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

PHP FOR THE NON-DEVELOPER

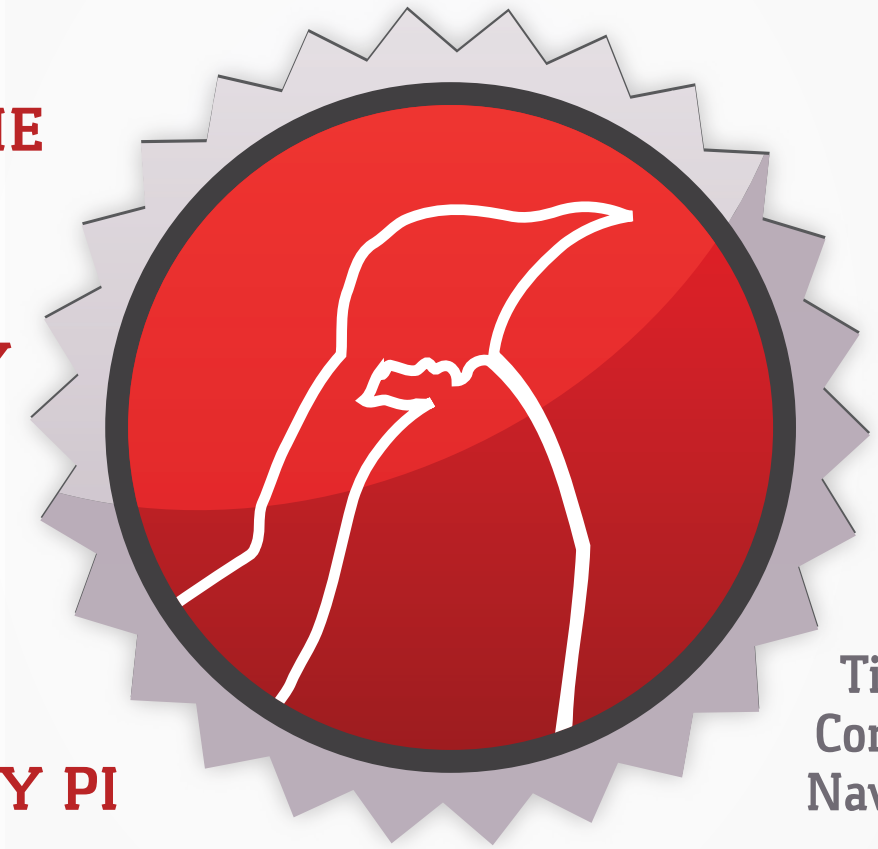
INTRO TO MULTITENANT APPLICATIONS

DECEMBER 2014 | ISSUE 248 | www.linuxjournal.com

READERS' CHOICE AWARDS 2014

SEE HOW THE GNOME DESKTOP FARES IN A USABILITY STUDY

DISCOVER ROGUE WAPs WITH A RASPBERRY PI



\+ Time-Saving Command-Line Navigation Tips

A LOOK AT Power Shell History

WATCH: ISSUE OVERVIEW

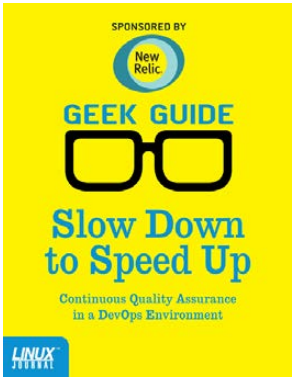
NEW!

Linux Journal eBook Series

GEEK GUIDES

FREE Download NOW!

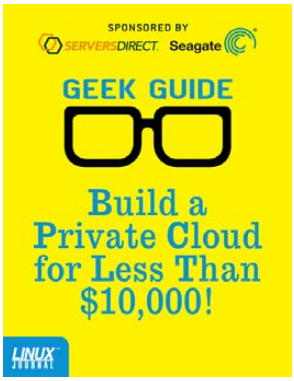
Slow Down to Speed Up:
Continuous Quality Assurance in a DevOps Environment



By Bill Childers

DevOps is one of the newest and largest movements in Information Technology in the past few years. The name DevOps is a portmanteau of "Development" and "Operations" and is meant to denote a fusion of these two functions in a company. Whether or not your business actually does combine the two functions, the lessons and tools learned from the DevOps movement and attitude can be applied throughout the entire Information Technology space. This eBook focuses on one of the key attributes of the DevOps movement: Quality Assurance. At any point, you should be able to release your product, code or configuration—so long as you continue keeping your deliverables in a deployable state. This is done by "slowing down" to include a Quality Assurance step at each point in your workflow. The sooner you catch an error or trouble condition and fix it, the faster you can get back on track. This will lower the amount of rework required and keep your team's momentum going in a forward direction, enabling your group to move on to new projects and challenges.

Build a Private Cloud for Less Than $10,000!



By Mike Diehl

This eBook presents a compelling argument as to why you should consider re-architecting your enterprise toward a private cloud. It outlines some of the design considerations that you need to be aware of before implementing your own private cloud, and it describes using the DevCloud installer in order to install OpenStack on an Ubuntu 14 server. Finally, this eBook will familiarize you with the features and day-to-day operations of an OpenStack-based private cloud architecture, all for less than $10K!

DOWNLOAD NOW AT: http://linuxjournal.com/geekguides

SILICON MECHANICS

SERVERS • STORAGE • HPC

Are you tired of dealing with proprietary storage?

zStax®
ZFS Unified Storage
Powered by NexentaStor

zStax StorCore ZFS Unified Storage from Silicon Mechanics is truly software-defined storage.

From modest data storage needs to a multi-tiered production storage environment, **zStax StorCore** ZFS unified storage appliances have the right mix of performance, capacity, and reliability to fit your needs.

zStax StorCore 64

The **zStax StorCore 64** utilizes the latest in dual-processor Intel® Xeon® platforms and fast SAS SSDs for caching. The zStax StorCore 64 platform is perfect for:

- small-medium office file servers
- streaming video hosts
- small data archives

zStax StorCore 104

The **zStax StorCore 104** is the flagship of the zStax product line. With its highly available configurations and scalable architecture, the zStax StorCore 104 platform is ideal for:

- backend storage for virtualized environments
- mission critical database applications
- always available active archives

Talk with an expert today: 866-352-1173 - http://www.siliconmechanics.com/zstax

CONTENTS

DECEMBER 2014
ISSUE 248

FEATURE

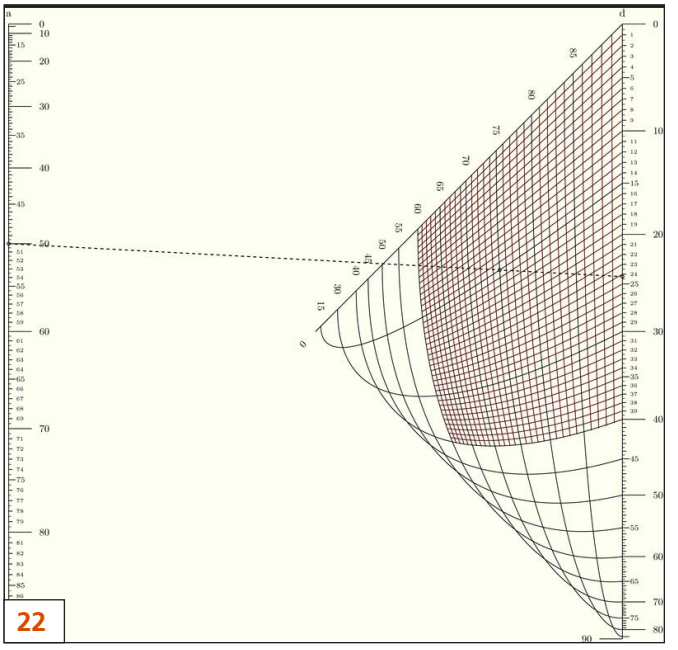
ON THE COVER

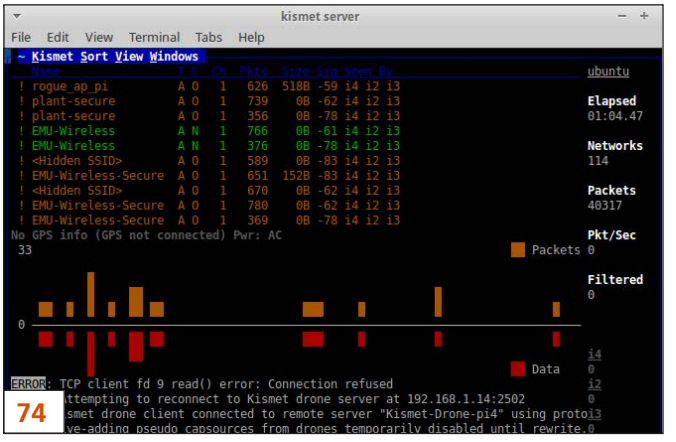
- PHP for the Non-Developer, p. 46
- Intro to Multitenant Applications, p. 30
- Readers' Choice Awards 2014, p. 58
- See How the GNOME Desktop Fares in a Usability Study, p. 92
- Discover Rogue Access WAPs with a Raspberry Pi, p. 74
- Time-Saving Command-Line Navigation Tips, p. 42
- A Look at Power Shell History, p. 38

Interested in joining our Reader Advisory Panel for 2015? Please send a brief e-mail explaining why you'd be a good fit to ljeditor@linuxjournal.com.

INDEPTH

COLUMNS





IN EVERY ISSUE

LINUX JOURNAL (ISSN 1075-3583) is published monthly by Belltown Media, Inc., 2121 Sage Road, Ste. 395, Houston, TX 77056 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

SHAWN POWERS

The Best of the Best

I love the Readers' Choice issue. I jokingly say it's because all the work is done by the community, but honestly, it's because I love hearing the feedback from everyone. Year after year, I inevitably learn about a new technology or application, and I'm usually surprised by at least one of the voting results. It's also an "unthemed" issue, which means our articles can be from any discipline in the Linux world. Welcome to the Readers' Choice issue of *Linux Journal*.

Reuven M. Lerner starts us out this month with the modern-day equivalent of virtual name-based Web hosting. Multitenant programming allows a single server and/or application to serve multiple clients while keeping the cross-pollination of data from taking place. SaaS isn't a new concept, but Reuven shows how to peel back the cloud layer and see what goes on behind the scenes. If you're a Web developer, you won't want to miss it.

I've written several articles through the years on SSH, which I like to think of as the Swiss Army Knife of command-line tools. Dave Taylor talks about an equally powerful tool this month, namely the `find` command. It seems like a simple concept, but with its various options, `find` can be a powerful addition to any Bash script.

Kyle Rankin talks about something I've never been able to get the hang of doing. He discusses the virtue of using the hjkl keys as opposed to arrow keys while editing text files. Every once in a while, you'll run across a computer that supports only the old-school method for moving the cursor around, but for Kyle, that old-school method is his preferred mode of operation. I have to admit, I'm a little jealous. I can play a first-person shooter with the "wasd" keys, but for some reason, my brain just can't get used to "hjkl". If you have no idea what I'm talking about, be sure to

VIDEO:
Shawn Powers runs through the latest issue.

read Kyle's column.

I boldly go where no Shawn Powers has gone before and touch on development of all things. My entire IT career has been one where I've avoided programming. Recently, I had a need to create a Web-based interface for a user, and I learned PHP. Granted, PHP isn't the most modern or powerful programming language, but if you're like me, it might be the perfect intro to development. I urge anyone who hates programming to read my column. It might surprise you how much fun writing code can be.

Chris Jenks has a cool article in this issue where he teaches us to hunt rogue access points in a network. There obviously are apps for Android that will do basic scanning, but Chris goes a lot further and discusses passively gathering data, looking for even hidden SSID networks. If you have a need to track down rogue access points, but fear sneaky users are hiding their tracks, Chris' article is perfect.

Jim Hall goes searching this month as well, but in his case, he's searching for usability. Jim describes the process for formal usability testing and how it shapes changes in projects like GNOME, which he uses as an example. If you've ever been curious about how and why desktop programs function the way they do, Jim's article will be of incredible interest. In fact, usability is most likely one of the factors that influences the results of our Readers' Choice awards year after year. If proper testing has taken place, the chances for a particular technology to gain popularity are much higher. We stuck with a similar format as last year in this issue, and gave the entire range of votes as opposed to just the top few candidates. If people were told only about the most popular one or two desktop environments, no one would ever hear about Linux itself! There are a few upsets this year over last year, but every category is interesting with a few hidden gems that might not be as popular, but certainly are worth checking out.

This issue is truly an eclectic one, and although there's not a specific focus, that means we get to focus on "awesome stuff", which is always enjoyable. We have tech tips, product announcements, cool programs and enough information to keep an entire herd of nerds busy for weeks. We hope you enjoy the Readers' Choice issue of *Linux Journal*. I know we enjoyed putting it together.■

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 e-mail at shawn@linuxjournal.com. Or, swing by the #linuxjournal IRC channel on Freenode.net.

letters

Leap Years in Bash

Jacques Amar is close (see the October 2014 Letters). The year 2000 was a leap year.

A year is a leap year if it is a multiple of 400 or is a multiple of four and not a multiple of 100.

—fest3er8

"Accessing the I/O Ports of the BeagleBone Black with Python" Article

Regarding Samuel Bucquet's article in the October 2014 issue, it looks like the author is running everything as root. This seems to be fairly common in articles about the Raspberry Pi or BBB where the hardware is accessed, as Linux puts up some security barriers to protect these ports from the ordinary user. There are reasonable ways to work around this limitation, such as exporting the pins in a boot script. Also, WebIOPi looks like a well put together solution for the Raspberry Pi that could be ported to the BBB. *LJ* should not be encouraging running as root, particularly on hardware that frequently ends up as mini-Web servers.

—Mike Beaver

Samuel Bucquet replies: *Thank you, Mr Beaver, for pointing this out.*

The access of the IO ports needs a lot of attention in a multi-user scenario. In the same way, if there is a possibility of multiple tasks accessing the same hardware simultaneously, you need to consider some locking mechanism. The majority of the tasks I described in the article are hardware administration tasks, but the reading and writing on the IO ports can be done as a regular user, like the user a Web server is running with to allow access of IO ports from a Web page.

Serial ports: *In order to use them as a regular user, there is nothing special*

to do, as long as the user is in the dialout group (check with "id").

GPIO: *You will have to initialize, as root, the GPIO that will be used, and position the permissions accordingly (like this for the Debian default user):*

```
# echo 48 > /sys/class/gpio/export
# chgrp -R debian /sys/class/gpio/gpio48/
# find /sys/class/gpio/gpio48/ -type f -exec chmod 664 {} \;
# find /sys/class/gpio/gpio48/ -type d -exec chmod 775 {} \;
```

But beware, you don't want multiple users fidgeting with the same GPIOs at the same time!

Then you can access the gpio with the "debian" user.

Components on the i2c bus: *As with the serial ports, if the user is in the "i2c" group, she can access components on the bus as a regular user. But there are a lot of sensitive components on an i2c bus, so if the integrity of the system is at risk, take*

great care as to who can access what.

Analog reading: *In the same way you did for the GPIO, put the permissions in /sys/bus/iio/devices/iio\:device0/ for the user, and again do not allow simultaneous access to the ADC.*

Handling of the permissions may vary a little, and `udev` *can be put to good use for other devices not automatically recognized, but the essential part is here.*

I might add that the solution I chose to handle for this is a one and only process to access an IO port.

A shared memory server relays readings and writings to the IO handling process. I use redis as a back end for the following reasons:

- *It allows me access to the IO from the network from multiple clients (rarely multiple writing though).*
- *It allows me to dispose of IO readings in a coherent way with transactions.*
- *It allows me to attach a validity period to each item of data.*
- *And it even allows a PUS/SUB mechanism.*

Of course, there is a good Python module too.

Shawn Powers and SSH

Writing letters to *LJ* is becoming habit-forming this month.

Shawn Powers left out the scariest part of SSH tunnels in his article "This Is Why We Can't Have Nice Things: SSH" in the October 2014 issue. It takes some thought and effort to set it up right, root access is needed, and public/private keys are required, but it's relatively easy to set up PPPoS (PPP over SSH) using pon and poff. Then, you configure NAT as needed at each end of the link to obtain Internet access unfettered by a firewall. Finally, redirect internal nodes to use the internal system that hosts that end of the tunnel as their default gateway. That should be enough to make the toughest admin blanch.
—fest3er8

Ha! Indeed, SSH is like the light saber of the tech world. It's super amazing, but it's fairly easy to cut your arm off!—Shawn Powers

Linux Journal in Audio Format?

I am responding to Shawn Powers' article in the August 2014 issue titled "First Health, Now Recreation". Since he mentioned Audible, I would like to

ask if he has tried converting *Linux Journal* into an audio file?

I love Audible, mainly because I can maximize the way I absorb information in a crowded train or while driving. (I just caught up on the last two *LJ* issues on flight when I was traveling to a conference yesterday.)

I have been thinking that if I could listen to *LJ*, that would be very productive! With the fact that most of the time 75% of the book is readable, maybe Dragon Speech (which I don't have) would work?
—Leo Chan

That's an interesting question, Leo. One of my writing tricks is that I have the computer read back the text to me so I can hear any mistakes that my eyes glance over. I'm not sure the robotic voice reading an entire issue would be great, especially when it comes to the code sections, but I wonder if folks with assistive reading devices have luck with Linux Journal*'s text. Any text-to-speech listeners out there?—Shawn Powers*

WRITE *LJ* A LETTER

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

PHOTO OF THE MONTH

Remember, send your Linux-related photos to ljeditor@linuxjournal.com!

