin/date +'%Y'`
MONTH=`/bin/date +'%m'`
DAY=`/bin/date +'%d'`

DIRECTORY="$BACKUP_ROOT/$YEAR/$MONTH/$DAY"
USERNAME=dbuser
DATABASE=dbname
HOST=localhost
PORT=3306

/bin/mkdir -p $DIRECTORY

/usr/bin/mysqldump -h $HOST --databases $DATABASE -u $USERNAME
 ➥| /bin/gzip --best --verbose >
 ➥$DIRECTORY/$DATABASE-dump.gz
```

The above shell script starts off by defining a bunch of variables, from the directory in which I want to store the backups, to the parts of the date (stored in $YEAR, $MONTH and $DAY). This is so I can have a separate directory for each day of the month. I could, of course, go further and have separate directories for each hour, but I've found that I rarely need more than one backup from a day.

Once I have defined those variables, I then use the `mkdir` command to create a new directory. The `-p` option tells `mkdir` that if necessary, it should create all of the directories it needs such that the entire path will exist.

Finally, I then run the database's "dump" command. In this particular

case, I'm using MySQL, so I'm using the `mysqldump` command. The output from this command is a stream of SQL that can be used to re-create the database. I thus take the output from `mysqldump` and pipe it into `gzip`, which compresses the output file. Finally, the resulting dumpfile is placed, in compressed form, inside the daily backup directory.

Depending on the size of your database and the amount of disk space you have on hand, you'll have to decide just how often you want to run dumps and how often you want to clean out old ones. I know from experience that dumping every hour can cause some load problems. On one virtual machine I've used, the overall administration team was unhappy that I was dumping and compressing every hour, which they saw as an unnecessary use of system resources.

If you're worried your system will run out of disk space, you might well want to run a space-checking program that'll alert you when the filesystem is low on free space. In addition, you can run a cron job that uses `find` to erase all dumpfiles from before a certain cutoff date. I'm always a bit nervous about programs that automatically erase backups, so I generally prefer not to do this. Rather, I run a program that warns me if the disk usage is going above 85% (which is usually low enough to ensure that I can fix the problem in time, even if I'm on a long flight). Then I can go in and remove the problematic files by hand.

When you back up your database, you should be sure to back up the configuration for that database as well. The database schema and data, which are part of the dumpfile, are certainly important. However, if you find yourself having to re-create your server from scratch, you'll want to know precisely how you configured the database server, with a particular emphasis on the filesystem configuration and memory allocations. I tend to use PostgreSQL for most of my work, and although postgresql.conf is simple to understand and configure, I still like to keep it around with my dumpfiles.

Another crucial thing to do is to check your database dumps occasionally to be sure that they are working the way you want. It turns out that the backups I thought I was making weren't actually happening, in no small part because I had modified the shell script and hadn't double-checked that it was creating useful backups. Occasionally pulling out one of your dumpfiles and restoring it to a separate (and offline!)

database to check its integrity is a good practice, both to ensure that the dump is working and that you remember how to restore it in the case of an emergency.

Storing Backups

But wait. It might be great to have these backups, but what if the server goes down entirely? In the case of the code, I mentioned to ensure that it was located on more than one machine, ensuring its integrity. By contrast, your database dumps are now on the server, such that if the server fails, your database dumps will be inaccessible.

This means you'll want to have your database dumps stored elsewhere, preferably automatically. How can you do that?

There are a few relatively easy and inexpensive solutions to this problem. If you have two servers—ideally in separate physical locations—you can use `rsync` to copy the files from one to the other. Don't `rsync` the database's actual files, since those might get corrupted in transfer and aren't designed to be copied when the server is running. By contrast, the dumpfiles that you have created are more than able to go elsewhere. Setting up a remote server, with a user specifically for handling these backup transfers, shouldn't be too hard and will go a long way toward ensuring the safety of your data.

I should note that using `rsync` in this way basically requires that you set up passwordless SSH, so that you can transfer without having to be physically present to enter the password.

Another possible solution is Amazon's Simple Storage Server (S3), which offers astonishing amounts of disk space at very low prices. I know that many companies use S3 as a simple (albeit slow) backup system. You can set up a cron job to run a program that copies the contents of a particular database dumpfile directory onto a particular server. The assumption here is that you're not ever going to use these backups, meaning that S3's slow searching and access will not be an issue once you're working on the server.

Similarly, you might consider using Dropbox. Dropbox is best known for its desktop client, but it has a "headless", text-based client that can be used on Linux servers without a GUI connected. One nice advantage of Dropbox is that you can share a folder with any number of people,

which means you can have Dropbox distribute your backup databases everywhere automatically, including to a number of people on your team. The backups arrive in their Dropbox folder, and you can be sure that the LAMP is conditional.

Finally, if you're running a WordPress site, you might want to consider VaultPress, a for-pay backup system. I must admit that in the weeks before I took my server down with a database backup error, I kept seeing ads in WordPress for VaultPress. "Who would buy that?", I asked myself, thinking that I'm smart enough to do backups myself. Of course, after disaster occurred and my database was ruined, I realized that $30/year to back up all of my data is cheap, and I should have done it before.

Conclusion

When it comes to your servers, think less like an optimistic programmer and more like an insurance agent. Perhaps disaster won't strike, but if it does, will you be able to recover? Making sure that even if your server is completely unavailable, you'll be able to bring up your program and any associated database is crucial.

My preferred solution involves combining a Git repository for code and configuration files, distributed across several machines and services. For the databases, however, it's not enough to dump your database; you'll need to get that dump onto a separate machine, and preferably test the backup file on a regular basis. That way, even if things go wrong, you'll be able to get back up in no time.■

Send comments or feedback via http://www.linuxjournal.com/contact or to ljeditor@linuxjournal.com.

RETURN TO CONTENTS

Let's Play Bunco!

Bunco—a dice game that makes Yahtzee look complicated!

PREVIOUS
Reuven M. Lerner's
At the Forge

NEXT
Kyle Rankin's
Hack and /

DAVE TAYLOR

Dave Taylor has been hacking shell scripts on UNIX and Linux systems for a really long time. He's the author of *Learning Unix for Mac OS X* and *Wicked Cool Shell Scripts*. You can find him on Twitter as @DaveTaylor, or reach him through his tech Q&A site: http://www.AskDaveTaylor.com.

I HAVEN'T DUG INTO ANY GAME PROGRAMMING FOR A WHILE, so I thought it was high time to do something in that realm. At first, I thought "Halo as a shell script?", but then I came to my senses. Instead, let's look at a simple dice game called Bunco. You may not have heard of it, but I bet your Mom has—it's a quite popular game for groups of gals at a local pub or tavern.

Played in six rounds with three dice, the game is simple. You roll all three dice and have to match the current round number. If all three dice match the current round number (for example, three 3s in round three), you score 21. If all three match but aren't the current round number, it's a Mini Bunco and worth five points. Failing both of those, each die with the same value as the round number is worth one point.

Played properly, the game also involves teams,

multiple tables including a winner's table, and usually cash prizes funded by everyone paying $5 or similar to play and based on specific winning scenarios like "most Buncos" or "most points". I'll skip that part here, however, and just focus on the dice part.

Let's Do the Math

Before I go too far into the programming side of things, let me talk briefly about the math behind the game. Dice are easy to work with because on a properly weighted die, the chance of a particular value coming up is 1:6.

Random tip: not sure whether your dice are balanced? Toss them in salty water and spin them. There are some really interesting YouTube videos from the D&D world showing how to do this test.

So what are the odds of three dice having the same value? The first die has a 100% chance of having a value (no leaners here), so that's easy. The second die has a 16.66% chance of being any particular value, and then the third die has the same chance of being that value, but of course, they multiply, so three dice have about a 2.7% chance of all having the same value.

Then, it's a 16.66% chance that those three dice would be the current round's number—or, in mathematical terms: 0.166 * 0.166 * 0.166 = 0.00462.

In other words, you have a 0.46% chance of rolling a Bunco, which is a bit less than once out of every 200 rolls of three dice.

It could be tougher though. If you were playing with five dice, the chance of rolling a Mini Bunco (or Yahtzee) is 0.077%, and if you were trying to accomplish a specific value, say just sixes, then it's 0.00012% likely on any given roll—which is to say, not bloody likely!

And So into the Coding

As with every game, the hardest part is really having a good random number generator that generates truly random values. That's actually hard to affect in a shell script though, so I'm going to sidestep this entire issue and assume that the shell's built-in random number generator will be sufficient.

What's nice is that it's super easy to work with. Just reference `$RANDOM`,

and you'll have a random value between 0 and MAXINT (32767):

```
$ echo $RANDOM $RANDOM $RANDOM
10252 22142 14863
```

To constrain that to values between 1–6 use the modulus function:

```
$ echo $(( $RANDOM % 6 ))
3
$ echo $(( $RANDOM % 6 ))
0
```

Oops! I forgot to shift it one. Here's another try:

```
$ echo $(( ( $RANDOM % 6 ) + 1 ))
6
```

That's the dice-rolling feature. Let's make it a function where you can specify the variable you'd like to have the generated value as part of the invocation:

```
rolldie()
{
   local result=$1
   rolled=$(( ( $RANDOM % 6 ) + 1 ))
   eval $result=$rolled
}
```

The use of the `eval` is to ensure that the variable specified in the invocation is actually assigned the calculated value. It's easy to work with:

```
rolldie die1
```

That will load a random value between 1–6 into the variable `die1`. To roll your three dice, it's straightforward:

```
rolldie die1 ; rolldie die2 ; rolldie die3
```

Now to test the values. First, let's test for a Bunco where all three dice have the same value, and it's the value of the current round too:

```
if [ $die1 -eq $die2 ] && [ $die2 -eq $die3 ] ; then
  if [ $die1 -eq $round ] ; then
    echo "BUNCO!"
    score=25
  else
    echo "Mini Bunco!"
    score=5
  fi
```

That's probably the hardest of the tests, and notice the unusual use of test in the first conditional: `[ cond1 ] && [ cond2 ]`. If you're thinking that you could also write it as `cond1 -a cond2`, you're right. As with so much in the shell, there's more than one way to get to the solution.

The remainder of the code is straightforward; you just need to test for whether the die matches the current round value:

```
if [ $die1 -eq $round ] ; then
  score=1
fi
if [ $die2 -eq $round ] ; then
  score=$(( $score + 1 ))
fi
if [ $die3 -eq $round ] ; then
  score=$(( $score + 1 ))
fi
```

The only thing to consider here is that you don't want to score die value vs. round if you've also scored a Bunco or Mini Bunco, so the entire second set of tests needs to be within the `else` clause of the first conditional (to see if all three dice have the same value).

Put it together and specify the round number on the command line,

and here's what you have at this point:

```
$ sh bunco.sh 5
You rolled: 1 1 5
score = 1
$ sh bunco.sh 2
You rolled: 6 4 3
score = 0
$ sh bunco.sh 1
You rolled: 1 1 1
BUNCO!
score = 25
```

A Bunco so quickly? Well, as I said, there might be a slight issue with the randomness of the random number generator in the shell.

You can test it once you have the script working by running it a few hundred times and then checking to see what percentage are Bunco or Mini Bunco, but I'll leave that as an exercise for you, dear reader. Well, maybe I'll come back to it next month.

Let's finish up this script by having it accumulate score and run for all six rounds instead of specifying a round on the command line. That's easily done, because it's just a wrapper around the entire script, or, better, the big conditional statement becomes a function all its own:

```
BuncoRound()
{
   # roll, display, and score a round of bunco!
   # round is specified when invoked, score added to totalscore

   local score=0 ; local round=$1 ; local hidescore=0

   rolldie die1 ; rolldie die2 ; rolldie die3
   echo Round $round. You rolled: $die1 $die2 $die3

   if [ $die1 -eq $die2 ] && [ $die2 -eq $die3 ] ; then
     if [ $die1 -eq $round ] ; then
```

```
      echo "  BUNCO!"
      score=25
      hidescore=1
    else
      echo "  Mini Bunco!"
      score=5
      hidescore=1
    fi
  else
    if [ $die1 -eq $round ] ; then
      score=1
    fi
    if [ $die2 -eq $round ] ; then
      score=$(( $score + 1 ))
    fi
    if [ $die3 -eq $round ] ; then
      score=$(( $score + 1 ))
    fi
  fi

  if [ $hidescore -eq 0 ] ; then
    echo "  score this round: $score"
  fi

  totalscore=$(( $totalscore + $score ))
}
```

I admit, I couldn't resist a few improvements as I went along, including the addition of it showing either `Bunco`, `Mini Bunco` or a score value (that's what `$hidescore` does).

Invoking it is a breeze, and you'll use a `for` loop:

```
for round in {1..6} ; do
  BuncoRound $round
done
```

That's about the entire program at this point. Let's run it once and see what happens:

```
$ sh bunco.sh 1
Round 1. You rolled: 2 3 3
  score this round: 0
Round 2. You rolled: 2 6 6
  score this round: 1
Round 3. You rolled: 1 2 4
  score this round: 0
Round 4. You rolled: 2 1 4
  score this round: 1
Round 5. You rolled: 5 5 6
  score this round: 2
Round 6. You rolled: 2 1 3
  score this round: 0
Game over. Your total score was 4
```

Ugh. Not too impressive, but it's probably a typical round. Again, you can run it a few hundred—or thousand—times, just save the "Game over" line, then do some quick statistical analysis to see how often you score more than 3 points in six rounds. (With three dice to roll a given value, you should hit that 50% of the time.)

It's not a complicated game by any means, but it makes for an interesting little programming project. Now, what if they used 20-sided die and let you re-roll one die per round and had a dozen rounds?■

Send comments or feedback via http://www.linuxjournal.com/contact or to ljeditor@linuxjournal.com.

RETURN TO CONTENTS

InterDrone

The International Drone Conference and Exposition

Discover the Future – at the World's Largest Commercial Drone Conference & Expo





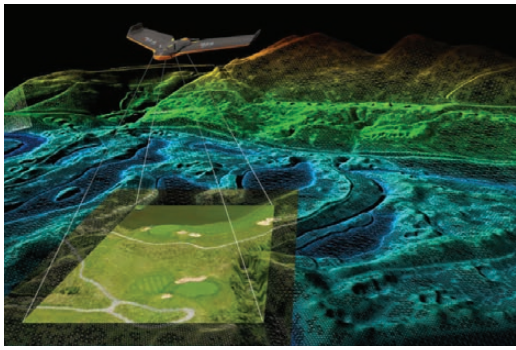


Image credit: Future Aerial Innovations

"If you want to see the state-of-the-art and expand your knowledge about the drone industry, InterDrone is the place to be."

—George Gorrill, Structural Engineer, Thomas Engineering Group

September 6-8, 2017

Las Vegas

www.InterDrone.com

Register Early for the Biggest Discount!

BZ Media

EMERALD EXPOSITIONS

Produced by BZ Media for Emerald Expositions

Preparing for Vacation

What to expect when you are expecting to go on vacation.

KYLE RANKIN

Kyle Rankin is VP of engineering operations at Final, Inc., the author of many books including *Linux Hardening in Hostile Networks*, *DevOps Troubleshooting* and *The Official Ubuntu Server Book*, and a columnist for *Linux Journal*. Follow him @kylerankin.

PREVIOUS
Dave Taylor's Work the Shell

NEXT
Shawn Powers' The Open-Source Classroom

EVERY YEAR OR TWO MY FAMILY AND I LIKE TO TAKE A VACATION ABROAD. Normally, vacation is a time to unplug, and if you are a sysadmin who's on an on-call rotation, someone else on the team typically takes over your on-call duties. Yet as you progress in your career, you start to gain more expertise and responsibilities over systems, and even with someone else on-call, there's a certain class of emergency where the team might need to reach out to you for help even when you're on vacation. I recently took a vacation abroad, and before I left, I went through a set of tasks to reduce the chance that I would need to jump on an emergency while I was away. So in this article, I describe some of the steps I take to prepare for a vacation that will help you unplug on your next trip.

Preparing Your Computer

One of the first questions you should answer before going on vacation is whether you will need to take your work laptop with you. Depending on your organization and its security controls, you might be able to perform basic emergency administrative tasks from your personal computer, tablet or phone, or you may be able to connect to production only from your work computer. In other cases, you may not need a computer, because you can just serve an advisory role over the phone or chat with other people on the team and walk them through what to do in the event of an emergency.

If you do need to take your computer, I highly recommend making a full backup before the trip. Your computer is more likely to be lost, stolen or broken while traveling than when sitting safely at the office, so I always take a backup of my work machine before a trip. Even better than taking a backup, leave your expensive work computer behind and use a cheaper more disposable machine for travel and just restore your important files and settings for work on it before you leave and wipe it when you return. If you decide to go the disposable computer route, I recommend working one or two full work days on this computer before the vacation to make sure all of your files and settings are in place.

Documentation

Good documentation is the best way to reduce or eliminate how much you have to step in when you aren't on call, whether you're on vacation or not. Everything from routine procedures to emergency response should be documented and kept up to date. Honestly, this falls under standard best practices as a sysadmin, so it's something you should have whether or not you are about to go on vacation.

First, all routine procedures from how you deploy code and configuration changes, how you manage tickets, how you perform security patches, how you add and remove users, and how the overall environment is structured should be documented in a clear step-by-step way. If you use automation tools for routine procedures, whether it's as simple as a few scripts or as complex as full orchestration tools, you should make sure you document not only how to use the automation tools, but also how to perform the same tasks manually should the

automation tools fail.

If you are on call, that means you have a monitoring system in place that scans your infrastructure for problems and pages you when it finds any. Every single system check in your monitoring tool should have a corresponding playbook that a sysadmin can follow to troubleshoot and fix the problem. If your monitoring tool allows you to customize the alerts it sends, create corresponding wiki entries for each alert name, and then customize the alert so that it provides a direct link to the playbook in the wiki.

If you happen to be the subject-matter expert on a particular system, make sure that documentation in particular is well fleshed out and understandable. These are the systems that will pull you out of your vacation, so look through those documents for any assumptions you may have made when writing them that a junior member of the team might not understand. Have other members of the team review the documentation and ask you questions.

One saying about documentation is that if something is documented in two places, one of them will be out of date. Even if you document something only in one place, there's a good chance it is out of date unless you perform routine maintenance. It's a good practice to review your documentation from time to time and update it where necessary and before a vacation is a particularly good time to do it. If you are the only person that knows about the new way to perform a procedure, you should make sure your documentation covers it.

Finally, have your team maintain a page to capture anything that happens while you are gone that they want to tell you about when you get back. If you are the main maintainer of a particular system, but they had to perform some emergency maintenance of it while you were gone, that's the kind of thing you'd like to know about when you get back. If there's a central place for the team to capture these notes, they will be more likely to write things down as they happen and less likely to forget about things when you get back.

Stable State

The more stable your infrastructure is before you leave and the more stable it stays while you are gone, the less likely you'll be disturbed on

your vacation. Right before a vacation is a terrible time to make a major change to critical systems. If you can, freeze changes in the weeks leading up to your vacation. Try to encourage other teams to push off any major changes until after you get back.

Before a vacation is also a great time to perform any preventative maintenance on your systems. Check for any systems about to hit a disk warning threshold and clear out space. In general, if you collect trending data, skim through it for any resources that are trending upward that might go past thresholds while you are gone. If you have any tasks that might add extra load to your systems while you are gone, pause or postpone them if you can. Make sure all of your backup scripts are working and all of your backups are up to date.

Emergency Contact Methods

Although it would be great to unplug completely while on vacation, there's a chance that someone from work might want to reach you in an emergency. Depending on where you plan to travel, some contact options may work better than others. For instance, some cell-phone plans that work while traveling might charge high rates for calls, but text messages and data bill at the same rates as at home. If you plan to get a local sim card, text messages sent over the cell network from home might cost more than those sent over the data plan. In the event of a local sim card, you will have to work out some way to communicate that new number to your team.

Discuss with your team what escalation path they should use to contact you in an emergency. For instance, in my case, I knew my cell-phone plan would provide me with unlimited text messages and the same data plan as at home, but I also didn't want work email to distract me. This presented a problem, as email is the primary way I'm paged. In my case, I disabled email syncing while I was on vacation and instructed everyone to contact me via text message in the case of emergency. I also needed to be on the secondary escalation path for any alerts that weren't resolved within a certain amount of time, so I configured my monitoring tool to use an email-to-SMS gateway as my email address for alerts.

If there are certain days when you know you (or your on-call counterpart at home) might be in areas with limited cell coverage, work

out those dates ahead of time and put them in your calendar. If nothing else, it might encourage others to wait on making a risky change if they know they absolutely will not be able to reach you for the next two days. In general, set expectations on your availability, and also make sure everyone takes any time zone differences into account.

Conclusion

Overall, a vacation should be a time for you to be completely removed from your work's on-call process. Whether that's possible or not, the more you prepare ahead of time, the less likely your vacation will be interrupted. Finally, when you get back, do a post mortem with your team about anything that went wrong and any documentation that was confusing or incomplete, so you can make improvements for your next vacation.■

Send comments or feedback via http://www.linuxjournal.com/contact or to ljeditor@linuxjournal.com.

RETURN TO CONTENTS

14th Annual

High Performance Computing for Wall Street- Cloud, AI and Data Centers

Show and Conference

September 12, 2017 (Tuesday) **Roosevelt Hotel, NYC**

Madison Ave and 45th St, next to Grand Central Station

Meetup September 12, Tuesday, for Finance, Trading, Banking, Exchanges, Brokerage, Funds, at one time and one place.

Register Today: HPC, Cloud, AI, Machine Learning, Data Centers, Big Data, Linux, Low Latency, Networks, Cost Savings.

Capital Markets, Systems, Architecture, Cloud, Machine Learning and AI is driving solutions for large data centers and HPC computing.

Go online for the full conference program and save $100. Includes general sessions, drill down sessions, an industry luncheon, $295 in advance. $395 on site.

Qualified end-users are invited to register at no charge. for the full conference.

Don't have time for the full Conference? Register for the free Show. at: www.flaggmgmt.com/hpc

Register online: www.flaggmgmt.com/hpc

HPC sponsors and exhibitors to show and demonstrate all new HPC systems at the Show.

Wall Street IT speakers and Gold Sponsors will lead drill-down sessions in the Grand Ballroom program.

Show Hours: Tues, Sept 12	8:00 - 4:00
Conference Hours:	8:30 - 4:50

2016 Sponsors

DELL EMC · Hewlett Packard Enterprise · cisco · SUSE · Supermicro · CRAY · Mellanox Technologies · Micron · redhat. · lenovo · Linux New Media The Pulse of Open Source · Linux Journal · information management · datanami · EnterpriseTech · HPCwire · Integration developer news · onwallstreet · NYC NoSQL · TABB Group · WSTA Wall Street Technology Association

Need Sponsorship and Exhibit Information?
Show & Conference: Flagg Management Inc
353 Lexington Avenue, New York 10016
(212) 286 0333 fax: (212) 286 0086
flaggmgmt@msn.com

Visit: www.flaggmgmt.com/hpc

Ansible: the Automation Framework That Thinks Like a Sysadmin

With Ansible, managing 50 servers is a lot like managing one server!

SHAWN POWERS

Shawn Powers is the Associate Editor for *Linux Journal*. He's also the Gadget Guy for LinuxJournal.com, and he has an interesting collection of vintage Garfield coffee mugs. Don't let his silly hairdo fool you, he's a pretty ordinary guy and can be reached via email at shawn@linuxjournal.com. Or, swing by the #linuxjournal IRC channel on Freenode.net.

PREVIOUS
Kyle Rankin's Hack and /

NEXT
New Products

I'VE WRITTEN ABOUT AND TRAINED FOLKS ON VARIOUS DEVOPS TOOLS THROUGH THE YEARS, and although they're awesome, it's obvious that most of them are designed from the mind of a developer. There's nothing wrong with that, because approaching configuration management programmatically is the whole point. Still, it wasn't until I started playing with Ansible that I felt like it was something a sysadmin

quickly would appreciate.

Part of that appreciation comes from the way Ansible communicates with its client computers—namely, via SSH. As sysadmins, you're all very familiar with connecting to computers via SSH, so right from the word "go", you have a better understanding of Ansible than the other alternatives.

With that in mind, I'm planning to spend my next few articles learning how to take advantage of Ansible. It's a great system, but when I was first exposed to it, it wasn't clear how to start. It's not that the learning curve is steep. In fact, if anything, the problem was that I didn't really have that much to learn before starting to use Ansible, and that made it confusing. For example, if you don't have to install an agent program (Ansible doesn't have any software installed on the client computers), how do you start?

Getting to the Starting Line

The reason Ansible was so difficult for me at first is because it's so flexible with how to configure the server/client relationship, I didn't know what I was supposed to do. The truth is that Ansible doesn't really care how you set up the SSH system; it will utilize whatever configuration you have. There are just a couple things to consider:

1. Ansible needs to connect to the client computer via SSH.

2. Once connected, Ansible needs to elevate privilege so it can configure the system, install packages and so on.

Unfortunately, those two considerations really open a can of worms. Connecting to a remote computer and elevating privilege is a scary thing to allow. For some reason, it feels less vulnerable when you simply install an agent on the remote computer and let Chef or Puppet handle privilege escalation. It's not that Ansible is any less secure, but rather, it puts the security decisions in your hands.

Next I'm going to list a bunch of potential configurations, along with the pros and cons of each. This isn't an exhaustive list, but it should get you thinking along the right lines for what will be ideal

in your environment. I also should note that I'm not going to mention systems like Vagrant, because although Vagrant is wonderful for building a quick infrastructure for testing and developing, it's so very different from a bunch of servers that the considerations are too dissimilar really to compare.

Some SSH Scenarios

1) SSHing into remote computer as root with password in Ansible config.

I started with a terrible idea. The "pros" of this setup is that it eliminates the need for privilege escalation, and there are no other user accounts required on the remote server. But, the cost for such convenience isn't worth it. First, most systems won't let you SSH in as root without changing the default configuration. Those default configurations are there because, quite frankly, it's just a bad idea to allow the root user to connect remotely. Second, putting a root password in a plain-text configuration file on the Ansible machine is mortifying. Really, I mentioned this possibility because it *is* a possibility, but it's one that should be avoided. Remember, Ansible allows you to configure the connection yourself, and it will let you do really dumb things. Please don't.

2) SSHing into a remote computer as a regular user, using a password stored in the Ansible config.

An advantage of this scenario is that it doesn't require much configuration of the clients. Most users are able to SSH in by default, so Ansible should be able to use credentials and log in fine. I personally dislike the idea of a password being stored in plain text in a configuration file, but at least it isn't the root password. If you use this method, be sure to consider how privilege escalation will take place on the remote server. I know I haven't talked about escalating privilege yet, but if you have a password in the config file, that same password likely will be used to gain sudo access. So with one slip, you've compromised not only the remote user's account, but also potentially the entire system.

3) SSHing into a remote computer as a regular user, authenticating with a key pair that has an empty passphrase.

This eliminates storing passwords in a configuration file, at least for the logging in part of the process. Key pairs without passphrases aren't

ideal, but it's something I often do in an environment like my house. On my internal network, I typically use a key pair without a passphrase to automate many things like cron jobs that require authentication. This isn't the most secure option, because a compromised private key means unrestricted access to the remote user's account, but I like it better than a password in a config file.

4) SSHing into a remote computer as a regular user, authenticating with a key pair that is secured by a passphrase.

This is a very secure way of handling remote access, because it requires two different authentication factors: 1) the private key and 2) the passphrase to decrypt it. If you're just running Ansible interactively, this might be the ideal setup. When you run a command, Ansible should prompt you for the private key's passphrase, and then it'll use the key pair to log in to the remote system. Yes, the same could be done by just using a standard password login and not specifying the password in the configuration file, but if you're going to be typing a password on the command line anyway, why not add the layer of protection a key pair offers?

5) SSHing with a passphrase-protected key pair, but using ssh-agent to "unlock" the private key.

This doesn't perfectly answer the question of unattended, automated Ansible commands, but it does make a fairly secure setup convenient as well. The ssh-agent program authenticates the passphrase one time and then uses that authentication to make future connections. When I'm using Ansible, this is what I think I'd like to be doing. If I'm completely honest, I still usually use key pairs without passphrases, but that's typically because I'm working on my home servers, not something prone to attack.

There are some other considerations to keep in mind when configuring your SSH environment. Perhaps you're able to restrict the Ansible user (which is often your local user name) so it can log in only from a specific IP address. Perhaps your Ansible server can live in a different subnet, behind a strong firewall so its private keys are more difficult to access remotely. Maybe the Ansible server doesn't have an SSH server installed on itself so there's no incoming access at all. Again, one of the strengths of Ansible is that it uses the SSH protocol for communication, and it's a protocol you've all had years to tweak into a system that works best

in your environment. I'm not a big fan of proclaiming what the "best practice" is, because in reality, the best practice is to consider your environment and choose the setup that fits your situation the best.

Privilege Escalation

Once your Ansible server connects to its clients via SSH, it needs to be able to escalate privilege. If you chose option 1 above, you're already root, and this is a moot point. But since no one chose option 1 (right?), you need to consider how a regular user on the client computer gains access. Ansible supports a wide variety of escalation systems, but in Linux, the most common options are sudo and su. As with SSH, there are a few situations to consider, although there are certainly other options.

1) Escalate privilege with su.

For Red Hat/CentOS users, the instinct might be to use su in order to gain system access. By default, those systems configure the root password during install, and to gain privileged access, you need to type it in. The problem with using su is that although it gives you total access to the remote system, it also gives you total access to the remote system. (Yes, that was sarcasm.) Also, the su program doesn't have the ability to authenticate with key pairs, so the password either must be interactively typed or stored in the configuration file. And since it's literally the root password, storing it in the config file should sound like a horrible idea, because it is.

2) Escalate privilege with sudo.

This is how Debian/Ubuntu systems are configured. A user in the correct group has access to `sudo` a command and execute it with root privileges. Out of the box, this still has the problem of password storage or interactive typing. Since storing the user's password in the configuration file seems a little less horrible, I guess this is a step up from using su, but it still gives complete access to a system if the password is compromised. (After all, typing `sudo su -` will allow users to become root just as if they had the root password.)

3) Escalate privilege with sudo and configure NOPASSWD in the sudoers file.

Again, in my local environment, this is what I do. It's not perfect, because it gives unrestricted root access to the user account and doesn't

require any passwords. But when I do this, and use SSH key pairs without passphrases, it allows me to automate Ansible commands easily. I'll note again, that although it is convenient, it is not a terribly secure idea.

4) Escalate privilege with sudo and configure NOPASSWD on specific executables.

This idea might be the best compromise of security and convenience. Basically, if you know what you plan to do with Ansible, you can give NOPASSWD privilege to the remote user for just those applications it will need to use. It might get a little confusing, since Ansible uses Python for lots of things, but with enough trial and error, you should be able to figure things out. It is more work, but does eliminate some of the glaring security holes.

Implementing Your Plan

Once you decide how you're going to handle Ansible authentication and privilege escalation, you need to set it up. After you become well versed at Ansible, you might be able to use the tool itself to help "bootstrap" new clients, but at first, it's important to configure clients manually so you know what's happening. It's far better to automate a process you're familiar with than to start with automation from the beginning.

I've written about SSH key pairs in the past, and there are countless articles online for setting it up. The short version, from your Ansible computer, looks something like this:

```
# ssh-keygen
# ssh-copy-id -i .ssh/id_dsa.pub remoteuser@remote.computer.ip
# ssh remoteuser@remote.computer.ip
```

If you've chosen to use no passphrase when creating your key pairs, that last step should get you into the remote computer without typing a password or passphrase.

In order to set up privilege escalation in sudo, you'll need to edit the sudoers file. You shouldn't edit the file directly, but rather use:

```
# sudo visudo
```

This will open the sudoers file and allow you to make changes safely (it error-checks when you save, so you don't accidentally lock yourself out with a typo). There are examples in the file, so you should be able to figure out how to assign the exact privileges you want.

Once it's all configured, you should test it manually before bringing Ansible into the picture. Try SSHing to the remote client, and then try escalating privilege using whatever methods you've chosen. Once you have configured the way you'll connect, it's time to install Ansible.

Installing Ansible

Since the Ansible program gets installed only on the single computer, it's not a big chore to get going. Red Hat/Ubuntu systems do package installs a bit differently, but neither is difficult.

In Red Hat/CentOS, first enable the EPEL repository:

```
sudo yum install epel-release
```

Then install Ansible:

```
sudo yum install ansible
```

In Ubuntu, first enable the Ansible PPA:

```
sudo apt-add-repository spa:ansible/ansible
(press ENTER to access the key and add the repo)
```

Then install Ansible:

```
sudo apt-get update
sudo apt-get install ansible
```

Configuring Ansible Hosts File

The Ansible system has no way of knowing which clients you want it to control unless you give it a list of computers. That list is very simple, and it looks something like this:

```
# file /etc/ansible/hosts

[webservers]
blogserver ansible_host=192.168.1.5
wikiserver ansible_host=192.168.1.10

[dbservers]
mysql_1 ansible_host=192.168.1.22
pgsql_1 ansible_host=192.168.1.23
```

The bracketed sections are specifying groups. Individual hosts can be listed in multiple groups, and Ansible can refer either to individual hosts or groups. This is also the configuration file where things like plain-text passwords would be stored, if that's the sort of setup you've planned. Each line in the configuration file configures a single host, and you can add multiple declarations after the `ansible_host` statement. Some useful options are:

```
ansible_ssh_pass
ansible_become
ansible_become_method
ansible_become_user
ansible_become_pass
```

The Ansible Vault

I also should note that although the setup is more complex, and not something you'll likely do during your first foray into the world of Ansible, the program does offer a way to encrypt passwords in a vault. Once you're familiar with Ansible and you want to put it into production, storing those passwords in an encrypted Ansible vault is ideal. But in the spirit of learning to crawl before you walk, I recommend starting in a non-production environment and using passwordless methods at first.

Finally, you should test your system to make sure your clients are connecting. The ping test will make sure the Ansible computer can ping each host:

```
ansible -m ping all
```

After running, you should see a message for each defined host showing a `ping: pong` if the ping was successful. This doesn't actually test authentication, just the network connectivity. Try this to test your authentication:

```
ansible -m shell -a 'uptime' webservers
```

You should see the results of the `uptime` command for each host in the `webservers` group.

In my next article, I'll start to dig in to Ansible's ability to manage the remote computers. I'll look at various modules and how you can use the ad-hoc mode to accomplish in a few keystrokes what would take a long time to handle individually on the command line. If you didn't get the results you expected from the sample Ansible commands above, take this time to make sure authentication is working. Check out http://docs.ansible.com for more help if you get stuck.■

Send comments or feedback via http://www.linuxjournal.com/contact or to ljeditor@linuxjournal.com.

RETURN TO CONTENTS

A STRATEGIC CONFERENCE

Developing for the precision agriculture market?

This is your chance to connect with thought leaders driving the future of precision agriculture.

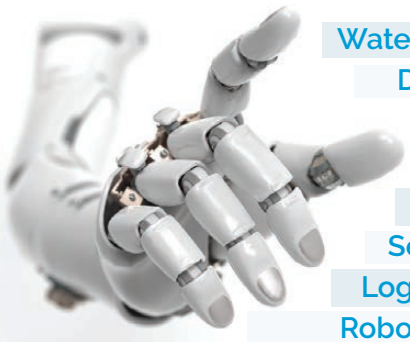
WHY ATTEND?

- **Networking**
  Connect with ag technology influencers across a variety of platforms. Delegates will establish senior-level contacts and foster a dialogue that endures after the conference's end.

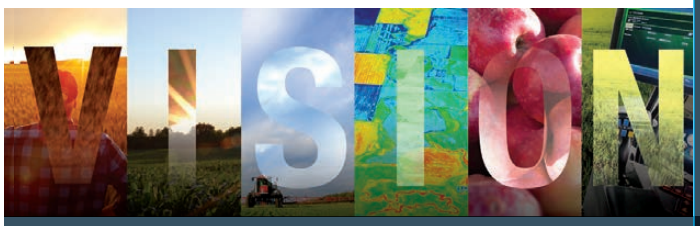
- **Conference**
  Take home a deeper understanding of the challenges faced by growers, service providers, and colleagues working in other parts of the nation and world.

- **Get an "In the Field" Perspective with a Pre-Conference Tour**
  Match learnings and contacts gleaned from the conference with a real-world view of Arizona's agricultural technology and the University of Arizona Robotics System.

WITH A FOCUS ON:



- Water + Irrigation
- Data
- Labor
- Energy
- Water Management
- Sensors + Iot
- Logistics
- Robotics
- Labels
- Traceability
- Food Safety
- Sustainability



SESSION TOPICS INCLUDE:

THE STATE OF THE INDUSTRY – PRECISION IN ROW AND SPECIALTY CROP PRODUCTION

EXPLORING THE CONNECTIVE TISSUE BETWEEN EXISTING AND EMERGING TECHNOLOGIES

140+ COMPANIES
35 LEADING SPEAKERS
12 COUNTRIES
185+ ATTENDEES

PRECISIONAg®
VISION Conference
October 10-12, 2017 | Phoenix, AZ

#PRECISIONAGVISION

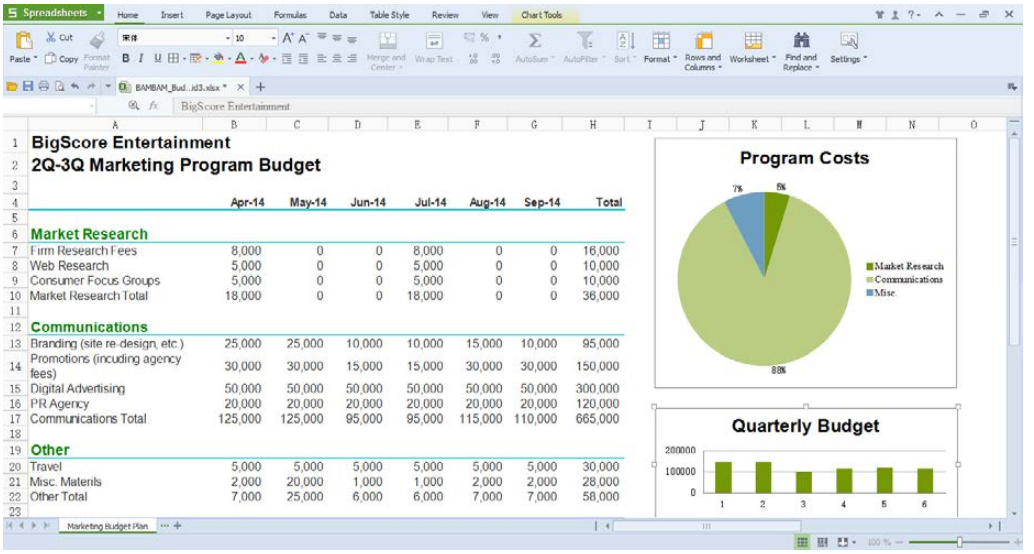
PRECISIONAGVISION.COM

PREVIOUS
Shawn Powers' The Open-Source Classroom

NEXT
Feature: Creating an Internet Radio Station with Icecast and Liquidsoap

JMR SiloStor NVMe SSD Drives

Compute-intensive workflows are the environments in which the newly developed JMR SiloStor NVMe family of SSD drives is designed to show its colors. Ideal for HPC, data centers, genome research, content creation, CGI/animation, codec processing and gaming, among others, the SiloStor drive family comes in three NVMe/PCIe configurations: single-drive module, x4 PCIe connectivity in 512GB/1TB/2TB capacities; dual-drive, x8 connectivity in 1TB/2TB/4TB capacities; and quad-drive module, x8 connectivity, available in 2TB/4TB/8TB capacities. The dual- and quad-drive cards incorporate a PCIe switch, and the drives can be striped (on a single card) for additional performance. All SiloStor designs incorporate active heatsink coolers on the drive modules themselves, maintaining low operating temperatures even during intensive sequential write operations. Key performance metrics include <1 mS average access time of <1 mS, 2 million hours MTBF, 1,200 TBW minimum endurance, 90,000/70,000 IOPS random 4K read/write speed and 4,000/3,000 MB/sequential read/write speed.
http://jmr.com



WPS Office 2016 for Linux

Promising the world's best office experience for the Linux community, WPS Software presents WPS Office 2016 for Linux: a high-performing yet considerably more affordable alternative to Microsoft Office that is fully compatible with and comparable to the constituent PowerPoint, Excel and Word applications. The WPS Office suite, with more than 1.2 billion installs across all platforms, is a complete office suite, including Writer, Presentation, Spreadsheets and a built-in PDF reader. Linux, Windows, Android and iOS versions are available. WPS Office 2016 for Linux offers enhancements for the international Linux user community including remote file sharing, added search functionality, updated WPS export to PDF hyperlinks and improved IO operations for improved WPS file access speed. Compatibility with Microsoft Office document formats includes PPT, DOC, DOCX, XLS and XLSX. The Linux edition of WPS Office is compatible with Fedora, CentOS, OpenSUSE, Ubuntu, Mint, Knoppix and other platforms, supporting both 32- and 64-bit computing environments. The latest update is made possible with the support of the WPS Office Linux community. http://www.wps.com and http://wps-community.org

Ocado Technology's Kubermesh

Instead of relying on servers concentrated in one large data center, the new Kubermesh is designed to simplify data-center architectures for smart factories by elegantly and cost effectively leveraging a distributed network of computing nodes spread across the enterprise. Developed by Ocado Technology, a division of Ocado (the world's largest online-only supermarket), the Kubermesh package uses container-based technology and the Kubernetes system to implement an on-premises private cloud architecture in which desktop computers can be configured as nodes supporting the compute or storage functionality typically delivered by high-performance servers in a data center. Ocado Technology observes that Kubermesh-based nodes are fault-tolerant, secure, flexible and designed to process the generous amounts of real-time data generated in smart factories. By distributing data-center functionality into a mesh network of nodes, Kubermesh alleviates the need for a dedicated data center and complex networking infrastructure, resulting in significant reductions in not just energy consumption but also the capital and significant operational expenditures that come with maintaining in-house high-performance servers. With Kubermesh, Ocado Technology hopes for internal gains through unlocking the potential of container technology and external gains as the Open Source community deploys and develops Kubermesh in new and exciting ways.
http://www.ocadotechnology.com

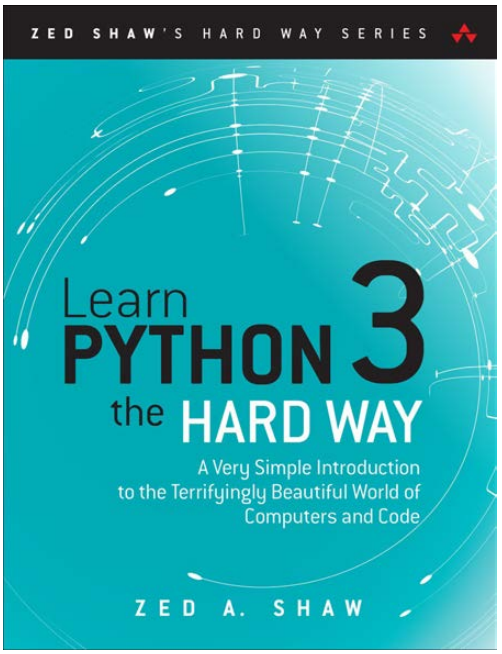
Nativ Vita

The motto "open to anything" underpins Nativ's development philosophy on all of its audio solutions, including its new Nativ Vita, "the world's first High-Resolution Music Player" and touchscreen control center that is designed to function as the central access point for one's entire music collection. This philosophy is evident in Nativ Vita's Linux and open-source internals, offering advantages like support for virtually any music service—even lesser-known and regional services like Jango Radio, KKBox and Paradise Radio—and extensibility far beyond pure audio applications. Naturally, Nativ Vita supports mainstream music services like Apple Music, SoundCloud, Vevo, Spotify, TIDAL, Pandora and Amazon Music, among others. Nativ Vita can store up to 4TB of music on its internal hard disk drives or SSDs and can access remote files on a PC, NAS or smartphone. Wireless streaming to multi-room speaker systems is achieved utilizing popular solutions like SONOS and Bluesound and to high-end headphones via Bluetooth aptX. A high-end digital output stage with myriad outputs ranging from AES/EBU to USB Audio Class 2.0 connect the Vita to an amplifier or USB DAC for best-in-class sound performance.
http://nativsound.com

Sander van Vugt's *Linux Foundation Certified Engineer* (LFCE) Video Course (Pearson IT Certification)

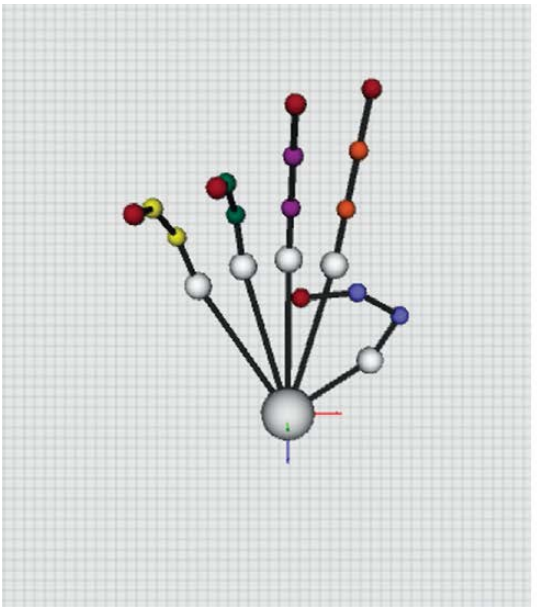
The Linux Foundation Certified Engineer (LFCE) certification is for administrators seeking to increase their breadth and depth of knowledge beyond the Linux Foundation Certified System Administrator (LFCS) level. Professionals among us striving for this greater expertise have at their disposal a new LFCE-oriented educational resource, Sander van Vugt's Linux Foundation Certified Engineer (LFCE) Video Course, featuring ten hours of comprehensive video instruction. Containing everything that exam candidates require to prepare for and pass the LFCE exam, this comprehensive training includes whiteboard concept teaching to illustrate difficult concepts, live CLI instruction to demonstrate Linux in action, screencast teaching, hands-on labs, solution videos and practice exam walk-throughs. Author van Vugt, with his 20+ years of practical Linux teaching experience, covers the LFCE material in five modules: Managing Networking, Managing File Services, Managing Web Services, Managing Mail Services and Managing Infrastructure Services. Publisher Pearson IT Certification adds that the resource is also appropriate for engineers or administrators who want to develop their Linux skills or write software for Linux.

http://www.informit.com/livelessons



Zed A. Shaw's *Learn Python 3 the Hard Way* (Addison-Wesley Professional)

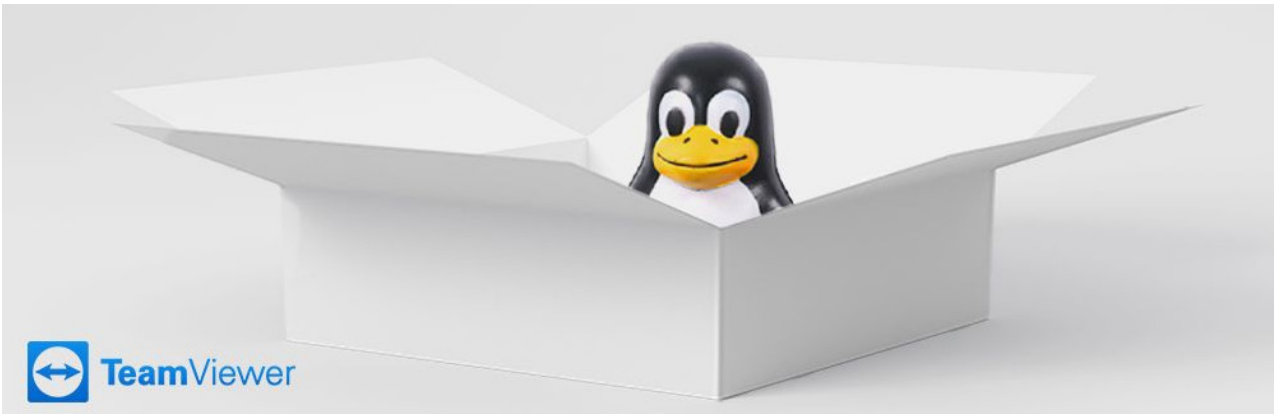
Author Zed A. Shaw makes a simple promise in his *Hard Way* series of books from publisher Addison-Wesley Professional: "It'll be hard at first. But soon, you'll just get it—and that will feel great!" Shaw's latest book in the series is called *Learn Python 3 the Hard Way: A Very Simple Introduction to the Terrifyingly Beautiful World of Computers and Code*. In the book, readers learn Python by working through 52 "brilliantly crafted exercises" in a purposefully proscribed manner. After reading the exercise, readers type the code precisely—with no copying and pasting! Then readers fix their mistakes and watch the program run. The process teaches essentials of how a computer works, what good programs look like, and how to read, write and think about computer code. Shaw teaches even more in 5+ hours of video where he shows readers how to break, fix and debug code—live, as he's doing the exercises. Lessons cover topics from installing a complete Python environment to working with code, basic mathematics, variables, looping and logic, object-oriented programming, Python packaging, automated testing and much more. Readers bring the discipline, commitment and persistence to Shaw's formula, and the output will be a Python programmer!
http://awprofessional.com



BeBop Sensors, Inc.'s Marcel Modular Data Gloves

In the fabric-embedded sensors space, all controllers need to be accurate and fast. "If latency is more than 6–8 milliseconds, you are out of the band", cautions BeBop Sensors Inc., maker of the new Marcel Modular Data Glove solution for virtual and augmented reality OEMs. Utilizing BeBop's patented fabric sensor technology and designed for accurate, real-time control and navigation in these environments, the BeBop Data Gloves are available to OEMs in 5, 10 and 14 sensor versions. They provide haptic—that is, kinesthetic—feedback and sense knuckle and abduction motion of the human hand. BeBop's basic configuration provides high-speed sensor processing as well as a 6 or 9 degrees-of-freedom inertial measurement unit that measures acceleration and angular rate. Fast, deterministic sensing provides sub-frame latency at 120Hz for real-time control of games and environments. Mezzanine boards can be added to the printed circuit board assembly stack to add functionality, such as translation and haptic electronics. A haptic audio creation kit is available, enabling content creators to customize and add to the haptic library.

http://bebopsensors.com



TeamViewer Linux Host

At last abandoning WINE and launching native Linux support, TeamViewer announced the availability of a new preview version of its Linux Host with native Linux support. The new release of TeamViewer, a solution for remote support, remote access and online meetings, addresses additional critical system administrator requirements, including support of Wake-On-LAN, assignment of TeamViewer accounts via GUI and additional regulation capabilities. Wake-On-LAN support givers users the power to wake up Linux devices that are in standby mode and connected to a power supply. Meanwhile, account assignment via GUI on the TeamViewer client permits account owners to share their contacts with each other to support the wider team as well as maintain devices around the clock 24/7. Finally, a "Confirm all" setting lets users ensure that all actions must be directly confirmed from that device, which improves overall security. The TeamViewer Linux Host requires at least Qt 5.2, Linux Kernel 2.6.27 and GLIBC 2.17.
http://teamviewer.com

Please send information about releases of Linux-related products to newproducts@linuxjournal.com or New Products c/o Linux Journal, PO Box 980985, Houston, TX 77098. Submissions are edited for length and content.

RETURN TO CONTENTS

Creating an Internet Radio Station with Icecast and Liquidsoap

Ever wanted to stream prerecorded music or a live event, such as a lecture or concert for an internet audience? With Icecast and Liquidsoap, you can set up a full-featured, flexible internet radio station using free software and open standards.

BILL DENGLER

PREVIOUS
New Products

NEXT
Feature: Linux Filesystem Events with inotify

Icecast is "a streaming media (audio/video) server that currently supports Ogg (Vorbis and Theora), Opus, WebM and MP3 streams. It can be used to create an internet radio station or a privately running jukebox and many things in between. It is very versatile in that new formats can be added relatively easily and supports open standards for communication and interaction."

Liquidsoap is "a powerful and flexible language for describing your streams. It offers a rich collection of operators that you can combine at will, giving you more power than you need for creating or transforming streams. But Liquidsoap is still very light and easy to use, in the UNIX tradition of simple strong components working together."

When combined, Icecast and Liquidsoap can create a flexible, feature-rich internet radio station. In this article, I describe how to configure Icecast to host an internet radio station. Then, I explain how to install and configure Liquidsoap to connect to Icecast, adding random (or sequential) music playback with smart cross-fading, prerecorded randomly inserted announcements and jingles, a song request system and support for live streams, with automated recording and seamless switching between live and automated programming. I also show how to configure the server to serve your stream in MP3, Ogg and Opus formats for maximum player compatibility.

Icecast, Vorbis and related projects are maintained by Xiph.Org (https://www.xiph.org), a nonprofit organization that develops open multimedia standards and software. To ensure that you are running the latest version of Icecast, with all (or most) features, you should install from an official Xiph.Org repository. Visit the list of official repositories at https://wiki.xiph.org/Icecast_Server/Installing_latest_version_(official_Xiph_repositories), and follow the instructions on that page to add the Icecast repository for your distribution. Then, install using your system's package manager. On Debian-based systems (such as Ubuntu), you may be asked to "configure Icecast" during package installation; select "no" as you will configure the server manually if you are following along with this article.

Open the Icecast configuration file using your preferred text editor. On Debian-based systems, the file is located at /etc/icecast2/icecast.xml. The location on other systems may differ; check your package's documentation for the correct path. The configuration file is in XML format and is divided

into several sections. First, enter your server's location and email into the location and admin fields, respectively—for example:

```
<location>The Heart of Gold</location>
<admin>zaphodb42@mail.example.com</admin>
```

Since each format you'll set up in Liquidsoap is a separate Icecast "source", you'll quickly exhaust the default source limit of two. So, change that to ten:

```
<sources>10</sources>
```

Unless you anticipate listeners connecting from slow or low-bandwidth environments, disabling Icecast's burst-on-connect feature will significantly decrease latency:

```
<burst-on-connect>0</burst-on-connect>
        <burst-size>0</burst-size>
```

The default passwords, "hackme", invite security compromise. Change them to something else. Also, it's probably a good idea to change the default admin user name. The following passwords are just examples; change them for your configuration both here and when they are mentioned later in the article:

```
<source-password>dontpanic</source-password>
    <relay-password>dontpanic42</relay-password>
            <admin-user>zaphod</admin-user>
            <admin-password>2Headsarebetterthanone!</admin-password>
```

Enter your system's fully qualified domain name in the hostname field:

```
<hostname>example.com</hostname>
```

Save and close the file. If you edited the file as root, you'll need to reset its permissions. On Debian-based systems, Icecast runs under user icecast2

Many distributions provide broken and out-of-date versions of Liquidsoap in their repositories. For this reason (along with improved ability to customize your installation), the Liquidsoap developers recommend installing it using the OCaml Package Manager (opam).

and group icecast. To fix permissions on a Debian-based system, run:

```
chown icecast2:icecast /etc/icecast2/icecast.xml
```

On Debian-based systems, Icecast's system service is disabled by default. Open the file /etc/default/icecast2, and set enabled to true. Then save and close the file.

Most modern Linux systems use systemd for service management. To enable Icecast on boot and start it for this session, run the following commands as root (using sudo or similar):

```
systemctl enable icecast2
systemctl start icecast2
```

Service names on various systems differ; if those commands don't work, check your system's documentation for the correct service name.

Many distributions provide broken and out-of-date versions of Liquidsoap in their repositories. For this reason (along with improved ability to customize your installation), the Liquidsoap developers recommend installing it using the OCaml Package Manager (opam). Use your distro's package manager to install opam. If you've been doing everything up to this point logged in as root, you'll now need to create a non-root user under which to install Liquidsoap. You also need to install sudo and give this new user permission to use it. On Debian-based

systems, the `adduser` and `gpasswd` utilities allow you to create users and add them to groups, respectively. On Debian-based systems, run the following commands as root to add a new user and grant it sudo access (for other systems, refer to the documentation). Let `username` represent the user name of the new user:

```
adduser username
gpasswd -a username sudo
```

Performing as your non-root user, initialize the OCaml Package Manager by running:

```
opam init
```

Answer "yes" when asked to modify your profile; this will place Liquidsoap on your path and allow it to be executed when you type its name. To apply opam changes, run:

```
eval `opam config env`
```

Next, install Liquidsoap's system dependencies:

```
opam install depext
opam depext taglib mad lame vorbis cry ssl samplerate
 ➥magic opus liquidsoap
```

Now, install liquidsoap by replacing `depext` with `install`:

```
opam install taglib mad lame vorbis cry ssl samplerate
 ➥magic opus liquidsoap
```

To set up a starting point for the station configuration and enable Liquidsoap as a service, the developers have created liquidsoap-daemon, a set of scripts for using Liquidsoap as a system service. Liquidsoap-daemon uses systemd for service management by default; therefore, it is compatible with most modern Linux distributions. To set it up, install Git using your

system's package manager, then run the following as your non-root user:

```
git clone https://github.com/savonet/liquidsoap-daemon
cd liquidsoap-daemon
./daemonize-liquidsoap.sh
```

You may be prompted to enter your user's password to authenticate sudo. Once the dæmon is installed, you'll now create a directory structure for storing music, jingles and archives of live streams in your non-root user's home directory. Run the following command:

```
mkdir -p ~/music/music1 ~/music/jingles ~/archives
```

Now, open the file main.liq in the liquidsoap-daemon directory. At this point, that file just contains:

```
output.dummy(blank())
```

This line sends no audio nowhere, which is not very interesting, so delete that line and add the following base configuration (lines starting with # are comments, so they are ignored by Liquidsoap). This base configuration sets up one music playlist with songs played in random order, jingles inserted approximately every seven songs, smart cross-fading, song requests and automatically recorded live streams. music.mp3, music.ogg and music.opus stream stored music and jingles in MP3, Ogg Vorbis and Ogg Opus formats respectively; stream.mp3, stream.ogg and stream.opus play a live stream when available, falling back to music when the live stream is down:

```
#Settings
set("server.telnet", true)
set("server.telnet.port", 1234)
set("harbor.bind_addr","0.0.0.0")
# Music playlists
music1 = playlist("~/music/music1")
# Some jingles
```

```
jingles = playlist("~/music/jingles")
# If something goes wrong, we'll play this
security = single("~/music/default.ogg")
# Start building the feed with music
radio = random([music1])
# Add the security, requests and smart crossfade
radio = fallback(track_sensitive = false,
 ➥[smart_crossfade(fallback([request.queue(id="request"),
➥radio])),security])
# Now add some jingles
radio = random(weights = [1, 7],[jingles, radio]) # This plays
# a jingle once every approximately seven songs, change 7 to
# another number to change this
# Add a skip command for the music stream
server.register(
usage="skip",
description="Skip the current song.",
"skip",
fun(_) -> begin source.skip(radio) "Done!" end
#Add support for live streams.
live =
audio_to_stereo(input.harbor("live",port=8080,password=
➥"dontpanic1764",buffer=1.0)) #dontpanic1764 is the
# password used to connect a live stream; it can (and should) be
# different from the source-password in icecast.xml.
full = fallback(track_sensitive=false,
[live,radio])
# Dump archives
file_name = '~/archives/%Y-%m-%d-%H:%M:%S$(if $(title),
➥"-$(title)","").ogg'
output.file(%vorbis,file_name,live,fallible=true)
# Stream it out
output.icecast(%mp3.vbr,
host = "localhost", port = 8000,
password = "dontpanic", mount = "music.mp3",
name="myStation Music Service", description="This is the myStation
```

```
 ➥music stream. Add some information about your station's automated
 ➥programming.",
radio)
output.icecast(%vorbis,
host = "localhost", port = 8000,
password = "dontpanic", mount = "music.ogg",
name="myStation Music Service", description="This is the myStation
 ➥music stream. Add some information about your station's
 ➥automated programming.",
radio)
output.icecast(%opus(vbr="unconstrained",bitrate=60),
host = "localhost", port = 8000,
password = "dontpanic", mount = "music.opus",
name="myStation Music Service", description="This is the myStation
 ➥music stream. Add some information about your station's
 ➥automated programming.",
radio)
output.icecast(%mp3.vbr,
host = "localhost", port = 8000,
password = "dontpanic", mount = "stream.mp3",
name="myStation Main Stream", description="The myStation main stream.",
full)
output.icecast(%vorbis,
host="localhost",port=8000,password="dontpanic",
mount="stream.ogg",
name="myStation Main Stream", description="The myStation main stream.",
full)
output.icecast(%opus(vbr="unconstrained",bitrate=60),
 ➥description="The myStation main stream.",
host="localhost",port=8000,password="dontpanic",
mount="stream.opus",
full)
```

Multiple Music Playlists

You may wish to set up multiple music playlists, perhaps with different types of music, and change the frequency at which songs from each

playlist are played. To do this, create directories under music for each playlist, named music2, music3 and so on. Then just copy the `music1` line in the music playlists section of main.liq, changing the reference to `music1` accordingly.

To insert songs randomly from the new playlist every n songs in the stream, add a line below `radio = random([music1])`, where `n` represents the approximate number of songs to play before inserting a song from the new playlist:

```
radio = random(weights = [1, n],[music2, radio])
```

Here's an example with three music playlists:

```
# Music playlists
music1 = playlist("~/music/music1")
music2 = playlist("~/music/music2")
music3 = playlist("~/music/music3")
...
radio = random([music1])
radio = random(weights = [1, 6],[music2, radio])
radio = random(weights = [1,12],[music3, radio])
```

File-Based Playlists

In the base configuration, Liquidsoap will search the directory ~/music/music1 recursively for songs to play. However, you also can give Liquidsoap a newline-delimited text file of paths to songs, either locally on your system or on the web. To do this, simply change the path to a directory to a path to your text file, like this:

```
music1 = playlist("~/music/music1.pls")
```

Sequential Playback

By default, Liquidsoap plays tracks in random order. If you want to play tracks sequentially, add `mode="sequential"` to your playlist definition, like this:

```
music1 = playlist("~/music/music1",mode="sequential")
```

Instead of using `random` (for example, when adding other playlists or jingles), use `rotate`:

```
radio = rotate(weights = [1, 7],[jingles, radio])
```

Sequential playback is best combined with file-based playlists as they both give you total control over the order in which tracks are played by Liquidsoap.

Compression and Normalization

If you'd like to add a more "radio-like" sound to your automated programming, Liquidsoap supports automatic compression and normalization. To compress and normalize the tracks of a playlist or input.harbor live stream, wrap it in an `nrj()` operator, like so:

```
music1 = nrj(playlist("~/music/music1"))
```

Talking Over Automated Programming

You can add a mountpoint allowing you to talk over the automated programming, which will have its volume reduced while you're connected. Add the following to your configuration above `#Add support for live streams`. The automated programming volume will be changed to 15% of normal while the mic is connected; change `p=0.15` to adjust:

```
# Talk over stream using microphone mount.
mic=input.harbor("mic",port=8080,password="dontpanic1764",buffer=1.0)
radio = smooth_add(delay=0.8, p=0.15, normal=radio, special=mic)
```

Finishing Up

Edit the configuration as necessary, then save and close the file. Record a file to ~/music/default.ogg; this file will be played when Liquidsoap cannot find other tracks to play. The file should tell listeners that the stream is down and give them information for contacting you to notify you of the problem. Populate the playlist(s) with music, then start Liquidsoap with the following command:

```
sudo systemctl start liquidsoap
```

Enable it on boot:

```
sudo systemctl enable liquidsoap
```

Once Liquidsoap is started, visit http://example.com:8000 in a web browser (where example.com is the fully qualified domain name of your server). If your system is configured properly, music.mp3, music.ogg and music.opus will appear, playing automated programming. Also, stream.mp3, stream.ogg and stream.opus will play automated programming unless a live stream is connected.

If Icecast appears but no mountpoints are listed, check the Liquidsoap logs at liquidsoap-daemon/log/run.log for errors. If Icecast doesn't load, restart it with `systemctl restart icecast2`.

To broadcast a live stream through your server, you will need a compatible source client. For Windows, I recommend Altacast (http://www.altacast.com/index.php/downloads). For Mac users, I suggest Ladiocast, available in the Mac App Store. For Linux, install DarkIce through your system's package manager. On iOS, I recommend iCast. On Android, I suggest Cool Mic. In all cases, use the following configuration:

- Host: your server's fully qualified domain name.
- Port: 8080
- Mount (mountpoint): live (or /live) for a live stream. If you enabled the ability to talk over automated programming, replace `live` with `mic` to talk over the music.
- username: source (some clients don't prompt for a user name, in which case, source is the implied default).
- password: dontpanic1764 (or the password you specified in the input.harbor configuration).

You can stream in Ogg Vorbis or MP3. Ogg Opus may or may not work, depending on your source client.

In addition to Liquidsoap's telnet interface, Icecast also has a web-based administrative interface that you can use to view listener statistics, kill Liquidsoap's streams or move listeners among mountpoints.

Liquidsoap offers control via TCP (over telnet or similar). The base configuration presented in this article enables a song request system and the ability to skip tracks on demand. By default, this interface is available only to users on the local system. The telnet protocol does not support authentication. If you want to make song request functionality available to your users, you'll need to write a program or script customized for your station that interfaces with Liquidsoap.

Connect to Liquidsoap via telnet, like so:

```
telnet localhost 1234
```

Once connected, you can request a song with the following, where `uri` is an absolute path to an audio file on your system or a URL of an audio file on the internet:

```
request.push uri
```

To skip the currently playing song and immediately play the next one, simply type `skip`.

For a list of all available commands, type `help`, or type `help` followed by the name of a command for usage information on a particular command.

To end your session, type `quit`.

In addition to Liquidsoap's telnet interface, Icecast also has a web-based administrative interface that you can use to view listener statistics, kill Liquidsoap's streams or move listeners among

mountpoints. Access it at http://example.com:8000/admin (where example.com is your server's fully qualified domain name). Use the admin-user and admin-password you set in icecast.xml.

At this point, you now have a fully functional streaming server that should fit the needs of most users. However, Liquidsoap is extremely flexible, allowing for more exotic setups for special use cases. Refer to the Liquidsoap documentation (http://liquidsoap.fm/doc-dev/reference.html) for information on additional language features that may be useful to you.■

Bill Dengler has been a Linux user and tinkerer since age nine. He was born totally blind due to a rare genetic condition called Norrie Disease, so he relies on a screen reader to access his computer. He is currently pursuing an International Baccalaureate diploma. Feel free to send him questions and comments at codeofdusk@gmail.com.

SUPERMICRO® MARKETPLACE

Powered by Silicon Mechanics

Broad Selection

Zero Defects

3-Year Warranty

Your Source for Supermicro Platform Technology

Twin, TwinPro & BigTwin
High-density, high-value servers

Configure Now

FatTwin
Highest performance per watt

Configure Now

SuperStorage
Flexible and efficient storage

Configure Now

1U Servers
Entry & enterprise1U form factor servers

Configure Now

2U Servers
Flexible 2U form factor servers

Configure Now

3U+ Servers
All 3U and larger form factor servers

Configure Now

Ultra Servers
Unrivaled performance, flexibility & scalability

Configure Now

MP Servers
Servers based on the Intel® Xeon® E7 Product Family

Configure Now

SuperWorkstations
Server-grade performance at your desk

Configure Now

Talk to a Supermicro Expert! 866.352.1173

Linux Filesystem Events with inotify

Triggering scripts with incron and systemd.

CHARLES FISHER

PREVIOUS
Feature: Creating an Internet Radio Station with Icecast and Liquidsoap

NEXT
Doc Searls' EOF

It is, at times, important to know when things change in the Linux OS. The uses to which systems are placed often include high-priority data that must be processed as soon as it is seen. The conventional method of finding and processing new file data is to poll for it, usually with cron. This is inefficient, and it can tax performance unreasonably if too many polling events are forked too often.

Linux has an efficient method for alerting user-space processes to changes impacting files of interest. The inotify Linux system calls were first discussed here in *Linux Journal* in a 2005 article by Robert Love (http://www.linuxjournal.com/article/8478), who primarily addressed the behavior of the new features from the perspective of C.

However, there also are stable shell-level utilities and new classes of monitoring dæmons for registering filesystem watches and reporting events. Linux installations using systemd also can access basic inotify functionality with path units. The inotify interface does have limitations—it can't monitor remote, network-mounted filesystems (that is, NFS); it does not report the userid involved in the event; it does not work with /proc or other pseudo-filesystems; and mmap() operations do not trigger it, among other concerns. Even with these limitations, it is a tremendously useful feature.

This article completes the work begun by Love and gives everyone who can write a Bourne shell script or set a crontab the ability to react to filesystem changes.

The inotifywait Utility

Working under Oracle Linux 7 (or similar versions of Red Hat/CentOS/Scientific Linux), the inotify shell tools are not installed by default, but you can load them with yum:

```
 # yum install inotify-tools
Loaded plugins: langpacks, ulninfo
ol7_UEKR4                                          | 1.2 kB   00:00
ol7_latest                                         | 1.4 kB   00:00
Resolving Dependencies
--> Running transaction check
---> Package inotify-tools.x86_64 0:3.14-8.el7 will be installed
--> Finished Dependency Resolution
```

```
Dependencies Resolved

================================================================
Package         Arch       Version        Repository     Size
================================================================
Installing:
inotify-tools   x86_64     3.14-8.el7     ol7_latest     50 k

Transaction Summary
================================================================
Install  1 Package

Total download size: 50 k
Installed size: 111 k
Is this ok [y/d/N]: y
Downloading packages:
inotify-tools-3.14-8.el7.x86_64.rpm               |  50 kB   00:00
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
Warning: RPMDB altered outside of yum.
  Installing : inotify-tools-3.14-8.el7.x86_64                 1/1
  Verifying  : inotify-tools-3.14-8.el7.x86_64                 1/1

Installed:
  inotify-tools.x86_64 0:3.14-8.el7

Complete!
```

The package will include two utilities (inotifywait and inotifywatch), documentation and a number of libraries. The inotifywait program is of primary interest.

Some derivatives of Red Hat 7 may not include inotify in their base repositories. If you find it missing, you can obtain it from Fedora's EPEL

repository (https://fedoraproject.org/wiki/EPEL), either by downloading the inotify RPM for manual installation or adding the EPEL repository to yum.

Any user on the system who can launch a shell may register watches—no special privileges are required to use the interface. This example watches the /tmp directory:

```
$ inotifywait -m /tmp
Setting up watches.
Watches established.
```

If another session on the system performs a few operations on the files in /tmp:

```
$ touch /tmp/hello
$ cp /etc/passwd /tmp
$ rm /tmp/passwd
$ touch /tmp/goodbye
$ rm /tmp/hello /tmp/goodbye
```

those changes are immediately visible to the user running inotifywait:

```
/tmp/ CREATE hello
/tmp/ OPEN hello
/tmp/ ATTRIB hello
/tmp/ CLOSE_WRITE,CLOSE hello
/tmp/ CREATE passwd
/tmp/ OPEN passwd
/tmp/ MODIFY passwd
/tmp/ CLOSE_WRITE,CLOSE passwd
/tmp/ DELETE passwd
/tmp/ CREATE goodbye
/tmp/ OPEN goodbye
/tmp/ ATTRIB goodbye
/tmp/ CLOSE_WRITE,CLOSE goodbye
/tmp/ DELETE hello
/tmp/ DELETE goodbye
```

A few relevant sections of the manual page explain what is happening:

```
$ man inotifywait | col -b | sed -n '/diagnostic/,/helpful/p'
  inotifywait will output diagnostic information on standard error and
  event information on standard output. The event output can be config-
  ured, but by default it consists of lines of the following form:

  watched_filename EVENT_NAMES event_filename


  watched_filename
    is the name of the file on which the event occurred. If the
    file is a directory, a trailing slash is output.