LINUX JOURNAL

At Your Service

SUBSCRIPTIONS: *Linux Journal* is available in a variety of digital formats, including PDF, .epub, .mobi and an on-line digital edition, as well as apps for iOS and Android devices. Renewing your subscription, changing your e-mail address for issue delivery, paying your invoice, viewing your account details or other subscription inquiries can be done instantly on-line: **http://www.linuxjournal.com/subs**. E-mail us at subs@linuxjournal.com or reach us via postal mail at *Linux Journal*, PO Box 980985, Houston, TX 77098 USA. Please remember to include your complete name and address when contacting us.

ACCESSING THE DIGITAL ARCHIVE: Your monthly download notifications will have links to the various formats and to the digital archive. To access the digital archive at any time, log in at **http://www.linuxjournal.com/digital**.

LETTERS TO THE EDITOR: We welcome your letters and encourage you to submit them at **http://www.linuxjournal.com/contact** or mail them to *Linux Journal, PO Box 980985,* Houston, TX 77098 USA. Letters may be edited for space and clarity.

WRITING FOR US: We always are looking for contributed articles, tutorials and real-world stories for the magazine. An author's guide, a list of topics and due dates can be found on-line: **http://www.linuxjournal.com/author**.

FREE e-NEWSLETTERS: *Linux Journal* editors publish newsletters on both a weekly and monthly basis. Receive late-breaking news, technical tips and tricks, an inside look at upcoming issues and links to in-depth stories featured on **http://www.linuxjournal.com**. Subscribe for free today: **http://www.linuxjournal.com/enewsletters**.

ADVERTISING: *Linux Journal* is a great resource for readers and advertisers alike. Request a media kit, view our current editorial calendar and advertising due dates, or learn more about other advertising and marketing opportunities by visiting us on-line: **http://ww.linuxjournal.com/advertising**. Contact us directly for further information: ads@linuxjournal.com or +1 713-344-1956 ext. 2.

diff -u
WHAT'S NEW IN KERNEL DEVELOPMENT

Containers are very tricky to implement. Trying to isolate sets of resources from each other completely, so that they resemble a discrete system, and doing it in a secure way, has to be addressed on a feature-by-feature basis, with many caveats and uncertainties. Over time, this makes the core kernel code more secure and robust, but each individual feature may have surprising issues.

The whole **namespace** idea—corralling subsets of system resources like user IDs and group IDs, and performing on-the-fly translations between the resource names within the container and the corresponding names in the outer system—is tough to manage.

Recently, **Marian Marinov** noticed that process counters in the outer system counted processes as being owned by the same user if his or her UIDs (user IDs) were the same inside two separate containers. The same was true for GIDs (group IDs). He didn't like this, because the two containers represented two logically isolated systems, and in that context, the same UIDs could refer to different users entirely. They shouldn't be counted together.

He wanted to patch the kernel to isolate these values, so that the process counters wouldn't get them mixed up with each other. But, **Eric W. Biederman** didn't like Marian's idea of creating namespace-specific data structures within the kernel. He offered some workarounds, but Marian didn't like any of them. He said they were less efficient than his idea, and they would require him to put a lot of effort into redesigning his particular use case, which ran a batch of identical containers, each built out of a single master template.

Ultimately, Eric suggested implementing something akin to Marian's idea, but only for certain filesystems. The problem, he said, was that **XFS** exposed too much of its inner workings to userspace, making it hard to perform the namespace translations correctly. If they limited support only to "normal" filesystems, Eric said, it would be much easier to give Marian what he wanted.

But, **James Bottomley** pointed out that Linux distributions wouldn't sacrifice XFS for anything. It had already been tried with the **USER_NS** feature. Distributions wouldn't accept USER_NS unless it supported XFS. James argued that the same would be true here.

Eric replied that the two cases were different. His solution would not preclude using XFS in a Linux distribution; it would only preclude using a particular use case that didn't currently exist anyway. And, Eric also argued that XFS already had serious issues that made it less container-friendly than other filesystems. It was hard to migrate an XFS filesystem to a system with different endianness and word size. This meant one of the common container uses—migrating processes and containers between machines—already was partially off the table for XFS, at least for the moment. That being the case, Eric said, it didn't make sense to go to great lengths to support it in a feature it couldn't use until so many other XFS characteristics had been fixed.

The debate undoubtedly will continue. Ultimately, the question involves identifying where to draw a line between seemingly integrated features of the kernel. What parts of the system can be containerized safely? What parts have to wait until other issues are addressed? In some cases, the end result will be much cleaner kernel code; in other cases, in the short term, much messier.

Some features get so big and complicated that they can't be changed easily anymore. In particular, it becomes harder to fix design flaws, because each fix has to account for all the existing special wonkiness. The **printk()** function is one example. Its code apparently has become such a nightmare that kernel developers must choose worse solutions to their problems, just in order to avoid a redesign process that is made so difficult by printk()'s current insane implementation.

Recently, **Petr Mladek** submitted some code to allow calling printk() from within an **NMI** (non-maskable interrupt). This is useful when a system is in the midst of crashing and needs to output logging data to help the user identify what went wrong. The problem was that printk() needed to take a lock that might be held by another process. That's a big no-no in NMIs, because the whole point of NMIs is that they never can be interrupted by other processes. The

printk() would loop forever, waiting for a process to release a lock, when that process would never get the CPU cycles it needed to release that lock. Presto, deadlock.

Petr's code solved this by taking the lock only if available and failing over to an alternate solution if necessary. Overall, Petr's code improved the situation, because users actually were seeing lockups that could be better-diagnosed with printk()s in NMIs. Specifically, **Jiri Kosina** said, "we've actually seen the lockups triggered by the RCU stall detector trying to dump stacks on all CPUs, and hard-locking the machine up while doing so."

But, as **Frédéric Weisbecker** put it, the printk() code base was an "ancient design" with "fundamental flaws". Its poor design forced Petr's patch to be 1,000 lines long when such a fix ordinarily might be much smaller (**Linus Torvalds** later estimated 15 lines as a good size for Petr's features). Frédéric suggested, "shouldn't we rather redesign it to use a lockless ring buffer like ftrace or perf ones?"

Jiri agreed that the printk() code base was "a stinking pile of you-know-what", and that a redesign would be better than Petr's stop-gap patch. But in fact, he said, the correct design was not yet known, but regardless certainly would take a long time to implement and would delay Petr's important fix that addressed real-world crashes. And as Frédéric added, there also was the danger that "if we push back this bugfix, nobody will actually do that desired rewrite."

At some point, Frédéric asked for Linus' opinion, and Linus essentially torpedoed Petr's whole approach. He said:

> Printing from NMI context isn't really supposed to work, and we all *know* it's not supposed to work.
>
> I'd much rather disallow it, and if there is one or two places that really want to print a warning and know that they are in NMI context, have a special workaround just for them, with something that does *not* try to make printk in general work any better.
>
> Dammit, NMI context is special. I absolutely refuse to buy into the broken concept that we should make more stuff work in NMI context. Hell no, we should *not* try to make more crap work in NMI. NMI people should be careful.
>
> Make a trivial "printk_nmi()" wrapper that tries to do a trylock on logbuf_lock, and *maybe* the

existing sequence of:

```
if (console_trylock_for_printk())
        console_unlock();
```

then works for actually triggering the printout. But the wrapper should be 15 lines of code for "if possible, try to print things", and *not* a thousand lines of changes.

Which, Petr said, was exactly what his patch did, but he just needed 1,000 lines of code instead of 15 because of how broken printk() was already. And Jiri said, "I find it rather outrageous that fixing *real bugs* (leading to hangs) becomes impossible due to printk() being too complex. It's very unfortunate that the same level of pushback didn't happen when new features (that actually *made* it so complicated) have been pushed; that would be much more valuable and appropriate."

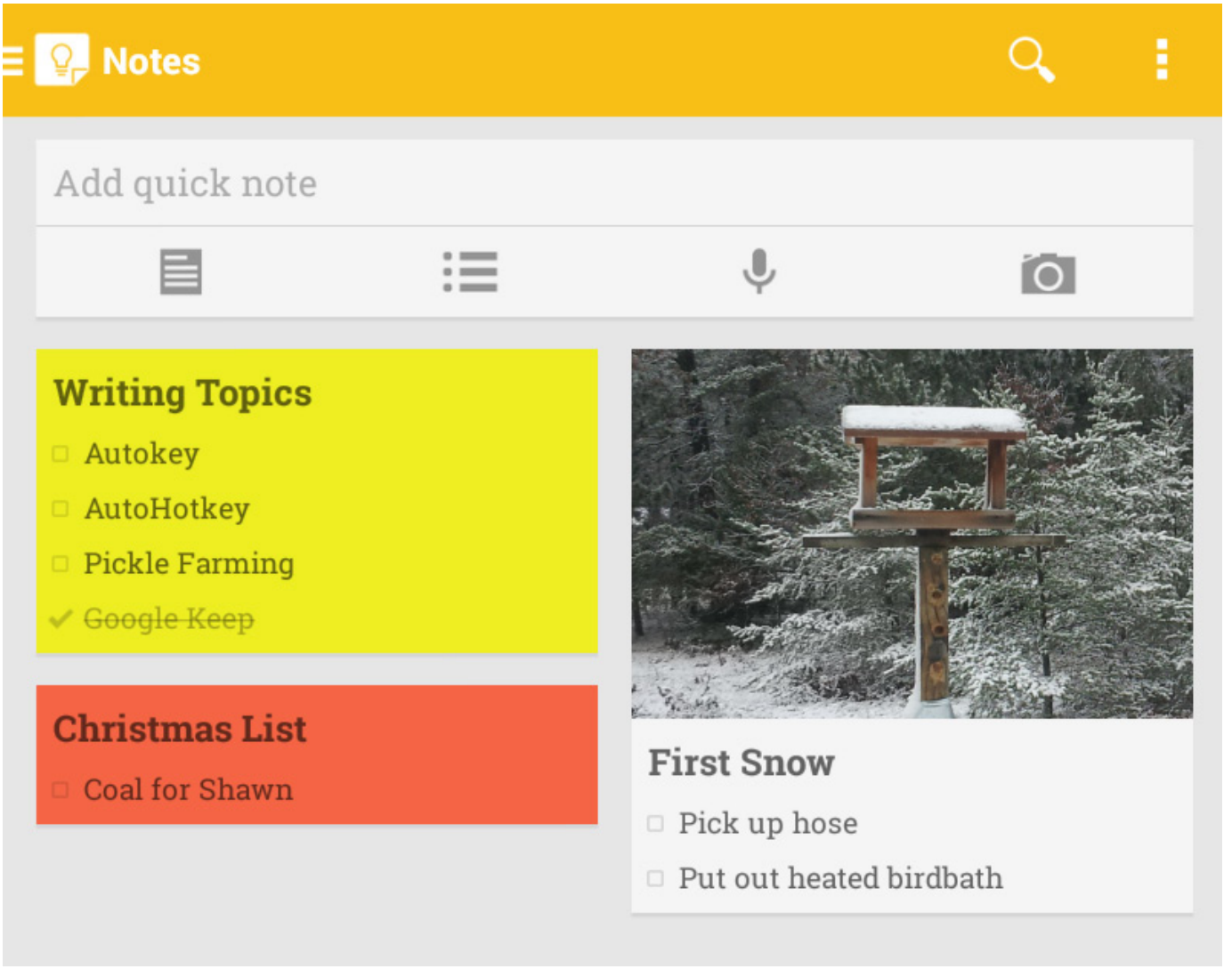
At this point, **Paul McKenney** offered a compromise. Since Petr's patch was inspired by the **RCU** (read-copy-update) stall detector using NMIs to dump the stack, and thus needing printk(), Paul could rewrite the RCU code to avoid using NMIs for the stack dump. This way, regular printk() would work, without requiring Petr's patch.

The problem with this was that RCU wouldn't do *quite* as good a job of dumping the stack data. As Jiri put it, "this is prone to producing not really consistent stacktraces though, right? As the target task is still running at the time the stack is being walked, it might produce stacktraces that are potentially nonsensical."

But, Linus was insistent. He said, "We should stop using nmi as if it was something 'normal'. It isn't. Code running in nmi context should be special, and should be very very aware that it is special. That goes way beyond 'don't use printk'. We seem to have gone way way too far in using nmi context. So we should get *rid* of code in nmi context rather than then complain about printk being buggy."

So, Paul's solution, even being known to provide worse stack dumps than Petr's, would be adopted, simply because it could avoid making further changes to printk(). Jiri said, "I feel bad about the fact that we are now hostages of our printk() implementation, which doesn't allow for any fixes/improvements. Having the possibility to printk() from NMI would be nice and more robust... otherwise, we'll be getting people trying to do it in the future over and over again, even if we now get rid of it at once."—**ZACK BROWN**

Android Candy: Google Keep



I love Evernote. I pay for a premium membership, and to be honest, I don't think I even use the premium features. I just love Evernote so much, I want to support the company. But in the spirit of fair comparison, I forced myself to try Google Keep.

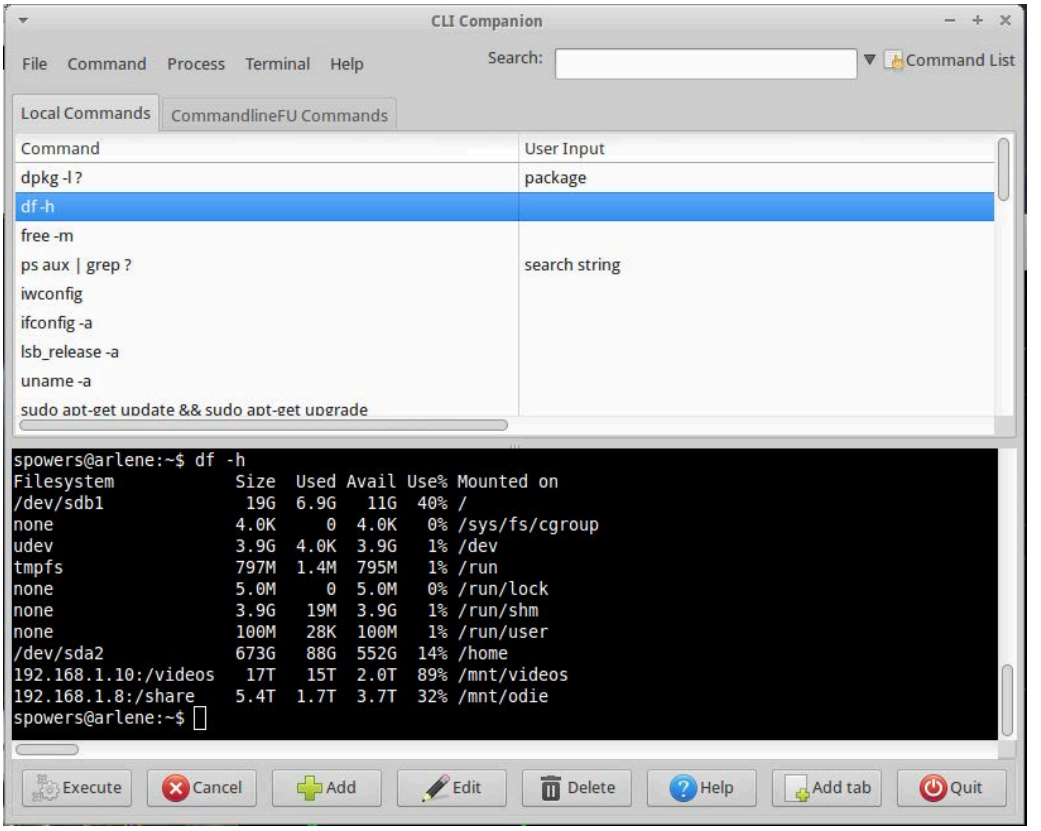
It's pretty neat. Honestly though, even though Google Keep has matured quite a bit, I don't see it as a competitor with Evernote. I thought that was what it was going to be, but to me it seems

more like a really awesome sticky-note program that syncs seamlessly between devices. The Web interface (**http://keep.google.com**) mirrors the Android app almost exactly, and it syncs in what seems like real time. You can grab notes to rearrange them, and yes, you can search notes using Google's powerful search engine.

Perhaps it's the lack of integration with other apps, or perhaps it's just that Google Keep is so new compared to Evernote, but I couldn't use it on a daily basis for more than quick notes here and there. I'd be curious to know if there are any Google Keep fanatics out there who can't imagine life without it. For me, Evernote combined with Nixnote is still the ultimate tool for keeping track of, well, everything! Download Google Keep from the Google Play store and give it a try. It really is a neat app, and if you're not an Evernote addict, it might be perfect!

—SHAWN POWERS

A GUI for Your CLI?



For new Linux users, the command line is arguably the most intimidating thing. For crusty veterans like me, green text on a black background is as cozy as fuzzy slippers by a fireplace, but I still see CLI Companion as a pretty cool application.

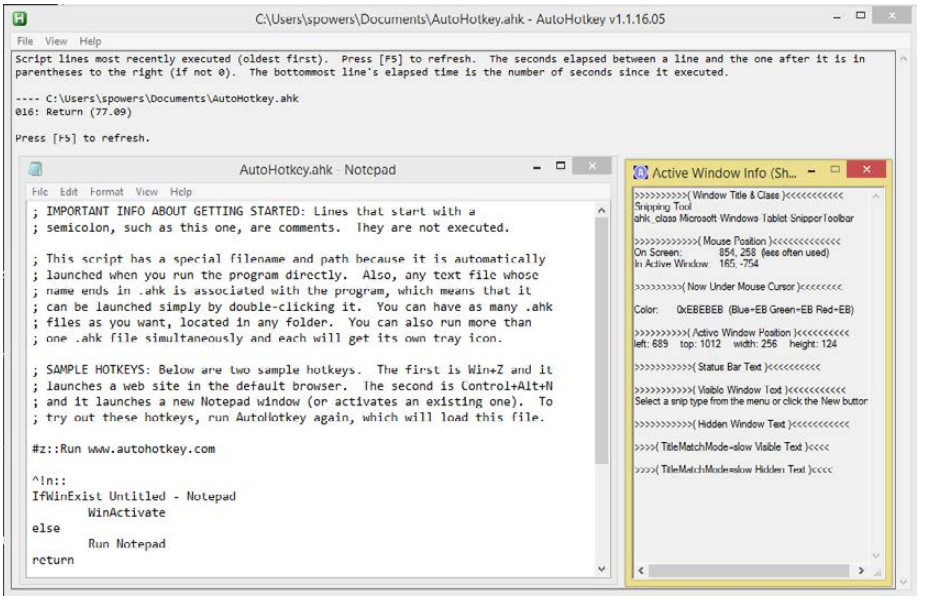
The concept is pretty easy. It's a GUI environment that allows you to double-click your way into entering CLI commands. You can create your own commands (sort of like bookmarks for the command line), or you can search from within the large database of common applications. Oh, and if the command requires command-line arguments? The application pops up a window prompting you to enter them in.

If you're a dyed-in-the-wool Linux professional, CLI Companion might seem like a silly thing to install. If the command line is intimidating, on the other hand, it might be the perfect tool to help you gain mastery. Heck, as I scrolled through the database of applications, I learned a few command-line tools I didn't know existed!

If a GUI CLI is an oxymoron you'd like to check out, surf over to **https://launchpad.net/clicompanion** and check it out.

—SHAWN POWERS

Non-Linux FOSS: Don't Type All Those Words!



We mention Autokey later in this issue as a great tool for text replacement in real time on Linux. Thankfully, there's an option for Windows users that actually is even more powerful than Autokey! AutoHotkey is a similarly named application that runs strictly under Windows. It's still FOSS, but there's unfortunately no version for Linux.

The premise for AutoHotkey is the same as Autokey for Linux. Type a quick short bit of text, and it will expand that shortcut into the predefined text you tell it to use. I find this useful while programming, as creating those curly braces in pairs is very useful. The program is a bit of a bear to configure, because there's no GUI to configure keys. In order to configure the program, you write a text-based script that defines your shortcuts.

AutoHotkey (AHK it's sometimes called) even allows you to pre-compile your shortcuts into an executable so you don't need to re-program them when you move to a new computer. Grab your .exe file, and run it when you visit your folks for the holidays. (But don't make a shortcut that automatically misspells your sibling's name when your parents type it...or if you do, don't blame me!)

AutoHotkey is free, and it's available at **http://ahkscript.org**. There's a nice quick-start tutorial as well to help you get started, because like I mentioned, it's a little rough at first. If you're stuck typing a lot of text on a Windows machine, check out AutoHotkey today!

—SHAWN POWERS

Computing without a Computer

I've covered a lot of various pieces of software that are designed to help you do scientific calculations of one type or another, but I have neglected a whole class of computational tools that is rarely used anymore. Before there was the electronic computer, computations had to be made by hand, so they were error-prone. To try to minimize these human errors, shortcuts and aids of one form or another were developed.

A common computational problem is to solve equations of some number of variables. The tool that was developed for this class of problem is the nomograph, or nomogram. A nomograph uses a graphical representation of an equation to make solving the equation as simple as setting down a straightedge and reading off the result. Once a nomograph is constructed, it is one of the fastest ways to solve an equation by hand.

In this article, I explore some common nomographs that many of you likely will have seen, and I take a look at a Python package, PyNomo (**http://www.pynomo.org**), that you can use to create your own. I also walk through creating some new nomographs, which hopefully will inspire you to try creating some too.

First, let me explain what a nomograph actually is. Electrical engineers already should have seen and used one example, the Smith chart. This chart provides a very quick way to solve problems

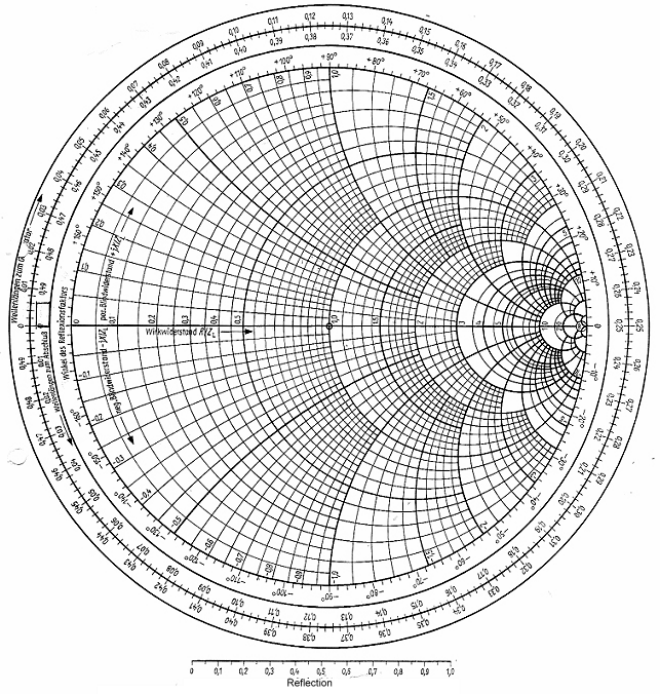


Figure 1. With a Smith chart, you can work on problems around transmission lines and circuit matching.

involved with transmission lines and matching circuits. Solving these types of problems by hand was a very tedious task that wasted quite a lot of time, so the introduction of the Smith chart increased productivity immensely.

A Smith chart is scaled in normalized impedance, or normalized admittance, or both. The scaling around the outside is in wavelengths and degrees. The wavelength scale measures the distance along the transmission line between the generator and the load. The degree scale measures the angle of the voltage reflection coefficient at that point. Since impedance and admittance change as frequency changes, you can solve problems only for one frequency at a time. The result calculated at one frequency is a single point on the Smith chart. For wider bandwidth problems, you just need to solve for a number of frequencies to get the behaviour over the full range. But, because this isn't meant to be a lesson in electrical engineering, I will leave it as an exercise for the reader to see just how many other problems can be solved with a Smith chart.

Another example, which should be recognizable to any parent, is the height/weight charts used by doctors. These charts allow a doctor to take the weight and height of a child and see where he or she fits on a nonlinear scale that compares one child to the available statistics of a population very quickly. This is much easier than plugging those values into an equation and trying to calculate it manually.

But, what can you do if you want to use a totally new type of nomograph? Enter the Python module PyNomo. The easiest way to install PyNomo is to use pip. You would type:

```
pip install PyNomo
```

You may need to preface this command with `sudo` if you want it installed as a system module. To get started, you need to import everything from the nomographer section with:

```
from pynomo.nomographer import *
```

This section contains the main Nomographer class that actually generates the nomograph you want to create. There are ten types of nomographs that you can create with PyNomo:

- Type 1: three parallel lines

- Type 2: N or Z
- Type 3: N parallel lines
- Type 4: proportion
- Type 5: contour
- Type 6: ladder
- Type 7: angle
- Type 8: single
- Type 9: general determinant
- Type 10: one curved line

Each of these also is described by a mathematical relationship between the various elements. For example, a type 1 nomograph is described by the relationship:

```
F1(u1) + F2(u2) + F3(u3) = 0
```

Each element of a given nomograph must be of one type or another. But, they can be mixed together as separate elements of a complete nomograph. A simple example, borrowed from the PyNomo examples on the main Web site, is a temperature converter for converting between Celsius and Fahrenheit degrees. It is generated out of two type 8 blocks. Each block is defined by a parameter object, where you can set maximum and minimum values, titles and tick levels, as well as several other options. A block for a scale going from –40 to 90 degrees Fahrenheit would look like this:

```
F_para={'tag':'A',
        'u_min':'-40.0,
        'u_max':'90.0,
        'function':lambda u:celcius(u),
        'title':r'$^\circ$ F',
        'tick_levels':4,
        'tick_text_levels':3,
        'align_func':celcius,
        'title_x_shift':0.5
        }
```

You will need a similar parameter list for the Celsius scale. Once you have that, you need to create block definitions for each of the scales, which looks like this:

```
C_block={'block_type':'type_8',
         'f_params':C_para }
```

The last step is to define a parameter list for the main Nomographer class. For the temperature converter, you can use something like the following:

```
main_params={'filename':'temp_converter.pdf',
             'paper_height':20.0,
             'paper_width':2.0,
             'block_params':[C_block,F_block],
             'transformations':[('scale paper')]
             }
```

Now you can create the nomograph you are working on with the Python command:

```
Nomographer(main_params)
```

A more complicated example is a nomograph to help with the calculations involved in celestial navigation. To handle such a complex problem, you need to use a type 9 nomograph. This type is a completely general form. You need to define a determinant form to describe all of the various interactions. If the constituents are functions of one variable, they will create a regular scale. If they are of two variables, they will create a grid section. For example, one of the single scales in this example would look like this:

```
'g':lambda u:-cos(u*pi/180.0)
```

Whereas the grid is defined by:

```
'g_grid':lambda u,v:-sin(u*pi/180.0)*sin(v*pi/180.0)
```

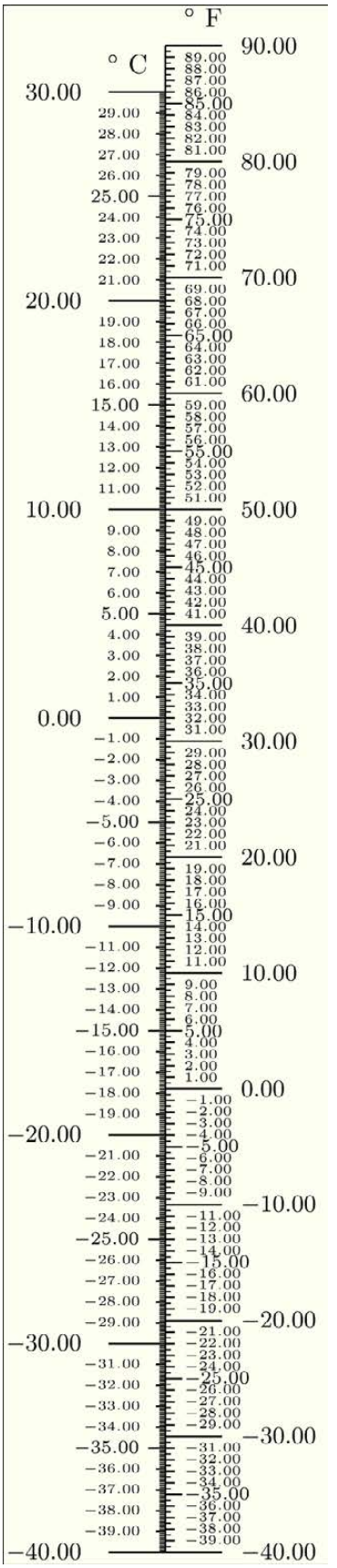


Figure 2. A simple nomograph is a Celsius-Fahrenheit temperature conversion scale.

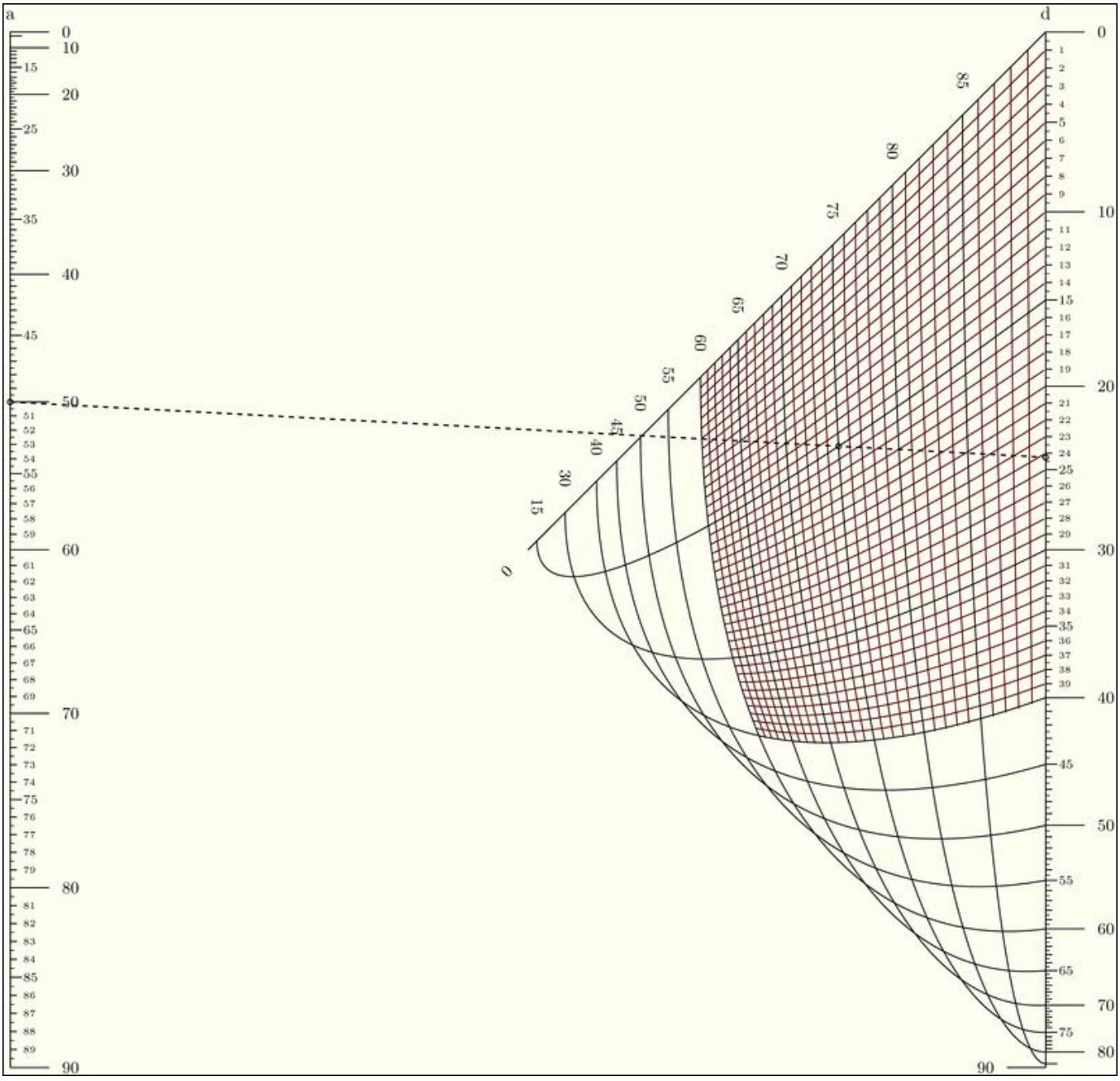


Figure 3. You even can do something as complicated as celestial navigation with a nomograph.

Once this nomograph is constructed, you can use it to compute the altitude azimuth.

PyNomo goes through several steps in generating the nomograph. The last step is to apply any transformations to the various parts. Transformations to individual

components can be applied only to type 9 nomographs. If you do apply transformations to individual components, you need to make sure that relative scalings between the various parts are still correct. For other nomograph types, transformations can be applied only to the entire nomograph. There aren't a large number of transformations available yet, but there are enough to handle most customizations that you may want to make. The transformations available are:

- scale paper: scale the nomograph to the size defined by paper_height and paper_width.
- rotate: rotates the nomograph through the given number of degrees.
- polygon: applies a twisting transformation to the tops and bottoms of the various scales.
- optimize: tries to optimize numerically the sum squared lengths of the axes with respect to paper area.

With these transformations, you should be able to get the look you want for your nomograph.

Now that you know about nomographs, and even more important, how to make them, you really have no excuse to avoid your trip to that isolated South Pacific island. Go ahead and play with PyNomo and see what other kinds of nomographs you can make and use.