  EVENT_NAMES
    are the names of the inotify events which occurred, separated by
    commas.

  event_filename
    is output only when the event occurred on a directory, and in
    this case the name of the file within the directory which caused
    this event is output.

    By default, any special characters in filenames are not escaped
    in any way. This can make the output of inotifywait difficult
    to parse in awk scripts or similar. The --csv and --format
    options will be helpful in this case.
```

It also is possible to filter the output by registering particular events of interest with the `-e` option, the list of which is shown here:

access	create	move_self
attrib	delete	moved_to
close_write	delete_self	moved_from
close_nowrite	modify	open
close	move	unmount

A common application is testing for the arrival of new files. Since inotify must be given the name of an existing filesystem object to watch, the directory containing the new files is provided. A trigger of interest is also easy to provide—new files should be complete and ready for processing when the `close_write` trigger fires. Below is an example script to watch for these events:

```
#!/bin/sh
unset IFS                                    # default of space, tab and nl
                                             # Wait for filesystem events
inotifywait -m -e close_write \
   /tmp /var/tmp /home/oracle/arch-orcl/ |
while read dir op file
do [[ "${dir}" == '/tmp/' && "${file}" == *.txt ]] &&
      echo "Import job should start on $file ($dir $op)."

   [[ "${dir}" == '/var/tmp/' && "${file}" == CLOSE_WEEK*.txt ]] &&
      echo Weekly backup is ready.

   [[ "${dir}" == '/home/oracle/arch-orcl/' && "${file}" == *.ARC ]] &&
      su - oracle -c 'ORACLE_SID=orcl ~oracle/bin/log_shipper' &

   [[ "${dir}" == '/tmp/' && "${file}" == SHUT ]] && break

   ((step+=1))
done

echo We processed $step events.
```

There are a few problems with the script as presented—of all the available shells on Linux, only ksh93 (that is, the AT&T Korn shell) will report the "step" variable correctly at the end of the script. All the other shells will report this variable as null.

The reason for this behavior can be found in a brief explanation on the manual page for Bash: "Each command in a pipeline is executed as a separate process (i.e., in a subshell)." The MirBSD clone of the Korn shell

has a slightly longer explanation:

```
# man mksh | col -b | sed -n '/The parts/,/do so/p'
  The parts of a pipeline, like below, are executed in subshells. Thus,
  variable assignments inside them fail. Use co-processes instead.

  foo | bar | read baz          # will not change $baz
  foo | bar |& read -p baz      # will, however, do so
```

And, the pdksh documentation in Oracle Linux 5 (from which MirBSD mksh emerged) has several more mentions of the subject:

```
General features of at&t ksh88 that are not (yet) in pdksh:
  - the last command of a pipeline is not run in the parent shell
  - `echo foo | read bar; echo $bar' prints foo in at&t ksh, nothing
    in pdksh (ie, the read is done in a separate process in pdksh).
  - in pdksh, if the last command of a pipeline is a shell builtin, it
    is not executed in the parent shell, so "echo a b | read foo bar"
    does not set foo and bar in the parent shell (at&t ksh will).
    This may get fixed in the future, but it may take a while.

$ man pdksh | col -b | sed -n '/BTW, the/,/aware/p'
  BTW, the most frequently reported bug is
    echo hi | read a; echo $a   # Does not print hi
  I'm aware of this and there is no need to report it.
```

This behavior is easy enough to demonstrate—running the script above with the default bash shell and providing a sequence of example events:

```
$ cp /etc/passwd /tmp/newdata.txt
$ cp /etc/group /var/tmp/CLOSE_WEEK20170407.txt
$ cp /etc/passwd /tmp/SHUT
```

gives the following script output:

```
# ./inotify.sh
```

```
Setting up watches.
Watches established.
Import job should start on newdata.txt (/tmp/ CLOSE_WRITE,CLOSE).
Weekly backup is ready.
We processed events.
```

Examining the process list while the script is running, you'll also see two shells, one forked for the control structure:

```
$ function pps { typeset a IFS=\| ; ps ax | while read a
do case $a in *$1*|+([!0-9])) echo $a;; esac; done }


$ pps inot
  PID TTY      STAT   TIME COMMAND
 3394 pts/1    S+     0:00 /bin/sh ./inotify.sh
 3395 pts/1    S+     0:00 inotifywait -m -e close_write /tmp /var/tmp
 3396 pts/1    S+     0:00 /bin/sh ./inotify.sh
```

As it was manipulated in a subshell, the "step" variable above was null when control flow reached the echo. Switching this from #/bin/sh to #/bin/ksh93 will correct the problem, and only one shell process will be seen:

```
# ./inotify.ksh93
Setting up watches.
Watches established.
Import job should start on newdata.txt (/tmp/ CLOSE_WRITE,CLOSE).
Weekly backup is ready.
We processed 2 events.


$ pps inot
  PID TTY      STAT   TIME COMMAND
 3583 pts/1    S+     0:00 /bin/ksh93 ./inotify.sh
 3584 pts/1    S+     0:00 inotifywait -m -e close_write /tmp /var/tmp
```

Although ksh93 behaves properly and in general handles scripts far more gracefully than all of the other Linux shells, it is rather large:

```
$ ll /bin/[bkm]+([aksh93]) /etc/alternatives/ksh
-rwxr-xr-x. 1 root root  960456 Dec  6 11:11 /bin/bash
lrwxrwxrwx. 1 root root      21 Apr  3 21:01 /bin/ksh ->
                                              /etc/alternatives/ksh
-rwxr-xr-x. 1 root root 1518944 Aug 31  2016 /bin/ksh93
-rwxr-xr-x. 1 root root  296208 May  3  2014 /bin/mksh
lrwxrwxrwx. 1 root root      10 Apr  3 21:01 /etc/alternatives/ksh ->
                                                  /bin/ksh93
```

The mksh binary is the smallest of the Bourne implementations above (some of these shells may be missing on your system, but you can install them with yum). For a long-term monitoring process, mksh is likely the best choice for reducing both processing and memory footprint, and it does not launch multiple copies of itself when idle assuming that a coprocess is used. Converting the script to use a Korn coprocess that is friendly to mksh is not difficult:

```
#!/bin/mksh
unset IFS                                  # default of space, tab and nl
                                           # Wait for filesystem events
inotifywait -m -e close_write \
   /tmp/ /var/tmp/ /home/oracle/arch-orcl/ \
   2>/dev/null |&                          # Launch as Korn coprocess

while read -p dir op file                  # Read from Korn coprocess
do [[ "${dir}" == '/tmp/' && "${file}" == *.txt ]] &&
      print "Import job should start on $file ($dir $op)."

   [[ "${dir}" == '/var/tmp/' && "${file}" == CLOSE_WEEK*.txt ]] &&
      print Weekly backup is ready.

   [[ "${dir}" == '/home/oracle/arch-orcl/' && "${file}" == *.ARC ]] &&
      su - oracle -c 'ORACLE_SID=orcl ~oracle/bin/log_shipper' &
```

```
    [[ "${dir}" == '/tmp/' && "${file}" == SHUT ]] && break

    ((step+=1))
done

echo We processed $step events.
```

Note that The Korn and Bolsky reference on the Korn shell outlines the following requirements in a program operating as a coprocess (https://www.amazon.com/New-KornShell-Command-Programming-Language/dp/0131827006):

Caution: The co-process must:

- Send each output message to standard output.
- Have a Newline at the end of each message.
- Flush its standard output whenever it writes a message.

An `fflush(NULL)` is found in the main processing loop of the inotifywait source, and these requirements appear to be met.

The mksh version of the script is the most reasonable compromise for efficient use and correct behavior, and I have explained it at some length here to save readers trouble and frustration—it is important to avoid control structures executing in subshells in most of the Borne family. However, hopefully all of these ersatz shells someday fix this basic flaw and implement the Korn behavior correctly.

A Practical Application—Oracle Log Shipping

Oracle databases that are configured for hot backups produce a stream of "archived redo log files" that are used for database recovery. These are the most critical backup files that are produced in an Oracle database.

These files are numbered sequentially and are written to a log

directory configured by the DBA. An inotifywatch can trigger activities to compress, encrypt and/or distribute the archived logs to backup and disaster recovery servers for safekeeping. You can configure Oracle RMAN to do most of these functions, but the OS tools are more capable, flexible and simpler to use.

There are a number of important design parameters for a script handling archived logs:

- A "critical section" must be established that allows only a single process to manipulate the archived log files at a time. Oracle will sometimes write bursts of log files, and inotify might cause the handler script to be spawned repeatedly in a short amount of time. Only one instance of the handler script can be allowed to run—any others spawned during the handler's lifetime must immediately exit. This will be achieved with a textbook application of the flock program from the util-linux package.

- The optimum compression available for production applications appears to be lzip (http://www.nongnu.org/lzip). The author claims that the integrity of his archive format is superior to many more well known utilities, both in compression ability and also structural integrity (http://www.nongnu.org/lzip/xz_inadequate.html). The lzip binary is not in the standard repository for Oracle Linux—it is available in EPEL and is easily compiled from source.

- Note that 7-Zip (http://www.7-zip.org) uses the same LZMA algorithm as lzip, and it also will perform AES encryption on the data after compression. Encryption is a desirable feature, as it will exempt a business from breach disclosure laws (http://www.ncsl.org/research/telecommunications-and-information-technology/security-breach-notification-laws.aspx) in most US states if the backups are lost or stolen and they contain "Protected Personal Information" (PPI), such as birthdays or Social Security Numbers. The author of lzip does have harsh things to say regarding the quality of 7-Zip archives using LZMA2, and the `openssl enc` program can be used to apply AES encryption after compression to lzip archives

or any other type of file, as I discussed in a previous article (http://www.linuxjournal.com/content/flat-file-encryption-openssl-and-gpg). I'm foregoing file encryption in the script below and using lzip for clarity.

- The current log number will be recorded in a dot file in the Oracle user's home directory. If a log is skipped for some reason (a rare occurrence for an Oracle database), log shipping will stop. A missing log requires an immediate and full database backup (either cold or hot)—successful recoveries of Oracle databases cannot skip logs.

- The `scp` program will be used to copy the log to a remote server, and it should be called repeatedly until it returns successfully.

- I'm calling the genuine '93 Korn shell for this activity, as it is the most capable scripting shell and I don't want any surprises.

Given these design parameters, this is an implementation:

```
# cat ~oracle/archutils/process_logs

#!/bin/ksh93

set -euo pipefail
IFS=$'\n\t'  # http://redsymbol.net/articles/unofficial-bash-strict-mode/

(
 flock -n 9 || exit 1          # Critical section-allow only one process.

 ARCHDIR=~oracle/arch-${ORACLE_SID}

 APREFIX=${ORACLE_SID}_1_

 ASUFFIX=.ARC
```

```
 CURLOG=$(<~oracle/.curlog-$ORACLE_SID)

 File="${ARCHDIR}/${APREFIX}${CURLOG}${ASUFFIX}"

 [[ ! -f "$File" ]] && exit

 while [[ -f "$File" ]]
 do ((NEXTCURLOG=CURLOG+1))

    NextFile="${ARCHDIR}/${APREFIX}${NEXTCURLOG}${ASUFFIX}"

    [[ ! -f "$NextFile" ]] && sleep 60  # Ensure ARCH has finished

    nice /usr/local/bin/lzip -9q "$File"

    until scp "${File}.lz" "yourcompany.com:~oracle/arch-$ORACLE_SID"
    do sleep 5
    done

    CURLOG=$NEXTCURLOG

    File="$NextFile"
 done

 echo $CURLOG > ~oracle/.curlog-$ORACLE_SID

) 9>~oracle/.processing_logs-$ORACLE_SID
```

The above script can be executed manually for testing even while the inotify handler is running, as the flock protects it.

A standby server, or a DataGuard server in primitive standby mode, can apply the archived logs at regular intervals. The script below forces a 12-hour delay in log application for the recovery of dropped or damaged objects, so inotify cannot be easily used in this case—cron is a more reasonable approach for delayed file processing, and a run every 20 minutes will keep the standby at the

desired recovery point:

```
# cat ~oracle/archutils/delay-lock.sh

#!/bin/ksh93

(
 flock -n 9 || exit 1              # Critical section-only one process.