—JOEY BERNARD

They Said It

You may be disappointed if you fail, but you are doomed if you don't try.
—Beverly Sills

One person with a belief is equal to a force of 99 who have only interests.
—John Stuart Mill

The freethinking of one age is the common sense of the next.
—Matthew Arnold

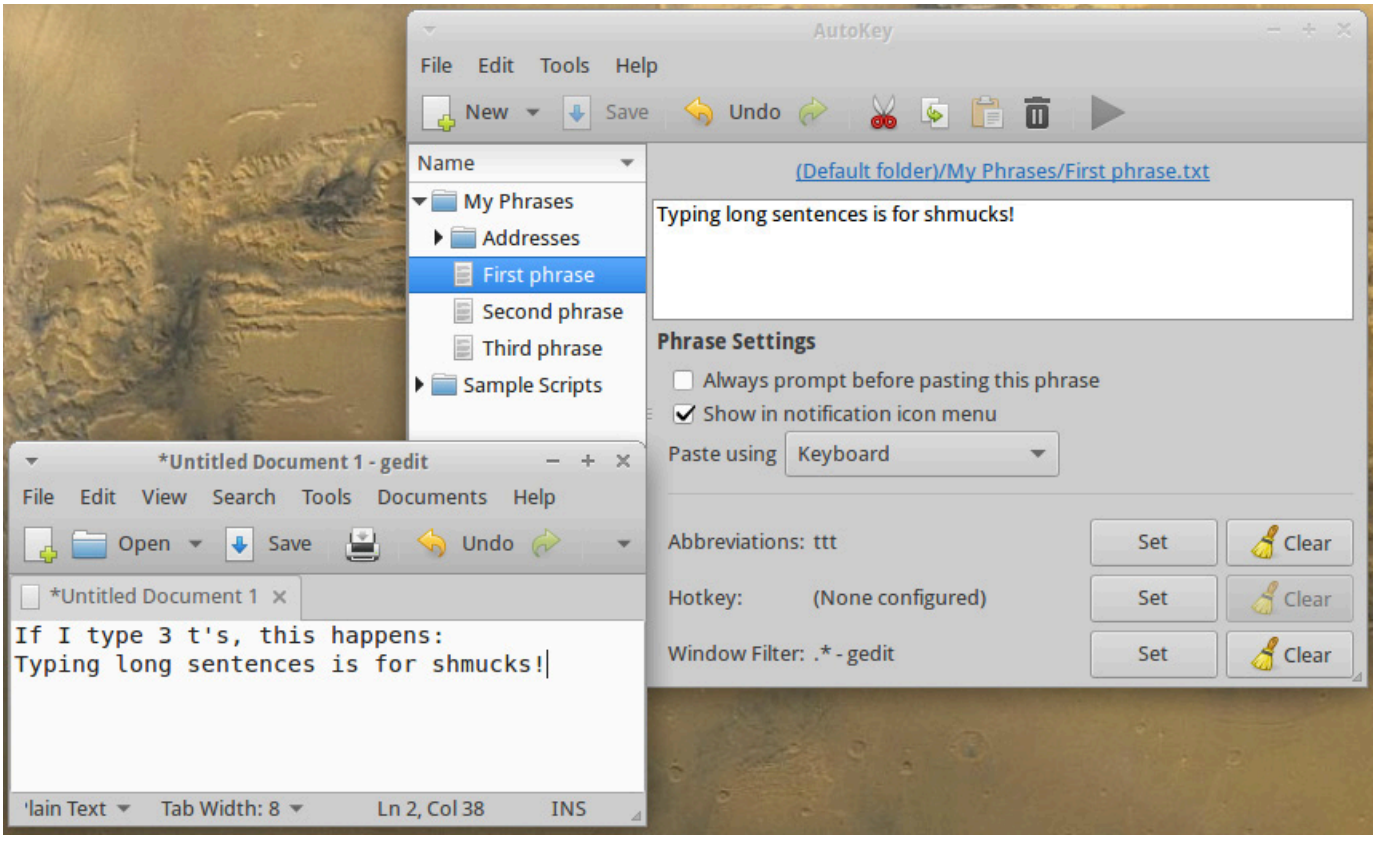
Real success is finding your lifework in the work that you love.
—David McCullough

The secret of happiness is to make others believe they are the cause of it.
—Al Batt

Autokey: Shorthand for Typists

For years I avoided installing keyboard shortcut tools on my computers. I thought dog-gonnit, if something needed to be typed out, I'd type every letter myself. Recently I capitulated, however, and I must say, going back seems unlikely. If you've never tried a text-replacement app, I highly recommend doing so. The time it saves is incredible, and after I abandoned my grouchy old ways, I've grown to love it.

Unfortunately, there aren't too many options in Linux for really good text-replacement apps. My personal favorite is Autokey. It can be a challenge to set up,



but that's mainly because it's so powerful. If you want text replacement to work only in a particular app (think programming code shortcuts), it can be set to work only with those apps. If you want to have a special hotkey required before text replacement works, that's an option too. In the screenshot here, you can see I make a simple auto-replacement shortcut so that every time I type "ttt", it replaces it with a sentence. It works only in gedit, because that's the constraint I set in the settings.

Even though it has a very complicated interface, the Autokey program is a great program, and one we hope continues to get updated once it no longer works. It's currently available for Ubuntu 12.04 and below, but it will install on recent versions without a problem. Autokey takes this month's Editors' Choice award, because once you start using it, you won't be able to imagine life without it! Get your copy today at **https://code.google.com/p/autokey**.

—SHAWN POWERS

REUVEN M. LERNER

Multitenant Sites

One server and one program can service many sites. Here's an introduction to "multitenant" applications.

For some time now, there has been tremendous growth in the world of Web applications. It's quite amazing to see what you can do just via a Web browser—not only can you buy just about anything, but also a growing number of sites offer "software as a service", often abbreviated as SaaS. The idea is that in exchange for a monthly service fee, you get access to a service. Many thousands of such services exist that take care of anything from Git repositories (for example, GitHub and BitBucket), e-mail services (for example, AWeber and MailChimp), invoicing systems, time-tracking systems, calendar systems, e-commerce systems, e-learning systems—you name it.

As Web developers, you can create your own SaaS applications. That's right—with little more than a Linux box, a database, a programming language and a Web framework, you're positioned to create a new SaaS application. With a good idea, some hard work and good marketing, you'll be on your way to having a successful business.

There are numerous models for how SaaS can work. Sometimes, you have a user name on a system, and you're simply interacting with your view of the world. But sometimes, an SaaS app gives you what appears to be an entirely new domain. So if I get an account on SuperDuperSaas.com, everything I do will be under lerner.SuperDuperSaas.com.

Programs allowing for this are known as "multitenant" applications. It's possible, of course, that each new subdomain involves the rollout of a new virtual machine. But there also are ways that you can make a single computer, with a single instance of the application, provide the same illusion of an infinite number

of domains. Moreover, doing so is not nearly as difficult as you might think.

In this article, I look at several techniques that make it possible for you to create and maintain such multitenant applications. These techniques can be used in an SaaS product or any other application in which the software can and should respond differently to a variety of hostnames or domain names.

It's All Thanks to HTTP

HTTP, the Hypertext Transfer Protocol, is so ubiquitous that most people barely give it any thought. Even someone like me, who works nearly every day on Web applications, knows that HTTP exists and what it does—and yet, I don't think about it too much. However, multitenant applications owe their existence to growth in the earliest days of the Web.

The first version of HTTP that I encountered, back in 1993, was described as version 0.9. That version was a simpler protocol than the one we know today, but it already included the basic GET and POST actions—that is, you could connect to an HTTP server on port 80, and say:

```
GET /
```

The server would, if all went well, send the contents of its home page (typically formatted with HTML) back to the HTTP client. At that point, the connection would close.

Although HTTP 0.9 worked well for many simple cases, the explosive growth of the Web meant that it wasn't good enough for many complex ones. One particularly common, and particularly painful, case was that of Web hosting companies: HTTP 0.9 required that each Web site have its own IP address. If you set up a Linux-based server with a single IP address but multiple hostnames, it wasn't possible for the HTTP server to distinguish between them.

This changed when HTTP 1.0 was released and required that a "Host" header be sent along with the action and pathname. Now, a simple request looked like:

```
GET / HTTP/1.0
Host: lerner.co.il
```

The first line changed, such that it incorporated the version number of HTTP that was being used. This was done so as to have backward compatibility with HTTP 0.9 clients. The second line was defined to be the first of several "request headers", name-value pairs that could be sent

Given that a server now could distinguish between different hosts, even on the same IP address, it was possible to have a single server provide Web hosting capabilities for any number of different domains and hostnames.

from the client to the server.

These request headers have grown in scope through the years, and now include everything from the hostname to cookies to content type to caching information. But for my purposes in this article, the most important part of this request was the "Host" request header. Given that a server now could distinguish between different hosts, even on the same IP address, it was possible to have a single server provide Web hosting capabilities for any number of different domains and hostnames.

In other words, it was now possible to have the same Web server provide hosting to CompanyA.com and CompanyB.com, without either knowing of or seeing each other. The Web server would know to route requests for CompanyA.com to one directory of programs and HTML files, and CompanyB.com to a second, completely separate directory of programs and HTML files.

This might be obvious to anyone who knows about domains, hostnames and DNS, but from the perspective of the server, it didn't matter if it had to distinguish CompanyA.com from CompanyB.com, or abc.CompanyA.com from def.CompanyA.com. That is, different hostnames within the same domain were treated similarly to different domains. True, DNS and HTTP server configuration files made it easier to send *.CompanyA.com to the same location, but at the end of the day, your HTTP server sees different hostnames and, thus, can react differently.

"Virtual hosts", as they became known, shared an IP address and a computer, and so from the perspective of a programmer or IT manager, they were all under the same umbrella. From the perspective of the outside world, these were completely different Web sites. Perhaps they shared an IP address, and thus a hosting provider, but that was the only thing they had in common.

Multitenant

Today, it's trivial to service different hostnames under the same HTTP server. As I indicated previously, you simply tell Apache (or nginx, or whatever HTTP server you use) that the two hosts exist in different directories, and that they should be treated differently. With such a configuration in place, there is no connection whatsoever between the different hostnames. This actually makes it easier to move Web sites from one machine to another. You scoop up the virtual host's configuration file and move it to another machine, along with the programs and static assets—that is, HTML files and images.

Indeed, a huge industry of cheap, on-demand Web hosting perhaps has made this the most common way servers are allocated and used. Even my own personal server has five to ten different virtual hosts on it at any given time, between personal projects and demos of client applications.

A multitenant application turns this idea on its head. Rather than using a single server, with a single IP address, to service a large number of different applications, each with its own hostname, you will have many different instances of the same application. That is, you'll have both CompanyA.com and CompanyB.com point not only to the same IP address, but also to the same instance of your Web application.

This might sound strange, until you consider that because modern versions of HTTP always pass a "Host" header, and because all of the HTTP request headers are available to a Web application, you can write a single application that will work on multiple hosts. Consider that BigCompany.com has two different divisions and a separate Web site for each division. The site should be completely identical in both cases, except that the contact phone number and address should reflect the coast that the user has reached.

You can use the "Host" request header in an "if" statement inside the application, and thus display the information that is appropriate. This is a classic example of multitenant sites, although it's certainly not the most complex of them.

Multitenant with Sinatra

Let's implement the above scenario using Sinatra, a very small and lightweight Web application framework written in the Ruby language. In the July 2014 issue of *LJ*, I covered a similarly small framework, known as Flask, written in Python. Such frameworks often are perfect for simple sites and example code.

There are a number of ways that you can create a Sinatra application. My preference is to do so in a directory, along with a Gemfile and a config.ru file. This took less than five minutes for me to set up on my own computer. First, I created a directory called "multiatf". In that directory, I created a file called "Gemfile", which is where I will name the Ruby gems I'll be using for this application:

```
source 'https://rubygems.org'
gem "sinatra", :require => "sinatra/base"
gem 'shotgun'
```

The first line says that I want to retrieve gems from Rubygems.org, the official and standard location. The second line says that I want to use the "sinatra" gem, but that I don't want to require "sinatra", but rather "sinatra/base". Finally, I name the "shotgun" gem, which provides for automatic reloading of Sinatra apps—precisely the sort of thing I want when I'm developing an application.

Before continuing, I then run `bundle install`, which ensures that all of the gems named in the Gemfile have been installed. It creates a file named "Gemfile.lock", which lists the precise names and versions of each gem I'll be using in my application. This list includes those gems I have named explicitly and those upon which my named gems depend. It is worth taking a look at Gemfile.lock sometime; it may well give you insights into how your Sinatra and Rails applications work.

Next, I write a "config.ru" file, sometimes known as a "rackup file", which tells Rack—Ruby's standard interface between HTTP servers and applications—where my application's code is located and how to execute it. The file looks like this:

```
require 'bundler'

Bundler.require

require './multiatf.rb'
run Sinatra::Application
```

The first line loads the "bundler" gem. Bundler is an increasingly indispensable gem in the Ruby world, in that it manages the versions of your gems for you, ensuring that they will not require conflicting versions of a gem. After loading Bundler, you then use the "require" class method, which reviews your Gemfile.lock and loads the gems named within.

Next, the "require" statement reads a Ruby file named "multiatf.rb" in the current directory. That is the actual application code, and it's the file I

will be writing and modifying most of all. Loading it means that Ruby will read the contents of the code. In the case of my Sinatra app, that means taking the various "get" and "post" declarations and turning them into the appropriate routing map, such that the appropriate code block is executed for each URL.

Then, once the application has been loaded, config.ru invokes `Sinatra::Application`. That starts the application up and running.

The final step in putting the application together is the multiatf.rb file. This also consists of very little code, but potentially could be quite large:

```
require 'sinatra'

get '/' do
  "Hello from server '#{request.host}'"
end
```

The first line loads the Sinatra code. Next is something that looks vaguely like a method definition, but isn't. Rather, it tells Sinatra that if someone makes a request to the / URL, it should return a string. In this case, the string isn't static, but rather contains a dynamic portion, including the value of "request.host". As you can imagine, this value will vary according to the hostname you are using.

To start this up on my development machine, I ran:

```
shotgun multiatf.rb
```

This produces output telling me that Shotgun is now running my application on port 9393, using Ruby's built-in WEBrick server. I can now go to my Web browser, and load up http://localhost:9393, and because of the `get /` declaration in my Sinatra file, that method is fired. I get a nice message telling me:

```
"Hello from server 'localhost'"
```

But, what if it isn't localhost? What if I go to another server name? For example, I added the following two lines to my /etc/hosts file:

```
127.0.0.1 atf1
127.0.0.1 atf2
```

In other words, when I tell my Web browser to go to host "atf1", it'll go to 127.0.0.1, and it will send, in the "Host" HTTP request header, the server name "atf1". The output then will be:

```
Hello from server 'atf1'
```

The same will be true for "atf2".

In many cases, showing different hostnames isn't enough. You may want to show a different business name or a different address.

Showing Different Content

Thus, you've seen how you can have different output, based on the value of the server name. This seemingly simple fact opens the door to the entire world of multitenant systems. For example, you could imagine a company doing business under a variety of names, which would want to have the same Web application running, but showing the current domain name. All you have to do is change your strings, or your templates, to reflect the current hostname.

In many cases, showing different hostnames isn't enough. You may want to show a different business name or a different address. In order for that to happen, you'll need some additional data. The best and most scalable way to do this is a relational database, but you can simulate one with a Ruby hash that will be good enough for the purposes of this article.

In this case, let's define the hash such that it contains two keys, one for each of the hosts to recognize. Then, let's pull out the company's name from the hash, based on the key.

I thus change multiatf.rb to read as follows:

```
require 'sinatra'

hosts = {'atf1' => {name: 'First ATF site',
                    address: '111 Main Street'},
         'atf2' => {name: 'Second ATF site',
                    address: '222 Elm Street'}
        }

get '/' do
  "Welcome to '#{hosts[request.host][:name]}', located at
    ➥'#{hosts[request.host][:address]}'!"
end
```

The idea here is simple, but the effects are profound. This is how each domain can appear different, even if the content is the same. You can imagine going even further than this, pulling in a different CSS stylesheet to an HTML page based on the hostname, or having it show different pictures.

If you are using a relational database, you can enter each new

tenant site in a table, giving each a unique ID number. You then can use that ID as a foreign key, adding (for example) this "site_id" value in a table describing merchandise. For example, one of my clients manages about 30 different sites, each with its own set of real-estate offerings. These 30 sites are actually running on a single Web application, with a single database. However, based on the hostname through which the user enters the site, the software displays a different set of properties. This has made the site and the software easy to manage, scale and grow. Each time a new site needs to be added, the biggest task is to update the SSL certificate, such that it includes the new hostname. Otherwise, the system works automatically, with the (nontechnical) company managers able to create new sites within several minutes, merely by filling out an HTML form. That form allows them to add a new entry into the "sites" table. The hostname is used to look up the site ID, whose value is then used to display properties.

Next month, I'll continue with this topic, discussing not only how you can have the same site produce similar content, but how you can configure it such that different users can manage their own sites, without interfering with the overall software and functionality.■

Reuven M. Lerner is a Web developer, consultant and trainer. He recently completed his PhD in Learning Sciences from Northwestern University. You can read his blog, Twitter feed and newsletter at http://lerner.co.il. Reuven lives with his wife and three children in Modi'in, Israel.

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

Resources

The Sinatra home page is at **http://sinatrarb.com**.

For information and ideas about multitenant sites in Ruby on Rails, you might want to read *Multitenancy with Rails*, an e-book written by Ryan Bigg and available at **https://leanpub.com/multi-tenancy-rails**. Although the book specifically addresses multitenancy with Rails, it offers many ideas and approaches that are appropriate for other software systems.

DAVE TAYLOR

Power Shell History and the `find` Command

Dave explores Bash command-line history and delves into the super-powerful find command.

It has been a while since I spent some time looking at basic command-line features and how they tie in to shell scripts, so I thought it'd be a good time to go back to basics. Let's start with some fundamentals of command-line history, actually.

If you're like me, you're stuck in a rut, either using `!xx` to repeat the most recent command starting with that letter or those letters, or using the arrow up/down keys to step through your command history and find a specific command. I definitely get into a `!v !cc !.` loop when I'm developing software, for example—useful.

But, there's a lot more that your Bash shell can do in terms of manipulating your command history. A common one is to search for a specific command, then repeat it with the command's numeric ID, like this:

```
$ history | grep find
  213   find $HOME -name "*.zip" -print
$ !213
```

Make a typo on the command? You can use arrow keys to monkey around and fix it, but that can be tedious. Instead, use the ^old^new replacement sequence:

```
$ pc oldfile newfile
-bash: pc: command not found
$ ^pc^cp
cp oldfile newfile
$
```

It's super helpful, and notice that Bash echoes the corrected command so you know what's going on too.

One of my favorites is to repeat

all the arguments of the previous command but not the command itself. Use `!*`, like this:

```
$ vi file1 file2 file3
$ cp !* ~
```

Got that? The `cp` command will copy each of the three files to your home directory for safekeeping.

You also can select individual parameters by index number too, as I demonstrate here:

```
$ echo listening to the soundtrack from cloud atlas
listening to the soundtrack from cloud atlas
$ echo !!:3
echo the
the
$ echo !!:3-5
-bash: :3-5: bad word specifier
```

Hmm...what went wrong on that second example? Did you figure it out? The `!!` applies to the command immediately before the current command, so the long string of `listening to the soundtrack from cloud atlas` is *not* what's being modified, and there are only two words in the immediately previous command (`echo the`).

There are two ways to get the latter example to work, either have it come immediately after the longer echo statement or tap into the power of Bash history modifiers even further by referencing the command by its number:

```
$ history | grep cloud atlas
  508   echo listening to the soundtrack from cloud atlas
$ echo !508:6-7
echo cloud atlas
cloud atlas
$
```

Obviously, the shell's habit of echoing the expanded command before it actually executes it makes these examples a bit annoying, but they demonstrate the concept, and quite frankly, sometimes when you're working with the shell, a bit of redundancy isn't such a horrible thing anyway.

The Incredible find Command

Enough command-line history though. Let's shift to the rather insanely complicated `find` command instead. There's actually quite a bit you can do with `find`, and that's why it ends up being a really complex utility. Worse, its command-line parameters are specified in a format quite different from most Linux commands because they're generally in -parameter value pairs, except the very first parameter, which specifies the starting directory for the search.

Unless it's command-line flags that aren't in the -parameter value format.

To make this more fun, there are different versions of `find` floating around, depending on what flavor of Linux you're running—UNIX, NetBSD, Darwin and so on. I'll try to talk about general approaches to working with `find`, but if you encounter hiccups, check your man page (`man find`) to see what your local variant should be.

The most basic use of `find` is to search for filenames that match a specific pattern—for example, all "C" source files in the home directory or any subdirectory thereunder:

```
$ find ~ -name "*.c" -print
```

On modern `find` commands, the `-print` parameter is redundant and can be omitted, but if you've an old-school version of `find`, omitting it might result in zero output—not so useful.

`find` can check a lot of different characteristics of files too. Here's the same search, but this time, let's say you want only C source files that are more than 1K and have been created 30 or fewer days old:

```
$ find ~ -name "*.c" -ctime -30d -size +1k -print
```

And now, here's an explanation, because I know you need one.

Times can be specified in seconds, minutes, hours, days or weeks by specifying "smhdw", respectively. Size is specified in kilobytes, megabytes, gigabytes, terabytes and petabytes with "kmgtp", respectively.

Then, a specified value defaults to an exact match, so `-ctime 30d` will match only files that were created exactly 30 days earlier—not quite so useful. So, the `-` prefix means "less than or equal to", and `+` means "greater than or equal to". Got it?

You also can search for files by permission string, which is darn useful for administrators. In fact, a very common search that you should do occasionally is this:

```
$ find / -perm -4000 -user root -print
```

This introduces yet another parameter: `-owner`. There's also a group value if you want to check group ID, although in modern Linux that seems to be far less utilized than it was back in the old days—progress, or something like that.

`find` has an `-exec` parameter that's worth exploring too, because it lets you invoke arbitrary Linux commands on each and every matching file. So you started with a basic command to list all C source files by looking for matches with the simple pattern `*.c`, but the output of that is just the filenames—boring.

Instead, `-exec` can let you output an `ls -l` result for each one just as easily:

```
$ find ~ -name "*.c" -exec ls -l '{}' \;

-rw-r--r--  1 taylor  staff  250 Jul  4 09:07
/Users/taylor/Documents/Books/Wicked-Cool-Scripts-10th-Ed/
➥Scripts.new/077-epoch.c

-rwxrwxrwx  1 taylor  staff  631 Sep  9  2009 /Users/taylor/
➥Documents/Linux Journal/distance-between.c

-rw-r--r--  1 taylor  staff  377 Apr  6  2012 /Users/taylor/
➥Documents/Linux Journal/fixdict.c

-rw-r--r--@ 1 taylor  staff  6848 May  1  2013
/Users/taylor/Documents/old.intuitive.com/intuitive.com/
➥scripts/adt/get-adt-news.c

-rw-r--r--  1 taylor  staff  6848 Jun  8  2013
/Users/taylor/Documents/server cli files and
archives/scripts/adt/get-adt-news.c

-rw-r--r--  1 taylor  staff  3085 Aug  1 22:47
/Users/taylor/Library/Application Support/Plex Media
Server/Plug-ins/Framework.bundle/Contents/Resources/
➥Platforms/Shared/Libraries/tornado/epoll.c
```

It seems like pretty complicated output, but really, it's just an `ls -l` with a lot of subdirectories making things look more complicated. You can handle that, right?

The parameters required for `-exec` are tricky. The `{}` sequence is replaced with the matching filename, but without the quotes, it'll be preprocessed and removed. And, the weird `\;` sequence is because `-exec` needs to know how far to read on the parameter list, so it reads until the `;`. But again, that'd be interpreted by Bash before even being given to `find` without the backslash—hence `\;`.

So, a really common way for sysadmins to search for setuid root programs is a slight variant on the above:

```
$ find / -perm -4000 -user root -exec ls -l '{}' \; |
   mail -s "setuid programs and scripts" admin
```

That'll send an e-mail report that lists all files with setuid and owner root to the e-mail address "admin".

That's it for this article. Next time, I'll talk about the `find|xargs` command pipe pair and the substantial problem with files and directories that contain spaces.■

Dave Taylor has been hacking shell scripts for more than 30 years. Really. He's the author of the popular *Wicked Cool Shell Scripts* (and just completed a 10th anniversary revision to the book, coming very soon from O'Reilly and NoStarch Press). He can be found on Twitter as @DaveTaylor and more generally at his tech site http://www.AskDaveTaylor.com.

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

KYLE RANKIN

Dr Hjkl on the Command Line

Who needs arrow keys when you have a perfectly good home row?

The first time I used vi was in a college programming course. It was the default editor on the computer lab's UNIX systems we used to compile our assignments. I remember when our professor first introduced vi and explained that you used the hjkl keys to move your cursor around instead of the arrow keys. Before this point, I was a pico user (that dates me a bit now), and it seemed so backward to me that vi used hjkl instead.

It wasn't until I became a heavy vim user that I began to appreciate the speed you gain from navigation keys appearing on home row. As a touch typist, I realized the arrow keys are in a no-man's land outside the home row compared to hjkl, and even though vim supported arrow keys, I used hjkl instead. I've been pleased to discover a number of different programs that also support the same level of key bindings.

I've written a few other columns in *Linux Journal* through the years along those lines (all with Dr Hjkl in the title), and here, I've decided to revive Dr Hjkl for another round of time-saving command-line navigation tips that will help keep your hands on the home row and off those arrow keys.

Most of my tips in this article are about reducing your reliance on the arrow keys and increasing your speed when on the command line. For many years, whenever I would find a mistake in a command I typed, I would do one of two things: use a combination of Home, End and the arrow keys (all way too far away from home row) to move the cursor back to the mistake so I could fix it, or sometimes I found it was faster to press Ctrl-C and type the whole command again. One day I observed another Linux user fly back and forth across words on the command line and realized there was a better way.

On the command line, you can replicate the behavior of dw with Alt-d.

Moving Between Words

The first simple speed improvement is the use of Alt-b and Alt-f to move backward or forward one word on the command line. This behaves somewhat like the b and w keys in vi to skip between words instead of one letter at a time. Alt-b acts like just like b in vim. Press Alt-b, and the cursor will move back one word and sit at the first letter of the previous word. Alt-f is slightly different; the cursor moves forward until it ends up at the space between words instead of at the beginning of the following word. So given the following command:

```
ls -l /var/log
```

If my cursor were at the end of the line and I pressed Alt-b, it now would be over the l in log. If I pressed it again, it would move to the v in var. If my cursor were at the beginning of the line and I pressed Alt-f, it would end up on the space between ls and -l. This annoys me enough that often I'll find myself going forward *an extra* word and then pressing Alt-b so the cursor is where I want it. Even though it's more keystrokes than the right-arrow key, it keeps my hands on the home row. Alternatively (as you'll see later), I simply could press Ctrl-f to move forward an extra letter instead.

Delete Words

The bulk of the time that I use Alt-b and Alt-f on the command line is to correct a typo earlier in the line. Now, you certainly could move the cursor over to the right position and then use Delete or Backspace to erase the error, but for minor mistakes, I've found it's much faster to delete the whole word and retype it. In vim, I would type cw to change the word under my cursor, or a slightly slower approach is to type dw to delete the word and then enter insert mode to make my changes. On the command line, you can replicate the behavior of dw with Alt-d. The Alt-d key will remove the word under the cursor completely so you can retype it. So, taking the same example from above:

```
ls -l /var/log
```

If my cursor were at the end of the line and I realized I wanted to change

`-l` to `-ltr`, I would press Alt-b three times to move it over the l in -l, then I would press Alt-d to delete the l, and finally type ltr.

Replace Home and End

It turns out the arrow keys aren't the only ones on the hit list. Home and End, although useful, also are out in that no-man's land away from the home row. In vi, you would just use ^ or $ to go to the beginning or end of the line. On the command line, you can replace Home and End with Ctrl-a and Ctrl-e, respectively. If you use screen or tmux with screen key bindings, you know that Ctrl-a already is called for, so you'll need to press Ctrl-a a to send that Ctrl-a through to the command line. Ctrl-a can be particularly useful when you realize you need to pipe some initial command through something you've already typed, and Ctrl-e is useful to move back to the end of the line afterward to finish your command after editing it somewhere in the middle.

A Few Final Shortcuts

Although the above shortcuts are enough to get started, before I finish, I want to highlight a few extra shortcuts that are less useful but worth knowing all the same. First, although Alt-b and Alt-f move backward and forward a word, respectively, their counterparts Ctrl-b and Ctrl-f will move backward and forward a single letter. Somewhat less useful, but still interesting, is the fact that you can use Alt-u to uppercase the full word under the cursor, Alt-l to lowercase the word, and Alt-c to capitalize the first letter in the word.

With all of these tips, I recommend posting a reminder to yourself somewhere on your computer. It may take a few weeks to ingrain a new habit like this into your command-line use, but once you get used to it, you won't go back. I've found that these shortcuts also apply in command-line clients like Irssi, so if I notice I have a typo in something I'm about to say in IRC, I can just press Alt-b until it's under the cursor, and press Alt-d to delete it, and then correct the error and press Enter.■

Kyle Rankin is a Sr. Systems Administrator in the San Francisco Bay Area and the author of a number of books, including *The Official Ubuntu Server Book*, *Knoppix Hacks* and *Ubuntu Hacks*. He is currently the president of the North Bay Linux Users' Group.

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

SHAWN POWERS

PHP for Non-Developers

Development doesn't have to be scary. With PHP, it's like shell scripting for the Web!

After years of making it clear that I'm not a developer in just about every article I've written here at *Linux Journal*, I do have a confession to make. I can write the "Hello World" equivalent in almost every programming language out there. In assembly, it might have been "1+1", but my lack of advanced skills should be evident. The thing is, I've always wanted to learn how to program, but I hate the process so much I never get past "Hello World".

Then I met PHP.

I know PHP is no longer cool. I know that compared to Python it's extremely limited. But I also know that with PHP, I actually was able to create useful programs from the beginning. I suspect that's why I love Bash scripting so much. With Bash, I always start with a problem to solve and use scripting to solve it. I never "learned" to write Bash scripts, I just did it. My goal in this article is to throw you right into writing useful PHP code. If you really want to make "Hello World" appear in a browser window, by all means do so. But I'm not going to teach you how!

Preliminaries

Server: One great thing about PHP is that just about every Web server has it installed and ready to go. If you get advanced and want to start making system calls to the underlying Linux OS, you might have to tweak php.ini a bit, but getting a PHP platform is usually as simple as installing a Web server with a LAMP stack.

One thing that makes programming easier for me is to make sure my user account has write access to the place I'm writing code. My process usually involves making a small change and clicking refresh on the browser to see if it worked. If you have to `sudo cp localfile serverfile` every

time you make a change, it will be no fun. Worry about proper file permissions when you release your code to the Internet, but not while you're developing.

The only other server tweak I recommend is adding a .htaccess that turns on error reporting. I don't always have errors display on the page, but if you get the dreaded "php blank page", it's nice to turn on error reporting quickly so you get a nice message telling you where you forgot a semicolon. I usually create a file called "err" and quickly rename it to ".htaccess" when I want to see errors displayed. You probably will have to modify your Apache server, telling it to "AllowOverride" for the folder you're using, but it's worth the effort. Here's what my .htaccess file looks like:

```
php_flag  display_errors        on
php_value error_reporting       2039
```

Editor: PHP code is just text. You can use any text editor you like. I highly suggest using a text editor that does syntax highlighting though, because it makes it much easier to spot typos. I usually use vim when I do Bash scripting, but with PHP, I find myself scrolling more, so something like gedit works better for me. If you really want to get fancy, programs like Sublime or SlickEdit are amazing, but really gedit is fine. Just make sure whatever editor you use has syntax highlighting, it's awesome. (If you need to edit remote files, I'd check out using SSHFS to mount the remote folder and edit them as if they were local files. It works great in gedit.)

The Goal

For this first foray together into the world of programming, let's create something useful, but still simple. I also want to start with something worth building on to. So let's make a start page for your browser. Remember iGoogle? Well, you're going to start with something far more simple, but because you're making it yourself, you can expand later. (I find coding to be a perfect way to spend nights of insomnia, just saying.)

How PHP Works

It took me a little while to understand this concept, so I'm going to explain it quickly. PHP code can be added to standard HTML, or it can be strictly PHP code that creates HTML as output. These following two snippets of code do the same thing.

test.php:

```
<html>
<body>
<h1>The current day is: <?php echo date("l"); ?>!</h1>
</body>
</html>
```

test2.php:

```
<?php
echo "<html>";
echo "<body>";
echo "<h1>The current day is: " . date("l") . "!</h1>";
echo "</body>";
echo "</html>";
?>
```

Feel free to copy that into two separate PHP files and browse to the page. You should see something like Figure 1. The reason I spelled this out is because different people code PHP in different ways. Just remember that if it's not inside a `<?php ?>` container, it is treated like plain-old HTML. Anything inside those `<?php ?>` containers are executed, and the output is sent to the browser window. Often, PHP code won't have any output, and so nothing is displayed on the screen. But, if you want to use PHP, it must be inside those tags, namely: `<?php PHP STUFF HERE ?>`.

Let's Make Something!

I created a simple landing page in PHP. (Figure 2 shows its output.) You can

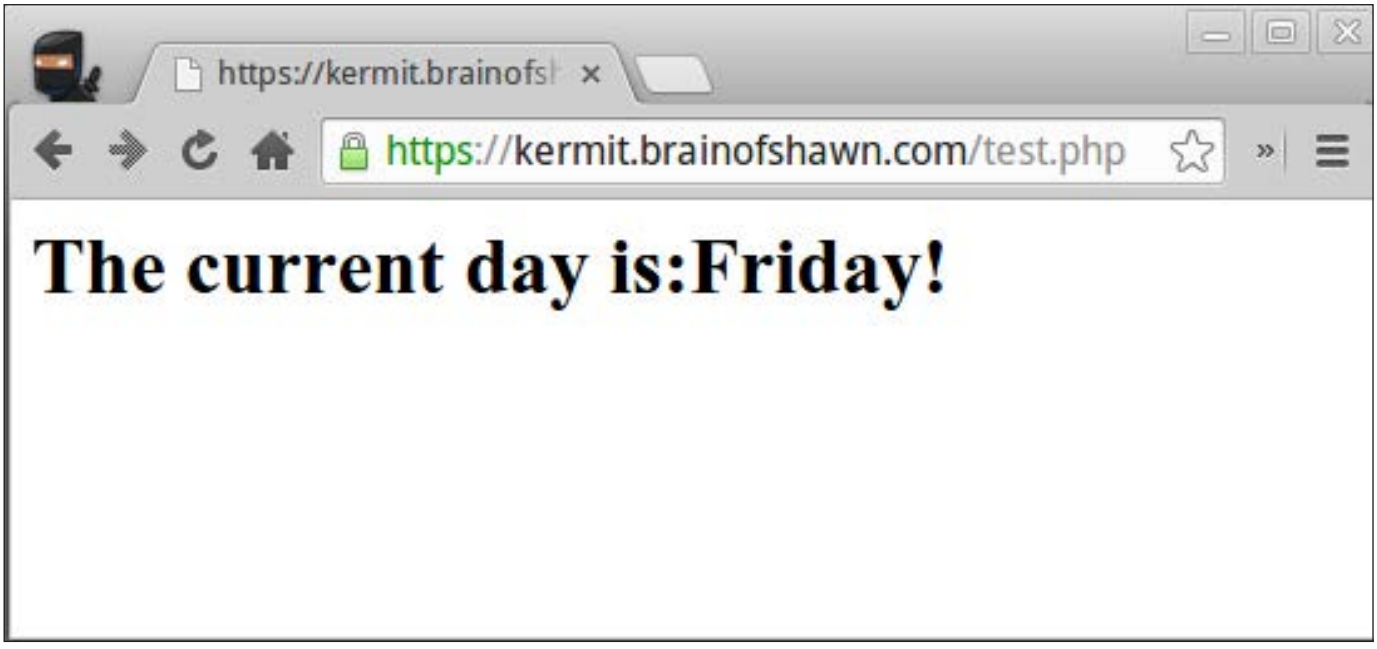


Figure 1. It's not "Hello World", but I'll admit, it's dangerously close.