 WINDOW=43200                      # 12 hours

 LOG_DEST=~oracle/arch-$ORACLE_SID

 OLDLOG_DEST=$LOG_DEST-applied

 function fage { print $(( $(date +%s) - $(stat -c %Y "$1") ))
  } # File age in seconds - Requires GNU extended date & stat

 cd $LOG_DEST

 of=$(ls -t | tail -1)             # Oldest file in directory

 [[ -z "$of" || $(fage "$of") -lt $WINDOW ]] && exit

 for x in $(ls -rt)                    # Order by ascending file mtime
 do if [[ $(fage "$x") -ge $WINDOW ]]
    then y=$(basename $x .lz)          # lzip compression is optional

         [[ "$y" != "$x" ]] && /usr/local/bin/lzip -dkq "$x"

         $ORACLE_HOME/bin/sqlplus '/ as sysdba' > /dev/null 2>&1 <<-EOF
           recover standby database;
           $LOG_DEST/$y
           cancel
           quit
           EOF
```

```
        [[ "$y" != "$x" ]] && rm "$y"

        mv "$x" $OLDLOG_DEST
    fi

 done
) 9> ~oracle/.recovering-$ORACLE_SID
```

I've covered these specific examples here because they introduce tools to control concurrency, which is a common issue when using inotify, and they advance a few features that increase reliability and minimize storage requirements. Hopefully enthusiastic readers will introduce many improvements to these approaches.

The incron System

Lukas Jelinek is the author of the incron package that allows users to specify tables of inotify events that are executed by the master incrond process. Despite the reference to "cron", the package does not schedule events at regular intervals—it is a tool for filesystem events, and the cron reference is slightly misleading.

The incron package is available from EPEL. If you have installed the repository, you can load it with yum:

```
# yum install incron
Loaded plugins: langpacks, ulninfo
Resolving Dependencies
--> Running transaction check
---> Package incron.x86_64 0:0.5.10-8.el7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================
 Package       Arch       Version           Repository    Size
================================================================
Installing:
```

```
 incron        x86_64     0.5.10-8.el7      epel          92 k

Transaction Summary
================================================================
Install  1 Package

Total download size: 92 k
Installed size: 249 k
Is this ok [y/d/N]: y
Downloading packages:
incron-0.5.10-8.el7.x86_64.rpm                        |  92 kB   00:01
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : incron-0.5.10-8.el7.x86_64                          1/1
  Verifying  : incron-0.5.10-8.el7.x86_64                          1/1

Installed:
  incron.x86_64 0:0.5.10-8.el7

Complete!
```

On a systemd distribution with the appropriate service units, you can start and enable incron at boot with the following commands:

```
# systemctl start incrond
# systemctl enable incrond
Created symlink from
   /etc/systemd/system/multi-user.target.wants/incrond.service
to /usr/lib/systemd/system/incrond.service.
```

In the default configuration, any user can establish incron schedules. The incrontab format uses three fields:

```
<path> <mask> <command>
```

Below is an example entry that was set with the `-e` option:

```
$ incrontab -e        #vi session follows

$ incrontab -l
/tmp/ IN_ALL_EVENTS /home/luser/myincron.sh $@ $% $#
```

You can record a simple script and mark it with execute permission:

```
$ cat myincron.sh
#!/bin/sh

echo -e "path: $1 op: $2 \t file: $3" >> ~/op

$ chmod 755 myincron.sh
```

Then, if you repeat the original /tmp file manipulations at the start of this article, the script will record the following output:

```
$ cat ~/op

path: /tmp/ op: IN_ATTRIB       file: hello
path: /tmp/ op: IN_CREATE       file: hello
path: /tmp/ op: IN_OPEN         file: hello
path: /tmp/ op: IN_CLOSE_WRITE  file: hello
path: /tmp/ op: IN_OPEN         file: passwd
path: /tmp/ op: IN_CLOSE_WRITE  file: passwd
path: /tmp/ op: IN_MODIFY       file: passwd
path: /tmp/ op: IN_CREATE       file: passwd
path: /tmp/ op: IN_DELETE       file: passwd
path: /tmp/ op: IN_CREATE       file: goodbye
path: /tmp/ op: IN_ATTRIB       file: goodbye
path: /tmp/ op: IN_OPEN         file: goodbye
path: /tmp/ op: IN_CLOSE_WRITE  file: goodbye
path: /tmp/ op: IN_DELETE       file: hello
path: /tmp/ op: IN_DELETE       file: goodbye
```

While the `IN_CLOSE_WRITE` event on a directory object is usually of greatest interest, most of the standard inotify events are available within incron, which also offers several unique amalgams:

```
$ man 5 incrontab | col -b | sed -n '/EVENT SYMBOLS/,/child process/p'

EVENT SYMBOLS

These basic event mask symbols are defined:

IN_ACCESS           File was accessed (read) (*)
IN_ATTRIB           Metadata changed (permissions, timestamps, extended
                    attributes, etc.) (*)
IN_CLOSE_WRITE      File opened for writing was closed (*)
IN_CLOSE_NOWRITE    File not opened for writing was closed (*)
IN_CREATE           File/directory created in watched directory (*)
IN_DELETE           File/directory deleted from watched directory (*)
IN_DELETE_SELF      Watched file/directory was itself deleted
IN_MODIFY           File was modified (*)
IN_MOVE_SELF        Watched file/directory was itself moved
IN_MOVED_FROM       File moved out of watched directory (*)
IN_MOVED_TO         File moved into watched directory (*)
IN_OPEN             File was opened (*)

When monitoring a directory, the events marked with an asterisk (*)
above can occur for files in the directory, in which case the name
field in the returned event data identifies the name of the file within
the directory.

The IN_ALL_EVENTS symbol is defined as a bit mask of all of the above
events. Two additional convenience symbols are IN_MOVE, which is a com-
bination of IN_MOVED_FROM and IN_MOVED_TO, and IN_CLOSE, which combines
IN_CLOSE_WRITE and IN_CLOSE_NOWRITE.

The following further symbols can be specified in the mask:
```

```
IN_DONT_FOLLOW     Don't dereference pathname if it is a symbolic link
IN_ONESHOT    Monitor pathname for only one event
IN_ONLYDIR    Only watch pathname if it is a directory

Additionally, there is a symbol which doesn't appear in the inotify sym-
bol set. It is IN_NO_LOOP. This symbol disables monitoring events until
the current one is completely handled (until its child process exits).
```

The incron system likely presents the most comprehensive interface to inotify of all the tools researched and listed here. Additional configuration options can be set in /etc/incron.conf to tweak incron's behavior for those that require a non-standard configuration.

Path Units under systemd

When your Linux installation is running systemd as PID 1, limited inotify functionality is available through "path units" as is discussed in a lighthearted article by Paul Brown at *OCS-Mag* (http://www.ocsmag.com/2015/09/02/monitoring-file-access-for-dummies).

The relevant manual page has useful information on the subject:

```
$ man systemd.path | col -b | sed -n '/Internally,/,/systems./p'

Internally, path units use the inotify(7) API to monitor file systems.
Due to that, it suffers by the same limitations as inotify, and for
example cannot be used to monitor files or directories changed by other
machines on remote NFS file systems.
```

Note that when a systemd path unit spawns a shell script, the `$HOME` and tilde (~) operator for the owner's home directory may not be defined. Using the tilde operator to reference another user's home directory (for example, ~nobody/) does work, even when applied to the self-same user running the script. The Oracle script above was explicit and did not reference ~ without specifying the target user, so I'm using it as an example here.

Using inotify triggers with systemd path units requires two files.

The first file specifies the filesystem location of interest:

```
$ cat /etc/systemd/system/oralog.path

[Unit]
Description=Oracle Archivelog Monitoring
Documentation=http://docs.yourserver.com

[Path]
PathChanged=/home/oracle/arch-orcl/

[Install]
WantedBy=multi-user.target
```

The `PathChanged` parameter above roughly corresponds to the `close-write` event used in my previous direct inotify calls. The full collection of inotify events is not (currently) supported by systemd—it is limited to `PathExists`, `PathChanged` and `PathModified`, which are described in `man systemd.path`.

The second file is a service unit describing a program to be executed. It must have the same name, but a different extension, as the path unit:

```
$ cat /etc/systemd/system/oralog.service

[Unit]
Description=Oracle Archivelog Monitoring
Documentation=http://docs.yourserver.com

[Service]
Type=oneshot
Environment=ORACLE_SID=orcl
ExecStart=/bin/sh -c '/root/process_logs >> /tmp/plog.txt 2>&1'
```

The `oneshot` parameter above alerts systemd that the program that it forks is expected to exit and should not be respawned automatically—the

restarts are limited to triggers from the path unit. The above service configuration will provide the best options for logging—divert them to /dev/null if they are not needed.

Use `systemctl start` on the path unit to begin monitoring—a common error is using it on the service unit, which will directly run the handler only once. Enable the path unit if the monitoring should survive a reboot.

Although this limited functionality may be enough for some casual uses of inotify, it is a shame that the full functionality of inotifywait and incron are not represented here. Perhaps it will come in time.

Conclusion

Although the inotify tools are powerful, they do have limitations. To repeat them, inotify cannot monitor remote (NFS) filesystems; it cannot report the userid involved in a triggering event; it does not work with /proc or other pseudo-filesystems; mmap() operations do not trigger it; and the inotify queue can overflow resulting in lost events, among other concerns.

Even with these weaknesses, the efficiency of inotify is superior to most other approaches for immediate notifications of filesystem activity. It also is quite flexible, and although the close-write directory trigger should suffice for most usage, it has ample tools for covering special use cases.

In any event, it is productive to replace polling activity with inotify watches, and system administrators should be liberal in educating the user community that the classic crontab is not an appropriate place to check for new files. Recalcitrant users should be confined to Ultrix on a VAX until they develop sufficient appreciation for modern tools and approaches, which should result in more efficient Linux systems and happier administrators. ■

Charles Fisher has an electrical engineering degree from the University of Iowa and works as a systems and database administrator for a Fortune 500 mining and manufacturing corporation. He has previously published both journal articles and technical manuals on Linux for UnixWorld and other McGraw-Hill publications.

Archiving /etc/passwd

Tracking changes to the password file involves many different types of inotify triggering events. The `vipw` utility commonly will make changes to a temporary file, then clobber the original with it. This can be seen when the inode number changes:

```
# ll -i /etc/passwd
199720973 -rw-r--r-- 1 root root 3928 Jul  7 12:24 /etc/passwd

# vipw
[ make changes ]
You are using shadow passwords on this system.
Would you like to edit /etc/shadow now [y/n]? n

# ll -i /etc/passwd
203784208 -rw-r--r-- 1 root root 3956 Jul  7 12:24 /etc/passwd
```

The destruction and replacement of /etc/passwd even occurs with setuid binaries called by unprivileged users:

```
$ ll -i /etc/passwd
203784196 -rw-r--r-- 1 root root 3928 Jun 29 14:55 /etc/passwd

$ chsh
Changing shell for fishecj.
Password:
New shell [/bin/bash]: /bin/csh
Shell changed.

$ ll -i /etc/passwd
199720970 -rw-r--r-- 1 root root 3927 Jul  7 12:23 /etc/passwd
```

For this reason, all inotify triggering events should be considered when

tracking this file. If there is concern with an inotify queue overflow (in which events are lost), then the OPEN, ACCESS and CLOSE_NOWRITE,CLOSE triggers likely can be immediately ignored.

All other inotify events on /etc/passwd might run the following script to version the changes into an RCS archive and mail them to an administrator:

```
#!/bin/sh

# This script tracks changes to the /etc/passwd file from inotify.
# Uses RCS for archiving. Watch for UID zero.

PWMAILS=Charlie.Root@openbsd.org

TPDIR=~/track_passwd

cd $TPDIR

if diff -q /etc/passwd $TPDIR/passwd
then exit                                          # they are the same
else sleep 5                                       # let passwd settle
     diff /etc/passwd $TPDIR/passwd 2>&1 |         # they are DIFFERENT
     mail -s "/etc/passwd changes $(hostname -s)" "$PWMAILS"
     cp -f /etc/passwd $TPDIR                      # copy for checkin

#    "SCCS, the source motel! Programs check in and never check out!"
#     -- Ken Thompson

     rcs -q -l passwd                              # lock the archive
     ci -q -m_ passwd                              # check in new ver
     co -q passwd                                  # drop the new copy
fi > /dev/null 2>&1
```

Here is an example email from the script for the above `chfn` operation:

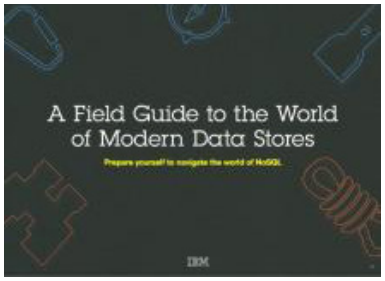
```
-----Original Message-----
From: root [mailto:root@myhost.com]
Sent: Thursday, July 06, 2017 2:35 PM
To: Fisher, Charles J. <Charles.Fisher@myhost.com>;
Subject: /etc/passwd changes myhost