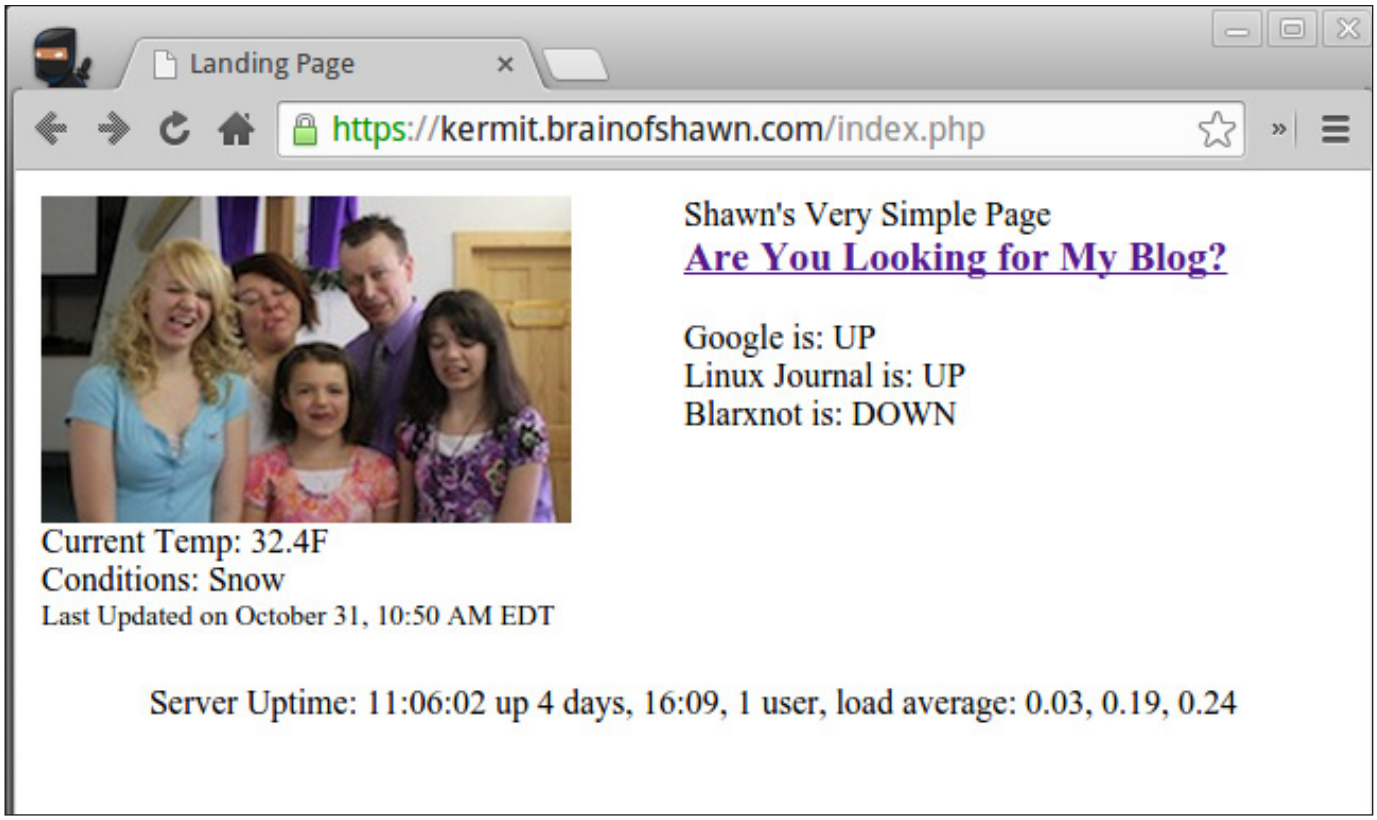


Figure 2. This is the portal. Your family probably will be less silly looking!

download it at **http://snar.co/ljphp**. It's also shown in Listing 1. Unzip the file, and put it in your Web server so you can edit it and make it work for you.

Notice the first line has `<?php` in it—that means it starts with a section of PHP code rather than HTML. The first thing you do is load a weather API into a PHP array. You'll need to register on Wunderground to get your own API key (they're free), and then modify the URL to fit your location. Basically, it's a two-step process: you load the JSON string into a variable called `$json` and then use the `json_decode` function to break that apart into an array. Check out the Working with Arrays sidebar for some tips on looking at the array itself so you can see what it contains.

After that little bit of variable work, the PHP section closes with a `?>` on line 8. From that point on, it's mostly HTML coding that should look fairly clear. Basically, I created a simple table. In the first cell, there is a picture of my family (you can add your own fam.jpg to the same folder where the php file lives), and then I open

another section of PHP code. Here I print some of the information from the Weather API. You can figure most of it out, but of particular note are the periods in the echo statements. If you want to print more than one thing on the screen, put a dot in between them outside the quotation marks.

After some more HTML code, there is a section that checks Web servers to see if they're up. It's important to know that "is_up" is not a built-in PHP

Listing 1. Example Landing Page

```
<?php

// This is a comment. The next lines fetch weather info
// and then turn it into a PHP array (You'll need your
// own API key!)

$json=file_get_contents("http://api.wunderground.com/
➥api/YOUR_API_GOES_HERE/conditions/pws:1/q/49749.json");
$weather = json_decode($json, TRUE);
?>

<html>
<head><title>Landing Page</title></head>
<body>
<table style="text-align: left; width: 100%;" border="0"
 ➥cellpadding="2" cellspacing="2">
    <tr>
        <td>
            <img style="width: 250px;" alt="The Fam!"
              ➥src="fam.jpg"><br>
            <?php
                echo "Current Temp: " .
                 ➥$weather['current_observation']['temp_f']
                  ➥. "F<br>";
                echo "Conditions: " .
                 ➥$weather['current_observation']['weather']
                  ➥. "<br>";
                echo "<small>" . $weather['current_observation']
                ➥['observation_time'] . "</small><br>";
            ?>
        </td>
    <td style="vertical-align: top;">
                Shawn's Very Simple Page<br>
        <big><strong><a href="http://www.brainofshawn.com">Are You
         ➥Looking for My Blog?</a></strong></big>
                <br><br>
                <?php
                        echo "Google is: ";
                        is_up("www.google.com");
                        echo "<br>";
                        echo "Linux Journal is: ";
                        is_up("www.linuxjournal.com");
                        echo "<br>";
                        echo "Blarxnot is: ";
                        is_up("wwww.blarxn0t.com");
                ?>
        </td>
    </tr>
    <tr>
        <td colspan="2">
        <?php echo "<br><center>Server Uptime: " .
         ➥shell_exec("/usr/bin/uptime") . "</center>"; ?>
        </td>
    </tr>
</table>
</body>
</html>

<?php
    function is_up($host,$port=80,$timeout=1)
    {
    $fsock = fsockopen($host, $port, $errno, $errstr, $timeout);
    if ( ! $fsock )
            {
            echo "DOWN";
    }
    else
    {
        echo "UP";
    }
    }
?>
```

function, rather it's something I wrote to make the code cleaner. If you look at the bottom of the file, you'll see where I defined the function, and you can see what it does. The only thing it checks is whether it can connect to port 80 on the specified address. I made up blarxn0t.com so one would fail. Hopefully no one buys blarxn0t.com just to spite me! (Note: don't try to visit it, with my luck, somebody bought it and made it porn!)

Working with Arrays

Arrays can be really confusing. In this article, I showed loading a JSON API feed into a PHP array, and then referenced a couple fields from inside that array. But how on earth do you know how to build those variable names like `$weather['current_observation']['temp_f']`? Thankfully, there's a great tool for displaying arrays on the screen. Here's a snippet of code that takes a copy of the JSON code from this article's example and displays the entire array so you can see the names of the array indexes. You should be able to see how I created that variable name above:

```
array_view.php
<?php
$json = file_get_contents("http://snar.co/jsonapi");
$weather = json_decode($json, TRUE);
echo "<pre>";
print_r($weather);
echo "</pre>";
?>
```

When you load this page, it should display the entire array in an easy-to-read format in your browser. The `<pre>` tags are important; otherwise, it just jumbles the output all into a single line. Whenever I'm trying to figure out the contents of an array, I make a quick PHP file like this so I can look at it. Hopefully, you'll be able to find the variable indexes, and it will make sense.

The last section of the PHP document does a really cool PHP trick and executes a local shell command, dumping the output to the browser screen. So the results of the `uptime` command are printed at the bottom of the landing page. If you have a Windows server, it probably will just error out, but I'm assuming we're all using Linux here.

To Infinity and Beyond!

This first simple PHP page is just that, very simple. I touched on a few advanced subjects though, and I hope you explore more from here. Arrays are really cool and make working with APIs from Web sites a breeze. If you liked this little taste of PHP, I recommend you look into the following:

- IF/THEN/ELSE conditionals.
- For loops.
- While loops.
- Reading and writing files on the system.
- GET and POST variables.
- Functions.
- Including other PHP files.

But, I urge you—don't just look them up and memorize how they work. Come up with a problem and then solve it! Everything I've learned in PHP started with a Google search like, "How do I display large numbers with commas in PHP", or "How do I round numbers in PHP", or even "How do I multithread queries in PHP for parallel processing?" (That last one is by far the most complex thing I've done so far!)

PHP coding is incredibly fun. I can't believe I'm saying that, but it's true. If you love Bash and can't imagine living without it, I suspect you'll love PHP, because it's a lot like Bash, but a Web browser gets to be the input/output mechanism! If you'd like me to do more project-based learning with programming languages, please let me know. I'd love a reason to come up with fun projects, but it only makes sense if everyone wants to play along!■

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 e-mail at shawn@linuxjournal.com. Or, swing by the #linuxjournal IRC channel on Freenode.net.

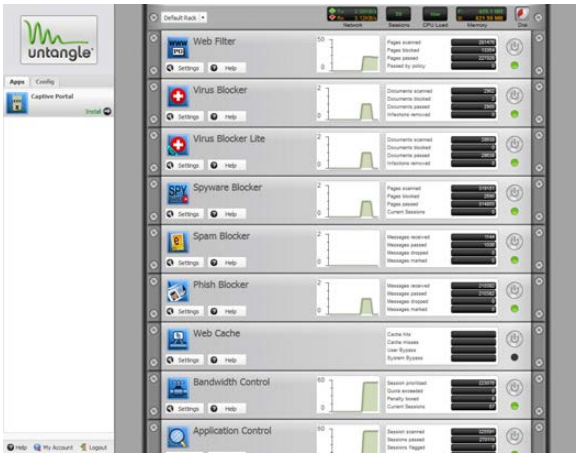
Send comments or feedback via http://www.linuxjournal.com/contact or to ljeditor@linuxjournal.com.

Opengear's CM7100 Console Server

Enterprises need solutions that help them leverage their existing equipment and tools securely and efficiently, asserts Opengear, provider of critical infrastructure management solutions. This should occur, Opengear adds, without myriad logins and security procedures when accessing multiple devices. This philosophy underpins Opengear's new CM7100 Console Server, a state-of-the-art appliance that offers dramatically simplified console management, high reliability and strong security for maximizing the uptime of enterprise networks. The CM7100 is a 16–48 port serial console server that enables network administrators to manage sprawling and complex infrastructure of switches, routers, PDUs, firewalls, servers, UPSes and other critical data-center equipment from a wide array of vendors securely and efficiently. Other key product features include enterprise-grade security with FIPS 140-2 certified SSL, SSH and VPN; dual LAN ports for main and secondary networks; and audit/trail logging to 4GB local storage or a remote log server.
http://opengear.com

The Icinga Project's Icinga Exchange

The new Icinga Exchange is a central repository for plugins, files, programs and add-ons that extend Icinga and compatible projects like Nagios, Shinken or Naemon. The repository allows users to search, store and rate Icinga-compatible extensions easily and connect with the people behind them. The project's Solr-based search engine and navigation tags ensure that searches are quick and easy. Users can choose from manual upload or an automated GitHub sync, the latter "taking the fuss out of maintaining a second project site", notes the Icinga Project. Currently more than 500 projects are available on Icinga Exchange, many of which migrated to the new repository with the previously established Monitoring Exchange repository. The Icinga Project also emphasizes its openness to plugins and add-ons that are compatible with related monitoring tools.
http://exchange.icinga.org



Untangle NG Firewall

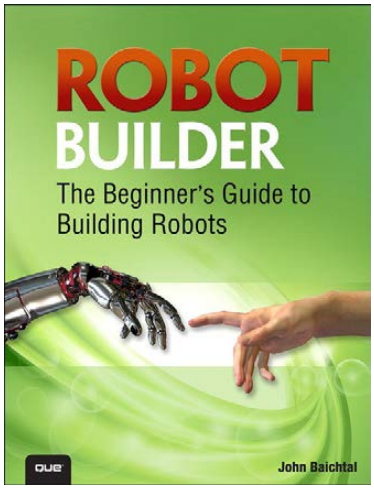
The network software and appliance specialist, Untangle, Inc., recently announced a faster, more agile version 11 of its Untangle NG Firewall software. Untangle claims that its NG Firewall product brings a combination of enterprise-grade capabilities and consumer-oriented simplicity to the management of every aspect of network control. Most notably, version 11 of the solution adds performance gains to the Virus Blocker to address the ever-present and growing threats of malware and to the Spam Blocker to keep unwanted e-mail at bay. In addition, besides an updated kernel, Untangle further added new technologies from the company's IC Control product, improving both HTTPS processing and Captive Portal.
http://untangle.com

SSH Communications' Secure Shell HealthCheck

Since news of the Shellshock Bash vulnerability broke, revelations about the impact of the bug have grown, including reports of lost or stolen Secure Shell (SSH) keys being used to access sensitive information covertly. In reaction to Shellshock, SSH Communications Security announced the release of Secure Shell HealthCheck, a solution for discovering major risk and compliance violations related to all aspects of SSH in data-center environments. SSH HealthCheck helps to reduce the risk of failed audits and cyber-security breaches from hackers or malicious insiders. SSH Communications Security, whose chief innovation officer is SSH inventor Tatu Ylönen, recommends properly scanning and managing SSH keys, including comparing and auditing them against approved keys. The company claims that only its tools can achieve this.
http://www.ssh.com

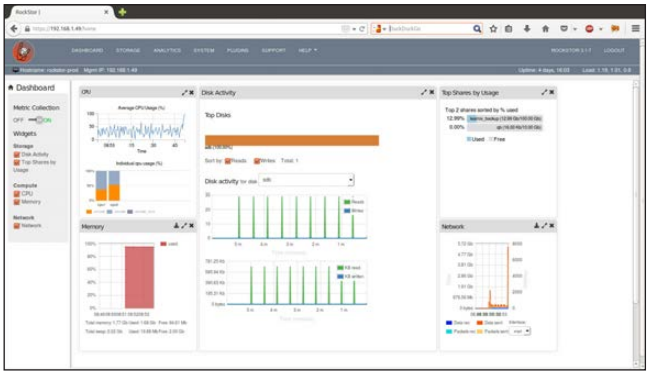
iTrinegy's NE-ONE

The NE-ONE solution was developed by iTrinegy to confront the demands that come from users' accessing applications across myriad devices and mixed networks. NE-ONE enables businesses to understand, predict and manage the performance of applications across today's diverse networks, keeping the performance of networks and applications at their best. iTrinegy says that with NE-ONE, it has eliminated the complexity of dealing with the ecosystem of public, private, cloud, mobile and virtual networks. NE-ONE also offers a modular design, so customers use just the solutions they need, when they need them. NE-ONE comes in two variants: Edge is the ready-to-go hardware-based solution, and Flex is the agile virtual appliance-based solution that can be deployed quickly wherever it's needed on to a network.
http://www.itrinegy.com



John Baichtal's *Robot Builder* (Que)

The topic of DIY robots is so us, fellow Linuxers. Publishing house Que says that author John Baichtal's new book *Robot Builder: The Beginner's Guide to Building Robots* will help readers learn the craft of robot building from the ground up in an easy, fun and hands-on way. As the subtitle implies, absolutely no experience is needed. Baichtal has been a veteran author in the DIY [insert geek toy] space for many years and has helped many an intrigued-by-robot geek get started with robotics. Baichtal and Que have brought together a wealth of practical robotics know-how into "one incredibly easy tutorial". Hundreds of full-color photos guide readers through every step and skill and get them building the first working robot (of 30 total) in the very first chapter. Throughout the book, skill building grows to expert level, and readers will find themselves powering motors, configuring sensors, constructing a chassis and even programming low-cost Arduino microcontrollers.
http://www.informit.com



Rockstor, Inc.'s RockStor NAS

In classic Linux style, the new Rockstor Network Attached Storage solution from the team at Rockstor, Inc., is free, open-source and versatile. Supporting all popular file-sharing protocols, including NFS, Samba/CIFS and SFTP, Rockstor is available as a complete Linux distribution in ISO file format or a USB image and can be installed on bare metal or as a virtual machine. Rockstor's underlying Linux distro is CentOS and the supported filesystem is BTRFS. A pluggable Smart Probe mechanism and a growing list of smart probes provide detailed on-demand information about the various aspects of storage infrastructure. Rockstor supports advanced features for businesses of all sizes ranging from small to large enterprises—for example, snapshots, fast cloning, thin provisioning, dynamic volume management and replication over WAN. The solution comes with a clean, user-friendly Web UI to manage storage operations conveniently.
http://www.rockstor.com

Eltechs ExaGear Desktop

Eltechs hopes to lure ARM-based mini-PC users to try its ExaGear Desktop, which the company says can run virtual Intel x86 apps 4.5 times faster than competitor QEMU. Eltechs ExaGear Desktop is a virtual machine that implements a virtual x86 Linux container on ARM-based mini PCs, enabling the direct and simultaneous running of both x86 Linux and native applications. In addition, MS Windows applications can be run via the Wine emulator. The current version does not support applications that require kernel modules nor does it support 3-D hardware.
http://eltechs.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.

READERS' CHOICE AWARDS 2014

We added a couple great new categories this year, so be sure to read on to see the results!

SHAWN POWERS

It's time for another Readers' Choice issue of *Linux Journal*! The format last year was well received, so we've followed suit making your voices heard loud again. I couldn't help but add some commentary in a few places, but for the most part, we just reported results. Please enjoy this year's Readers' Choice Awards!

BEST LINUX DISTRIBUTION

Although this year the Debian/Ubuntu-based distros took the lion's share of the votes, the "Best Linux Distribution" category is a bit like "Best Kind of Pizza"—even the bottom of the list is still pizza. It's hard to go wrong with Linux, and the wide variety of votes only proves how many different choices exist in our wonderful Open Source world.

16.5% Ubuntu

16.4%	Debian	2.5%	Xubuntu	.4%	Tails
11%	Linux Mint	2.3%	Other	.3%	Android-x86
8.5%	Arch Linux	1.6%	Red Hat Enterprise Linux	.3%	Bodhi Linux
8.3%	Fedora	1.4%	NixOS	.3%	Chakra
6%	CentOS	1.3%	elementary OS	.3%	Kali Linux
5.3%	openSUSE	1.2%	Lubuntu	.3%	PCLinuxOS
4.1%	Kubuntu	1%	CrunchBang	.3%	SolydK
2.9%	Gentoo	.7%	Mageia	.1%	Mandriva
2.7%	Slackware	.4%	LXLE	.1%	Oracle Linux

BEST MOBILE LINUX OS

Android is such a dominant force in the mobile world, we decided to allow Android variants to be counted separately. So although the underlying system on some of these are indeed Android, it seems far more informative this way.

37.1% Stock Android

27.6%	Sailfish OS	3%	Ubuntu Phone	.8%	Replicant
20.2%	CyanogenMod	1.5%	Amazon Fire OS	.8%	Tizen
3%	Other	1.4%	Ubuntu for Android		

BEST LINUX SMARTPHONE MANUFACTURER

29% Samsung

26.7%	Jolla	5.3%	LG	1%	GeeksPhone
16.5%	Nexus	3.7%	Sony	.6%	Amazon
7.1%	Other*	1.8%	Nokia		
7%	HTC	1.4%	Huawei		

**Under "Other", Motorola got many write-ins, followed by OnePlus.*

BEST LINUX TABLET

35.3% Google Nexus 7

14.8%	Google Nexus 10
14%	Samsung Galaxy Tab
9.8%	Samsung Galaxy Note
8.4%	ASUS Transformer Pad
6.4%	Other
4.7%	Kindle Fire HD
2%	ASUS MeMO Pad
1.6%	Dell Venue
1.4%	Acer Iconia One
.9%	Samsung Galaxy Note Edge
.7%	Ekoore Python S3

BEST OTHER LINUX-BASED GADGET (NOT INCLUDING SMARTPHONES OR TABLETS)

We are a Raspberry Pi-loving bunch, that's for sure! But really, who can blame us? With the new B+ model, the already awesome RPi is getting sleeker and more useful. I'm no fortune teller, but I suspect I know next year's winner already.

71.4% Raspberry Pi

8.1%	BeagleBone Black
4.3%	Other*
3.7%	Lego Mindstorms Ev3
3.4%	Moto 360
1.7%	Cubieboard
1.7%	Parrot A.R Drone
1.4%	Samsung Gear S
1.1%	Yamaha Motif XF8
.8%	Nvidia Jetson-K1 Development System
.5%	Cloudsto EVO Ubuntu Linux Mini PC
.5%	VoCore Open Hardware Computer
.4%	LG G Watch
.4%	RaZberry
.4%	VolksPC
.2%	IFC6410 Pico-ITX Board
.1%	JetBox 5300

**Under "Other", the most popular write-ins were Odroid and CuBox.*

BEST LAPTOP VENDOR

This category used to be a rating of which vendors worked the best with Linux, but thankfully, now most laptops work fairly well. So, we truly get to see the cream rise to the top and focus on things other than "it works with Linux". It's awesome living in the future.

32% Lenovo

19.3%	ASUS	1.9%	ThinkPenguin	.1%	Eurocom
18.5%	Dell	1.8%	LinuxCertified		
10.6%	System76	1.6%	ZaReason		
7.9%	Other*	1.5%	EmperorLinux		
4.5%	Acer	.3%	CyberPower		

**Under "Other", the most popular write-ins were (in this order) Apple running Linux, HP, Toshiba and Samsung.*

BEST LINUX DESKTOP WORKSTATION VENDOR

36.8% Dell

19.6%	System76	9.4%	Penguin Computing	1.4%	CyberPower
17.5%	Hewlett-Packard	2.7%	ZaReason		
10.3%	Other*	2.3%	Microway		

**Under "Other", most of the write-ins were "Build your own".*

BEST LINUX SERVER VENDOR

31.4% Dell

21.2%	IBM	4.7%	Other*	.7%	Servers Direct
19%	Hewlett-Packard	3%	ZaReason		
9.5%	Supermicro	2.8%	iXsystems		
5.9%	Penguin Computing	1.9%	Microway		

**Under "Other", most of the write-ins again were "Build your own".*

BEST ANDROID APP

26.7% JuiceSSH

24%	Other*	6.7%	Navfree
16.3%	Waze	1.7%	Memrise
14%	feedly	1.3%	QuizUp
7.9%	Google Goggles	.6%	FreedomPop Messaging
7%	Ayat		

**Under "Other", the most popular votes were (in this order) OsmAnd, F-Droid, ConnectBot, K-9 Mail, Google Maps, Google Now, Tasker, Wifi Analyzer, AirDroid, KeePassDroid and Titanium Backup.*

BEST CONTENT MANAGEMENT SYSTEM

34.7% WordPress

25.3%	Drupal	1.1%	ikiwiki
11.1%	Joomla!	.7%	eZ publish
10.5%	MediaWiki	.4%	Wolf CMS
10%	Other*	.3%	Elgg
4.3%	Alfresco	.2%	Blosxom
1.3%	WebGUI		

**Under "Other", the most popular write-ins were (in this order) DokuWiki, Plone, Django and Typo3.*

BEST LINUX-FRIENDLY WEB HOSTING COMPANY

When it comes to Web hosting, it's hard to find a company that isn't Linux-friendly these days. In fact, finding a hosting provider running Windows is more of a challenge. As is obvious by our winner ("Other"), the options are amazing. Perhaps a "Worst Web Hosting" category would be more useful!

22.8% Other*

22.5%	Amazon	2.9%	LAMP Host	.4%	Netfirms
13.1%	Rackspace	2.6%	Hurricane Electric	.4%	Prgmr
10.4%	Linode	.6%	Liquid Web		
6.5%	GoDaddy.com	.6%	RimuHosting		
5.6%	OVH	.5%	Host Media		
5.4%	DreamHost	.5%	Savvis		
4.8%	1&1	.4%	Blacknight Solutions		

**Under "Other", the most write-ins went to (in this order) Digital Ocean (by a landslide), followed by Hetzner, BlueHost and WebFaction.*

BEST WEB BROWSER

Firefox takes the gold this year by a significant margin. Even if you combine Chrome and Chromium, Firefox still takes the top spot. There was a time when we worried that the faithful Firefox would fade away, but thankfully, it's remained strong and continues to be a fast, viable, compatible browser.

53.8% Firefox

26.9%	Chrome	2%	Other	.4%	QupZill
8.1%	Chromium	.8%	SeaMonkey	.2%	Dillo
4%	Iceweasel	.5%	rekonq		
3%	Opera	.4%	dwb		

BEST E-MAIL CLIENT

If I didn't know firsthand how many hard-core geeks live among us, I might accuse Kyle Rankin of voting fraud. His beloved Mutt e-mail client doesn't take top spot, but for a program without any graphical interface, third place is impressive!

44.4% Mozilla Thunderbird

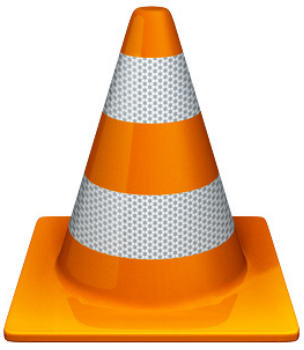
24.7%	Gmail	3.2%	Other	1.7%	Geary
6.8%	Mutt	2.2%	Claws Mail	1%	SeaMonkey
5.5%	Evolution	2%	Zimbra	.9%	Opera Mail
5.3%	KMail	1.8%	Alpine	.4%	Sylpheed

BEST AUDIO EDITING TOOL

69.1% Audacity

10.8%	FFmpeg	1.3%	SoX
9.7%	VLC	1.1%	Mixxx
4.9%	Ardour	.7%	LMMS
1.9%	Other	.5%	Format Junkie

BEST AUDIO PLAYER



We figured VLC would take top spot in the video player category (see below), but it was a bit of a surprise to see how many folks prefer it as an audio player as well. Perhaps it's become the one-stop shop for media playback. Either way, we're thrilled to see VLC on the top.

25.2% VLC

15.3%	Amarok	3.1%	XBMC	.4%	Mixxx
10.4%	Rhythmbox	3%	foobar2000	.4%	MPC-HC
8.6%	Clementine	2.4%	Xmms	.4%	Subsonic
6.1%	MPlayer	1.2%	DeaDBeeF	.3%	Nightingale
5.9%	Spotify	.9%	MOC	.2%	Decibel Audio Player
5.5%	Audacious	.8%	cmus		
4.6%	Banshee	.8%	Ncmpcpp		
4%	Other*	.6%	Guayadeque		

**Under "Other", Quod Libet had the most write-ins.*

BEST VIDEO PLAYER

64.7% VLC

14.5%	MPlayer	1.9%	Kaffeine	.2%	Daum Potplayer
6.4%	XBMC	1.6%	mpv	.1%	Clementine
2.7%	Totem	1.6%	MythTV		
2.7%	Other*	1.4%	Amarok		
2%	Plex	.3%	Xmms		

**Under "Other", most write-ins were for SMPlayer.*

BEST VIDEO EDITOR

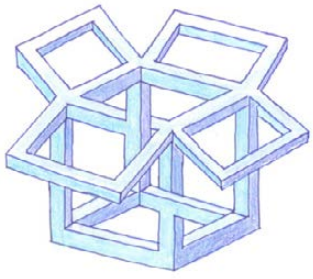
This is another testament to the geek factor when it comes to our readers. We didn't specify "non-linear editor", so by a transcoding technicality, VLC eked out a win in the video editing category. Well played, VLC, well played.

17.5% VLC

16.4%	Kdenlive	7.5%	Cinelerra	1.4%	LiVES
15.1%	Blender	4.9%	PiTiVi	.6%	Shotcut
13.2%	Avidemux	4.8%	LightWorks	.4%	Jahshaka
13.2%	OpenShot	4.7%	Other	.4%	Flowblade

BEST CLOUD-BASED FILE STORAGE

In a category that used to have few options, Dropbox still takes top spot, but the margin is closing. It's hard to argue against Dropbox's convenience and stability, but hosting your own data on ownCloud gives it quite a boost into the second-place spot.



30.5% Dropbox

23.6%	ownCloud	4.4%	SpiderOak		
16%	Google Drive	1.8%	Box		
8.3%	rsync	1%	Copy		
7.5%	Other*	.3%	AjaXplorer		
6.6%	Amazon S3				

**Under "Other", the most write-ins went to Younited and MEGA. Many also said things like "no cloud is the best choice/my files stay on my storage/local only".*

BEST LINUX GAME

I rarely play games, so every year I look forward to this category to find the most popular options for those few times I do. I'm personally tickled to see *NetHack* so high on the list, especially considering the opposition. There's just something about wandering around random tunnels that appeals to the old-school *DnD* player in all of us.

26.5% *Civilization 5*

23.5%	Other*	3.7%	*Scorched 3D*	.9%	*Ryzom*
8.7%	*Team Fortress 2*	3.6%	*Destiny*		
8.4%	*NetHack*	1.9%	*Ultima IV*		
7.1%	*X-Plane 10*	1.8%	*FreeCol*		
6.1%	*Dota*	1.4%	*Kpat*		
5.4%	*Bastion*	1.1%	*FreeOrion*		

**Under "Other", the most write-ins were (in this order)* Minecraft, 0 A.D., Frozen Bubble, Battle for Wesnoth, Portal *and* Counter Strike.

BEST BACKUP SOLUTION

23.5% Other*

18.9%	Clonezilla	3.2%	Back In Time
14.3%	Dropbox	2.1%	luckyBackup
11.7%	Bacula	1.7%	Tivoli Storage Manager
9%	rdiff-backup	.9%	Symantec Backup Exec
8.5%	CrashPlan	.5%	Areca-Backup
5.6%	Amanda	.3%	Storix

**Under "Other", the most popular answer was "my own scripts" or "rsync", followed by BackupPC, Duplicity, Spider Oak, rsnapshot and Deja Dup (in that order).*

BEST VIRTUALIZATION SOLUTION

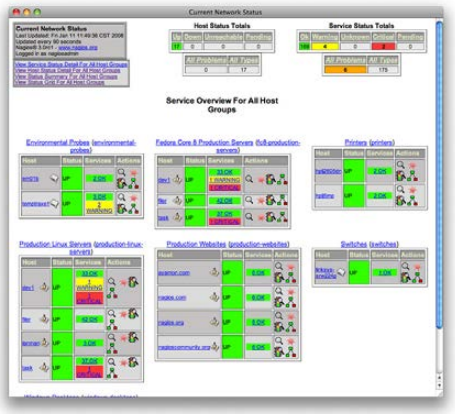
I think the relationship with Vagrant has helped Oracle's VirtualBox significantly in popularity. Yes, Vagrant works with other virtualization platforms, but since it so seamlessly integrates with VirtualBox, I think it gets quite a boost. Virtualization is such an efficient and reliable way to implement systems, bare-metal solutions are almost a thing of the past!

33.4% Oracle VM VirtualBox

22.3%	VMware	4.2%	Other*
21.1%	KVM	1.7%	OpenVZ
5.7%	XEN	1.3%	Linux-VServer
5.3%	QEMU	.1%	Symantec Workspace Virtualization
4.9%	OpenStack		

**Under "Other", the most write-ins went to Docker, ProxMox and LXC, in that order.*

BEST MONITORING APPLICATION



27.1% Nagios

20.7%	Wireshark	.7%	NTM (Network Traffic Monitor)
12.3%	htop	.7%	xosview
10.5%	Zabbix	.5%	Manage Engine
8.6%	Other*	.3%	FlowViewer
6.2%	Zenoss	.2%	Circonus
3.4%	Munin	.2%	SysPeek
2.8%	PC Monitor		
1.9%	New Relic		
1.2%	Opsview		
1%	SaltStack		

**Under "Other", most write-ins went to Icinga and OpenNMS.*

BEST DEVOPS CONFIGURATION MANAGEMENT TOOL

It was interesting to see Git take top spot in this category, because although it certainly would work to use standard version control on configuration files, I always assumed it would be used alongside tools like Chef or Puppet. If nothing else, the DevOps movement has taught crusty old system administrators like myself to treat configuration files like code. Version control is incredible, and it seems as though most readers agree.

39.4% Git

17.2%	Puppet	5%	Chef
8.9%	Ansible	5.4%	SaltStack
8.8%	cron jobs	4.6%	Other*
7.6%	Subversion	3%	CFEngine

**Under "Other", most write-ins went to NixOps.*

BEST OPEN-SOURCE SECURITY TOOL

21.9% Nmap

18.4%	Kali Linux (formerly BackTrack Linux)	6.5%	pfSense	3.7%	Other
15.6%	KeePass	6%	Tails	2.4%	AIDE
13.9%	Tor	4.9%	Netfilter	2.1%	OSSEC
		4.8%	Metasploit		

BEST PROGRAMMING LANGUAGE

30.2% Python

17.8%	C++
16.7%	C
7.1%	Perl
6.9%	Java
4.6%	Other*
4.3%	Ruby
2.4%	Go
2.4%	JavaScript
2.2%	QML
1.4%	Fortran
1.4%	Haskell
1.2%	Lisp
.6%	Erlang
.6%	Rust
.4%	D
.1%	Hack

**Under "Other", most write-ins went to Scala, PHP and Clojure (in that order).*

BEST SCRIPTING LANGUAGE

Python is incredibly powerful, and it appears to be a favorite in both the scripting and programming categories. As someone who knows Bash and a little PHP, I think it's clear what I need to focus on as I delve into the world of development. Meaningful whitespace, here I come!