57c57
< fishecj:x:123:456:Fisher, Charles J.:/home/fishecj:/bin/bash
---
> fishecj:x:123:456:Fisher, Charles J.:/home/fishecj:/bin/csh
```

Further processing on the third column of /etc/passwd might detect UID zero (a root user) or other important user classes for emergency action. This might include a rollback of the file from RCS to /etc and/or SMS messages to security contacts.

Send comments or feedback via http://www.linuxjournal.com/contact or to ljeditor@linuxjournal.com.

RETURN TO CONTENTS



A Field Guide to the World of Modern Data Stores

There are many types of databases and data analysis tools to choose from when building your application. Should you use a relational database? How about a key-value store? Maybe a document database? Is a graph database the right fit? What about polyglot persistence and the need for advanced analytics?

If you feel a bit overwhelmed, don't worry. This guide lays out the various database options and analytic solutions available to meet your app's unique needs.

You'll see how data can move across databases and development languages, so you can work in your favorite environment without the friction and productivity loss of the past.

Sponsor: IBM

> **https://geekguide.linuxjournal.com/content/field-guide-world-modern-data-stores**

Why NoSQL? Your database options in the new non-relational world

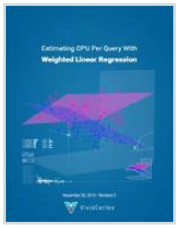
The continual increase in web, mobile and IoT applications, alongside emerging trends shifting online consumer behavior and new classes of data, is causing developers to reevaluate how their data is stored and managed. Today's applications require a database that is capable of providing a scalable, flexible solution to efficiently and safely manage the massive flow of data to and from a global user base.

Developers and IT alike are finding it difficult, and sometimes even impossible, to quickly incorporate all of this data into the relational model while dynamically scaling to maintain the performance levels users demand. This is causing many to look at NoSQL databases for the flexibility they offer, and is a big reason why the global NoSQL market is forecasted to nearly double and reach USD3.4 billion in 2020.

Sponsor: IBM

> **https://geekguide.linuxjournal.com/content/why-nosql-your-database-options-new-non-relational-world**

Estimating CPU Per Query With Weighted Linear Regression



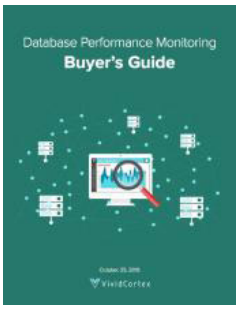
Your database server is suddenly using a lot of CPU resources. Quick, what caused it? This is a familiar question for engineers of all persuasions. And it's often impossible to answer.

There are good reasons why it's hard to figure out what consumes resources like CPU, IO, and memory in a complex piece of software such as a database. The first problem is that most database server software doesn't offer any way to measure or inspect that type of performance data. The database server isn't observable. This problem arises in turn from the complexity of the database server software and the way it does its work, which actually precludes measuring resource consumption accurately!

Author: Baron Schwartz

Sponsor: VividCortex

> **https://geekguide.linuxjournal.com/content/estimating-cpu-query-weighted-linear-regression**



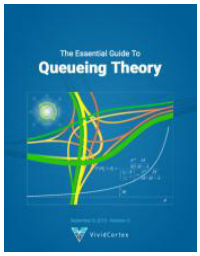
Database Performance Monitoring Buyer's Guide

More and more companies have begun to recognize database performance management as a vital need. Despite its widespread importance, good database performance management requires specialized expertise with custom approaches--yet all too often, organizations rely on one-size-fits-all solutions that theoretically "check the box" but in practice do little or nothing to help them find or prevent database-related outages and performance problems.

This buyer's guide is designed to help you understand what database management really requires, so your investments in a solution provide the greatest possible ultimate value.

Sponsor: VividCortex

> **https://geekguide.linuxjournal.com/content/database-performance-monitoring-buyer%E2%80%99s-guide**



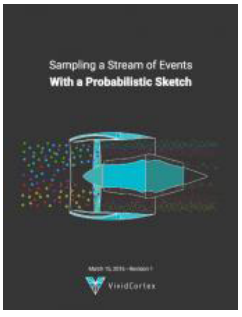
The Essential Guide To Queueing Theory

Whether you're an entrepreneur, engineer, or manager, learning about queueing theory is a great way to be more effective. Queueing theory is fundamental to getting good return on your efforts. That's because the results your systems and teams produce are heavily influenced by how much waiting takes place, and waiting is waste. Minimizing this waste is extremely important. It's one of the biggest levers you will find for improving the cost and performance of your teams and systems.

Author: Baron Schwartz

Sponsor: VividCortex

> **https://geekguide.linuxjournal.com/content/essential-guide-queueing-theory**



Sampling a Stream of Events With a Probabilistic Sketch

Stream processing is a hot topic today. As modern Big Data processing systems have evolved, stream processing has become recognized as a first-class citizen in the toolbox. That's because when you take away the how of Big Data and look at the underlying goals and end results, deriving real-time insights from huge, high-velocity, high-variety streams of data is a fundamental, core use case. This explains the explosive popularity of systems such as Apache Kafka, Apache Spark, Apache Samza, Apache Storm, and Apache Apex—to name just a few!

Author: Baron Schwartz

Sponsor: VividCortex

> **https://geekguide.linuxjournal.com/content/sampling-stream-events-probabilistic-sketch**

The Actually Distributed Web

Maybe now we can make what we should have made with the web in the first place.

DOC SEARLS

Doc Searls is Senior Editor of *Linux Journal*. He is also a fellow with the Berkman Center for Internet and Society at Harvard University and the Center for Information Technology and Society at UC Santa Barbara.

PREVIOUS
Feature: Linux Filesystem Events with inotify

I thought my mind was through getting blown until I heard in mid-June 2017 that Brave (https://brave.com) raised $35 million in less than 30 seconds (https://techcrunch.com/2017/06/01/brave-ico-35-million-30-seconds-brendan-eich), though an ICO (Initial Coin Offering: http://www.investopedia.com/terms/i/initial-coin-offering-ico.asp). I did know ICOs were hot stuff. I also knew Brave's ICO was about to happen, because Brendan Eich (https://en.wikipedia.org/wiki/Brendan_Eich), the company CEO, said so over breakfast two days earlier. So my seat belt was fastened, but the acceleration of the ICO still left my mental ass on the pavement two counties back.

Since then, I've hyper-focused on cryptocurrencies (https://en.wikipedia.org/wiki/Cryptocurrency), tokens

Figure 1. Crypto Currency Market Capitalizations (http://coinmarketcap.com/charts)

(http://tokenfactory.io/smart-beta), distributed ledgers (https://en.wikipedia.org/wiki/Distributed_ledger), ICOs and the rest of it for two reasons. One is that there is a craze going on. See Figure 1.

The other is that the investment here includes a measure of faith that we can once again imagine full agency for individuals as distributed peers on the internet, and that many positive personal, social, economic, political and other transformations will arise from that agency.

Phil Windley, who now chairs the Sovrin Foundation (https://sovrin.org), told me yesterday that this is the third tech revolution of his lifetime. "The first was the PC, and the second was the Internet. This is the third", he said. I'm inclined to agree, simply because so many of us are seeing a wide open future where before there was just a wall of silos. I lamented that wall here in *Linux Journal*, way back in September 2011 (http://www.linuxjournal.com/content/way-ranch):

> As entities on the Web, we have devolved. Client-server has become calf-cow. The client—that's you—is the calf, and the Web site is the cow. What you get from the cow is milk and cookies. The milk is what you go to the site for. The cookies are what the site gives to you, mostly for its

> own business purposes, chief among which is tracking you like an animal. There are perhaps a billion or more server-cows now, each with its own "brand" (as marketers and cattle owners like to say).
>
> This is not what the Net's founders had in mind. Nor was it what Tim Berners-Lee meant for his World Wide Web of hypertext documents to become. But it's what we've got, and it's getting worse.

So I want to share what I'm thinking about this whole new thing (which has no one label), in faith that we might bring a Linux-ish sensibility to it.

I am also encouraged that the Linux Foundation is already ahead of the curve with the Hyperledger Project (https://www.hyperledger.org): "an open source collaborative effort created to advance cross-industry blockchain technologies". Those industries already include "leaders in finance, banking, Internet of Things, supply chains, manufacturing and Technology".

The aspirations for new currencies, tokens, distributed ledgers and programming environments in this emerging mega-space are also in some ways similar to those of Linux, early on. Remember Linus' talk about "world domination" (http://catb.org/esr/writings/world-domination/world-domination-201.html) two decades before it came true? It's like that, without the Linus.

Both the internet and Linux were easy calls in the early 1990s, even if relatively few people called them. On the network side, closed "online services", such as AOL and Compuserve, were their own best argument for a network of networks that supported everybody and favored nobody. So did the closed, isolated and doomed networks inside every large enterprise. On the operating system side, BSD was already proving itself as an open alternative to countless warring and closed UNIXes (and was busy forking into three different branches, helping open the way for Linux).

Now the one clear thing is that the internet's original promise of supporting everybody and favoring nobody is still under-fulfilled, meaning the opportunities are still vast, regardless of how much of life on the net is lived inside the feudal castles of what in Europe they call GAFA: Google, Amazon, Facebook and Apple.

In his blog post about Protocol Labs in May 2017 (https://www.usv.com/blog/protocol-labs), Brad Burnham wrote "all of us at Union Square Ventures (https://www.usv.com) believe in the decentralized, emergent, permissionless innovation that was so central to the vitality of the early internet." He continued:

> The key to mitigating the market power of the web giants is open protocols further up the stack (http://www.usv.com/blog/fat-protocols). If an open public communications network (the Internet) unlocked the distribution bottlenecks that characterized the media industry, an open public data layer is the key to unleashing another wave of innovation. It is the mission of Protocol Labs to coordinate the efforts of a large and passionate community of open source contributors to create these protocols.
>
> It is an audacious mission. As you move higher in the stack the complexity of the protocols is exponentially greater. Luckily, they are not starting from scratch. Juan Benet, the founder of Protocol Labs, is the creator of IPFS (the Interplanetary File System: https://ipfs.io), an increasingly popular protocol that allows content on the web to be addressed directly instead of by reference to a file located on a specific server. This subtle but profound change means that a provably unique piece of content is no longer tied to a specific server but can exist anywhere there is a little surplus storage capacity on the web. Protocol Labs and everyone else working on open protocols today has another advantage that was not available to the creators of the original Internet protocols. They have blockchains.
>
> Blockchain based crypto tokens have been described as the native business model of open source software (http://continuations.com/post/148098927445/crypto-tokens-and-the-coming-age-of-protocol). They have the promise of being able to fund the critical shared infrastructure of the information economy in a way that equity can not. Protocols are more valuable when they are open and shared broadly. But equity is most valuable if a company can extract monopoly profits from a resource they exclusively control. When a protocol incorporates an incentive in the form of a crypto token, it can resolve this inherent contradiction.

To start imagining where this goes, it may help to revisit Marshall McLuhan's framework for understanding the effects of a new technology in the world, which I wrote about in my May 2017 EOF (http://www.linuxjournal.com/content/will-anything-make-linux-obsolete). Every new medium (read: technology) has four sets of effects, he said, which can be best discovered in answers to four questions:

- What does the medium enhance?
- What does the medium make obsolete?
- What does the medium retrieve that had been obsolesced earlier?
- What does the medium reverse, or flip into, when pushed to extremes?

He also provided a graphical frame for the answers (Figure 2).

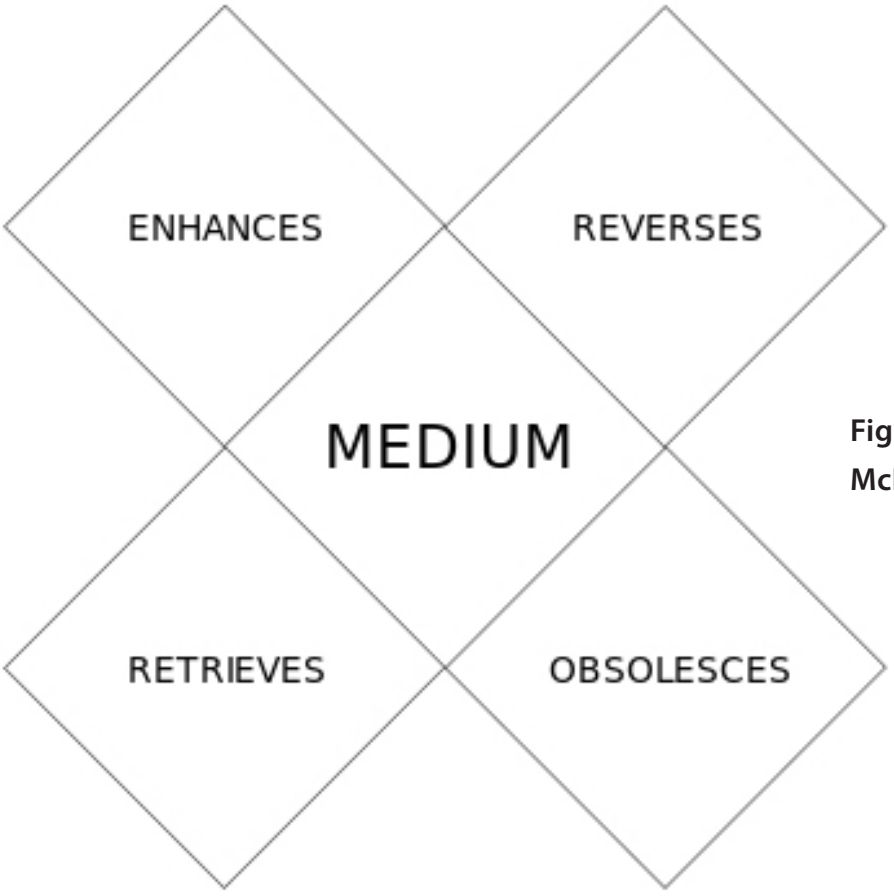


Figure 2. Marshall McLuhan's Framework

So, let's drop *cryptocurrencies* in the middle of that. My first *Hmm* says they—

- Enhance exchange.
- Retrieve the bazaar.
- Obsolesce fiat currency.
- Reverse into a Babel of mutually unintelligible currencies.

While *distributed ledgers*—

- Enhance peer-to-peer.
- Retrieve individual agency.
- Obsolesce platform dominance.
- Reverse into one-to-one.

Remember that this is a heuristic exercise, posed as questions to encourage many different answers. We need to ask these kinds of questions and keep revising our answers until the whole thing gets real.

I am sure the one real thing we already know is that protocols are causes and platforms are effects. Since platforms tend to be the most obvious effects, they also distract us from the boundless other things protocols cause.

Let's look at IPFS, for example. The Why page (https://ipfs.io/#why) at the IPFS site explains how it does for the web what HTTP cannot (or has not):

- "HTTP is inefficient and inexpensive...with video delivery, a P2P approach could save 60% in bandwidth costs" (http://math.oregonstate.edu/~kovchegy/web/papers/p2p-vdn.pdf).
- "Humanity's history is deleted daily...IPFS provides historic versioning (like git) and makes it simple to set up resilient networks for mirroring of data."

- "The web's centralization limits opportunity...IPFS remains true to the original vision of the open and flat web, but delivers the technology which makes that vision a reality."

- "Our apps are addicted to the backbone...IPFS powers the creation of diversely resilient networks which enable persistent availability with or without Internet backbone connectivity."

And here's the list for How:

Each file and all of the blocks within it are given a unique fingerprint called a cryptographic hash. IPFS removes duplications across the network and tracks version history for every file. Each network node stores only content it is interested in and some indexing information that helps figure out who is storing what. When looking up files, you're asking the network to find nodes storing the content behind a unique hash. Every file can be found by human-readable names using a decentralized naming system called IPNS. And, of course, there's a white paper, by @JuanBenet (https://github.com/ipfs/papers/raw/master/ipfs-cap2pfs/ipfs-p2p-file-system.pdf).

It's time to re-distribute the Web, and countless other things built in the stack above the internet's original and still transcendent protocols. IPFS is one way. There are many others.

Let's make them happen.■

Send comments or feedback via http://www.linuxjournal.com/contact or to ljeditor@linuxjournal.com.

RETURN TO CONTENTS

ADVERTISER INDEX

Thank you as always for supporting our advertisers by buying their products!

peer1 hosting

Where every interaction matters.

break down
your innovation barriers

power your business to its full potential

When you're presented with new opportunities, you want to focus on turning them into successes, not whether your IT solution can support them.

Peer 1 Hosting powers your business with our wholly owned FastFiber Network™, global footprint, and offers professionally managed public and private cloud solutions that are secure, scalable, and customized for your business.

Unsurpassed performance and reliability help build your business foundation to be rock-solid, ready for high growth, and deliver the fast user experience your customers expect.

Want more on cloud?
Call: 844.855.6655 | go.peer1.com/linux | Vew Cloud Webinar:

Public and Private Cloud | Managed Hosting | Dedicated Hosting | Colocation