37.1% Python

27%	Bash/Shell scripts
11.8%	Perl
8.4%	PHP
6.7%	JavaScript
4.9%	Ruby
2.1%	Other
2%	Lua

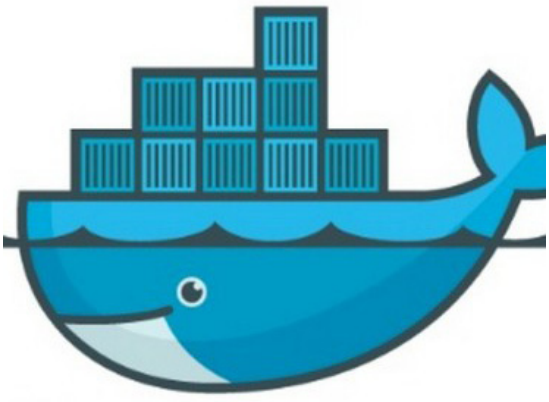
BEST TEXT EDITOR

45.6% vi/vim

11.6%	Emacs
11%	gedit
9.5%	Nano
8.8%	Kate
8.1%	Other*
4.1%	Geany
1.3%	joe

**Under "Other", a large amount of write-ins went to Sublime.*

BEST NEW LINUX/OPEN-SOURCE PRODUCT/PROJECT



Docker is clearly our winner here, and rightly so—what a game-changing technology. It's nice to see Jolla/Sailfish get some love as well. We love Android, but having a choice is a vital part of who we are as Open Source advocates.

28% Docker

19% Jolla and Sailfish OS
7% LibreOffice
5% ownCloud
5% Steam
5% Zenoss Control Center
4% Raspberry Pi
4% Git
3% Apache Cordova/ OpenOffice/Spark/Tika
2% Ansible
2% Elementary OS
2% OpenStack
2% Zabbix
2% CoreOS
2% Firefox OS
1% KDE Connect
1% NixOS and NixOps
1% Open Media Vault

COOLEST THING YOU'VE EVER DONE WITH LINUX

This is my favorite new category for the Readers' Choice Awards. Imagine attending a Linux conference and asking everyone the coolest thing they've done with Linux. That's basically what happened here! We've listed a handful of our favorites, but for the entire list, check out our Web site: **http://www.linuxjournal.com/rc2014/coolest**.

Note: the most common answers were "use it"; "rescue data/photos/whatever off broken Windows machines"; "convert friends/family/businesses to Linux"; "learn"; "teach"; "get a job"; "home automation"; and "build a home media server". The following list is of our favorite more-specific and unique answers, not the most common ones.

- Building my procmail pre-spam spam filter back in the mid-late 1990s.
- 450-node compute cluster.
- 7.1 channel preamp with integrated mopidy music player.
- A robot running Linux (for the Eurobot annual competition).
- Accidentally printing on the wrong continent.
- Adding an audio channel to a video while also syncing it.

- Analyzed NASA satellite data with self-written code.
- Annoyed the cat remotely.
- Automated my entire lighting setup in my house to respond to voice and my mobile apps.
- Automatic window plant irrigation system.
- Bathroom radio.
- Brewing beer.
- Built an application that runs on the International Space Station.
- Built a system for real-time toll collection for a major toll highway system.
- Built our own smartphone.
- Built Web-based home alarm system on Raspberry Pi.
- Cluster of Raspberry Pis to crack encrypted office documents.
- Controlled my Parrot drone.
- Controlled the comms for 186 Wind turbines.
- Controlling my Meade Telescope with Stellarium under Linux.
- Converted my old VHS family videos, using a laptop more than ten years old.
- Created a mesh network in the subarctic.
- Created an ocean environmental sensor buoy with radio data transmitter.
- Discovered new planets.
- Fixed a jabber server in Denver, USA, while in a hotel lobby in Amman, Jordan.
- Got Linus' autograph on a Red Hat 5.0 CD.
- Hacked my coffee machine to send me a text message when the coffee is ready.
- Introduced my daughter to Lego Mindstorm EV3.
- Monitor the temp and humidity of my wine cellar and open the doors when too hot or humid.
- Replaced the controller in my hot tub with a Raspberry Pi.
- Scripted opening and closing of a co-worker's CD tray every 15 seconds for four days.
- Used an LFS system to move ACH transfers for a national gas company.
- Flushed my toilet from another city.
- Remote chicken door.
- Web-based sprinkler controller for 16 stations on a Raspberry PI (also control the pool and yard lights).
- Chaining SSH tunnels together to get from work to home via three hops due to restrictive network settings.
- Built a system that monitors a renewable energy installation with two fixed solar arrays, a two axis sun tracking solar array and a wind turbine. Production and weather data are displayed on a Web site in real time.
- Back in the days of modems, I had my computer call up my girlfriend every morning, so she would wake up and go to work.
- Used a Wii controller, through Bluetooth with my Linux computer as an Infrared Camera, to detect the movement of my daughter's Fisher Price Sit and Spin Pony, and to control a video game.

SINGLE-MOST IMPORTANT TOOL YOU USE AT WORK

Note: this category was write-in only, and many people listed more than one tool. Many people also answered "my brain", "my fingers", "my computer" or "Linux itself", because we failed to specify that it be a Linux tool. Thus, figuring out the percentages for this category was nearly impossible. The list below gives the answers for the most-popular "Single-Most Important *Linux-specific* Tool You Use At Work", in order.

SSH

Vi/Vim	LibreOffice	Bash	Eclipse
Git	Emacs	Terminal/Shell	

While this entire list is full of awesome and useful tools, my needs line up with the majority of readers. SSH and Vim are my bread and butter. Someday, those Emacs people will see the light.

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 e-mail at shawn@linuxjournal.com. Or, swing by the #linuxjournal IRC channel on Freenode.net.

We'd like to make Readers' Choice Awards even better next year. Please send ideas for new categories and any comments or feedback via http://www.linuxjournal.com/contact or to ljeditor@linuxjournal.com.

SharePoint is at the Crossroads — Which Way Will You Go?

SharePoint in the cloud or on premises? Or both? Come to SPTechCon Austin 2015 and learn about the differences between Office 365, cloud-hosted SharePoint, on-premises SharePoint, and hybrid solutions and build your company's SharePoint Roadmap!

For developers, the future means a new app model and new app paradigms. For IT pros and SharePoint admins, it's trying to retain control over an installation that's now in the cloud. For information workers and their managers, it's about learning how to work 'social.' But it's not for everyone.

Where do you need to be?

The answer is simple: SPTechCon Austin. With a collection of the top SharePoint MVPs and expert speakers, more than 80 classes and tutorials to choose from and panels focused on the changes in SharePoint, SPTechCon will teach you how to master the present and plan for the future.

Migrate to SharePoint 2013! Prepare for Office 365!
Build Your Hybrid Model!

SPTechCon
The SharePoint Technology Conference

February 8-11, 2015
Renaissance Austin Hotel

80+ Classes

40+ Microsoft Expert Speakers

Get Your Texas-Sized Registration Discount—Register NOW!

www.sptechcon.com

A BZ Media Event

WEBCASTS

Learn the 5 Critical Success Factors to Accelerate IT Service Delivery in a Cloud-Enabled Data Center

Today's organizations face an unparalleled rate of change. Cloud-enabled data centers are increasingly seen as a way to accelerate IT service delivery and increase utilization of resources while reducing operating expenses. Building a cloud starts with virtualizing your IT environment, but an end-to-end cloud orchestration solution is key to optimizing the cloud to drive real productivity gains.

> **http://lnxjr.nl/IBM5factors**

Modernizing SAP Environments with Minimum Risk—a Path to Big Data

Sponsor: SAP | Topic: Big Data

Is the data explosion in today's world a liability or a competitive advantage for your business? Exploiting massive amounts of data to make sound business decisions is a business imperative for success and a high priority for many firms. With rapid advances in x86 processing power and storage, enterprise application and database workloads are increasingly being moved from UNIX to Linux as part of IT modernization efforts. Modernizing application environments has numerous TCO and ROI benefits but the transformation needs to be managed carefully and performed with minimal downtime. Join this webinar to hear from top IDC analyst, Richard Villars, about the path you can start taking now to enable your organization to get the benefits of turning data into actionable insights with exciting x86 technology.

> **http://lnxjr.nl/modsap**

WHITE PAPERS

White Paper: JBoss Enterprise Application Platform for OpenShift Enterprise

Sponsor: DLT Solutions

Red Hat's® JBoss Enterprise Application Platform for OpenShift Enterprise offering provides IT organizations with a simple and straightforward way to deploy and manage Java applications. This optional OpenShift Enterprise component further extends the developer and manageability benefits inherent in JBoss Enterprise Application Platform for on-premise cloud environments.

Unlike other multi-product offerings, this is not a bundling of two separate products. JBoss Enterprise Middleware has been hosted on the OpenShift public offering for more than 18 months. And many capabilities and features of JBoss Enterprise Application Platform 6 and JBoss Developer Studio 5 (which is also included in this offering) are based upon that experience.

This real-world understanding of how application servers operate and function in cloud environments is now available in this single on-premise offering, JBoss Enterprise Application Platform for OpenShift Enterprise, for enterprises looking for cloud benefits within their own datacenters.

> **http://lnxjr.nl/jbossapp**

WHITE PAPERS

Linux Management with Red Hat Satellite: Measuring Business Impact and ROI

Sponsor: Red Hat | **Topic:** Linux Management

Linux has become a key foundation for supporting today's rapidly growing IT environments. Linux is being used to deploy business applications and databases, trading on its reputation as a low-cost operating environment. For many IT organizations, Linux is a mainstay for deploying Web servers and has evolved from handling basic file, print, and utility workloads to running mission-critical applications and databases, physically, virtually, and in the cloud. As Linux grows in importance in terms of value to the business, managing Linux environments to high standards of service quality — availability, security, and performance — becomes an essential requirement for business success.

> http://lnxjr.nl/RHS-ROI

Standardized Operating Environments for IT Efficiency

Sponsor: Red Hat

The Red Hat® Standard Operating Environment SOE helps you define, deploy, and maintain Red Hat Enterprise Linux® and third-party applications as an SOE. The SOE is fully aligned with your requirements as an effective and managed process, and fully integrated with your IT environment and processes.

Benefits of an SOE:

SOE is a specification for a tested, standard selection of computer hardware, software, and their configuration for use on computers within an organization. The modular nature of the Red Hat SOE lets you select the most appropriate solutions to address your business' IT needs.

SOE leads to:

- Dramatically reduced deployment time.
- Software deployed and configured in a standardized manner.
- Simplified maintenance due to standardization.
- Increased stability and reduced support and management costs.
- There are many benefits to having an SOE within larger environments, such as:
  - Less total cost of ownership (TCO) for the IT environment.
  - More effective support.
  - Faster deployment times.
  - Standardization.

> http://lnxjr.nl/RH-SOE

Real-Time Rogue Wireless Access Point Detection with the Raspberry Pi

Build a Raspberry Pi-based Kismet sensor network to hunt rouge wireless access points.

CHRIS JENKS

Years ago, I worked for an automotive IT provider, and occasionally we went out to the plants to search for rogue Wireless Access Points (WAPs). A rogue WAP is one that the company hasn't approved to be there. So if someone were to go and buy a wireless router, and plug it in to the network, that would be a rogue WAP. A rogue WAP also could be someone using a cell phone or MiFi as a Wi-Fi hotspot.

The tools we used were laptops with Fluke Networks' AirMagnet, at the time a proprietary external Wi-Fi card and the software dashboard. The equipment required us to walk around the plants—and that is never safe due to the product lines, autonomous robots, parts trucks, HiLos, noise, roof access and so on. Also when IT people are walking around with laptops, employees on site will take notice. We became known, and the people with the rogue WAPs would turn them off before we could find the devices.

The payment card industry, with its data security standard (PCI-DSS), is the only one I could find that requires companies to do quarterly scans for rogue WAPs. Personally, I have three big problems with occasional scanning. One, as I said before, rogue WAPs get turned off during scans and turned back on after. Two, the scans are just snapshots in time. A snapshot doesn't show what the day-to-day environment looks like, and potential problems are missed. Third, I think there is more value for every company to do the scans, regardless of whether they're required.

Later, when I was a network engineer at a publishing company, I found it was good to know what was on my employer's network. The company wanted to know if employees followed policy. The company also was worried about data loss, especially around a couple projects. Other companies near us had set up their own wireless networks that caused interference with the ones we ran. Finally, I had to worry about penetration testers using tools like the WiFi Pineapple and the Pwn Plug. These allow network access over Wi-Fi beyond the company's physical perimeter.

One thing I always wanted was a passive real-time wireless sensor network to watch for changes in Wi-Fi. A passive system, like Kismet and Airodump-NG, collects all the packets in the radio frequency (RF) that the card can detect and displays them. This finds hidden WAPs too, by looking at the clients talking to them. In contrast, active systems, like the old Netsumbler, try to connect WAPs by broadcasting null SSID probes and displaying the WAPs that reply back. This misses hidden networks.

A couple years ago, I decided to go back to school to get a Bachelor's degree. I needed to find a single credit hour to fill for graduation. That one credit hour became an independent study on using the Raspberry Pi (RPi) to create a passive real-time wireless sensor network.

About the same time I left the automotive job, Larry Pesce of the SANS Institute wrote "Discovering Rogue Wireless Access Points Using Kismet and Disposable Hardware" (**http://www.giac.org/paper/gawn/7/discovering-rogue-wireless-access-points-kismet-disposable-hardware/107273**). This was a paper about real-time wireless sensors using the Linksys WRT54GL router and OpenWRT. But, I didn't find out about that until I had already re-invented the wheel with the RPi.

Today lots of wireless intrusion

Hardware

Below is the hardware per sensor—your prices may vary depending on where you buy and what's on sale.

Cost of parts: $69.95 per sensor; I used six Raspberry Pis in the project.

Raspberry Pi Wireless Sensor Drone:

- Raspberry Pi Model B: $35.00 (found on sale for $29.99).
- 5v 1amp power supply: $9.99.
- Plastic Raspberry Pi case: $8.99.
- TP-Link TL-WN722N: $14.99.
- Class 10 SDHC 8-gigabit Flash card: $5.99.

Network:

- Cat 5e cable between 25–50' long: already had.
- Linksys WRT54GL: already had.
- Linksys 16-port workgroup switch: already had.

Monitor and Kismet Sever:

- Laptop running Xubuntu Linux VM: already had.

detection systems exist on the market, but as listed in the Hardware sidebar, mine cost me little more than $400.00 USD to make. Based on numbers I could get, via Google Shopping, using Cisco Network's Wireless IDS data sheet from 2014, a similar set up would have cost about $11,500 USD. I've been told by a wireless engineer I know that he was quoted about twice that for just one piece of hardware from the Cisco design.

When I started looking into using the RPi for this, I kept coming across people using the RPi and Kismet for war driving, war walking and war biking. David Bryan of Trustwave's Spider Labs did a blog post in 2012 called "Wardrive, Raspberry Pi Style!" (**http://blog.spiderlabs.com/2012/12/wardrive-raspberry-pi-style.html**) where he talked about using Kismet with the RPi to track WAPs on his walk and drive around his area. He used a USB GPS device to map out where the access points were.

Wireless Survey

A wireless survey is usually a map of a building or location showing the signal strengths associated with wireless access points. Surveys are usually the first step when a new wireless network is installed. Surveys give the installers how many WAPs are needed, where they should be spaced, and what channels would be best to use in those areas.

Surveys normally are done with a WAP and a Wi-Fi-enabled device. The WAP is placed in a location, and signal strength is recorded as the client is moved around the area.

A rogue WAP or a survey WAP can be built from a Raspberry Pi with a wireless card and Hostapd. Most on-line documentation for a Hostapd WAP says to bridge the network cards on the RPi. This can be skipped if the WAP will not be used with clients that connect to the Internet.

Because the RPis are used as stationary devices, I didn't need GPS. One thing I did need though was a rough idea of where to place the sensors. Based on readings, Wi-Fi is good for about 328 feet (100 meters) with the omnidirectional antenna being used on the TP-Link card indoors. Having the existing wireless survey (or doing one) will be useful (see Wireless Survey sidebar). It will let you know where the existing WAPs are. This information also could come from the network documentation, if it exists. It is important to keep the detectors from being overpowered by approved access points. The wireless survey or network documentation also should provide the BSSIDs for the approved devices to be filtered out.

The RPis have no shortage of operating systems to run. My choice for this project was to run Kali Linux

Kali Linux:

Kali Linux is the new version of Backtrack Linux—one of the specialized Linux distributions for penetration testing and security. It is currently based off Debian Linux, with security-focused tools preinstalled. Kali runs everything via the root login.

Kali has builds available as ISOs and VMware images in 64-bit, 32-bit and custom-built ARM images for single card boards, Chrome OS and Android OS devices.

Airodump-NG

Airodump is a raw 802.11 packet capture device. It is part of the Aircrack-NG suite. Normally, Airodump-NG will capture a file of packets to be cracked by Aircrack-NG. However, in my case, I wanted the feature where Airodump-NG can list the clients and access points it sees around it, which is about 300 feet indoors (this distance is based on documentation by the 802.11 standard).

on the RPi, with Airodump-NG and Kismet. Originally, it was going to be just Kali and Kismet, but I ran into some limitations. The reason I chose Kali for this project, was the hardware drivers for the network card I used didn't need to be recompiled. Kali also came with Airodump-NG preinstalled, and an `apt-get update && apt-get install kismet` took care of installing the rest of what I needed.

Kismet has two modes that can be run. The first is the Kismet Server, which the Kismet User Interface (Kismet UI) connects to (Figure 1). The Kismet UI shows the WAP name, if is an access point or not, encrypted or not, the channel and other information. The "seen by" column is the list of capture sources that saw the WAPs.

What Is Kismet?

Kismet is an 802.11 wireless network detector, sniffer and intrusion detection system. Kismet will work with any wireless card that supports raw monitoring mode, and it can sniff 802.11b, 802.11a, 802.11g and 802.11n traffic (devices and drivers permitting).

Kismet also sports a plugin architecture allowing for additional non-802.11 protocols to be decoded.

Kismet identifies networks by passively collecting packets and detecting networks, which allows it to detect (and given time, expose the names of) hidden networks and the presence of non-beaconing networks via data traffic.

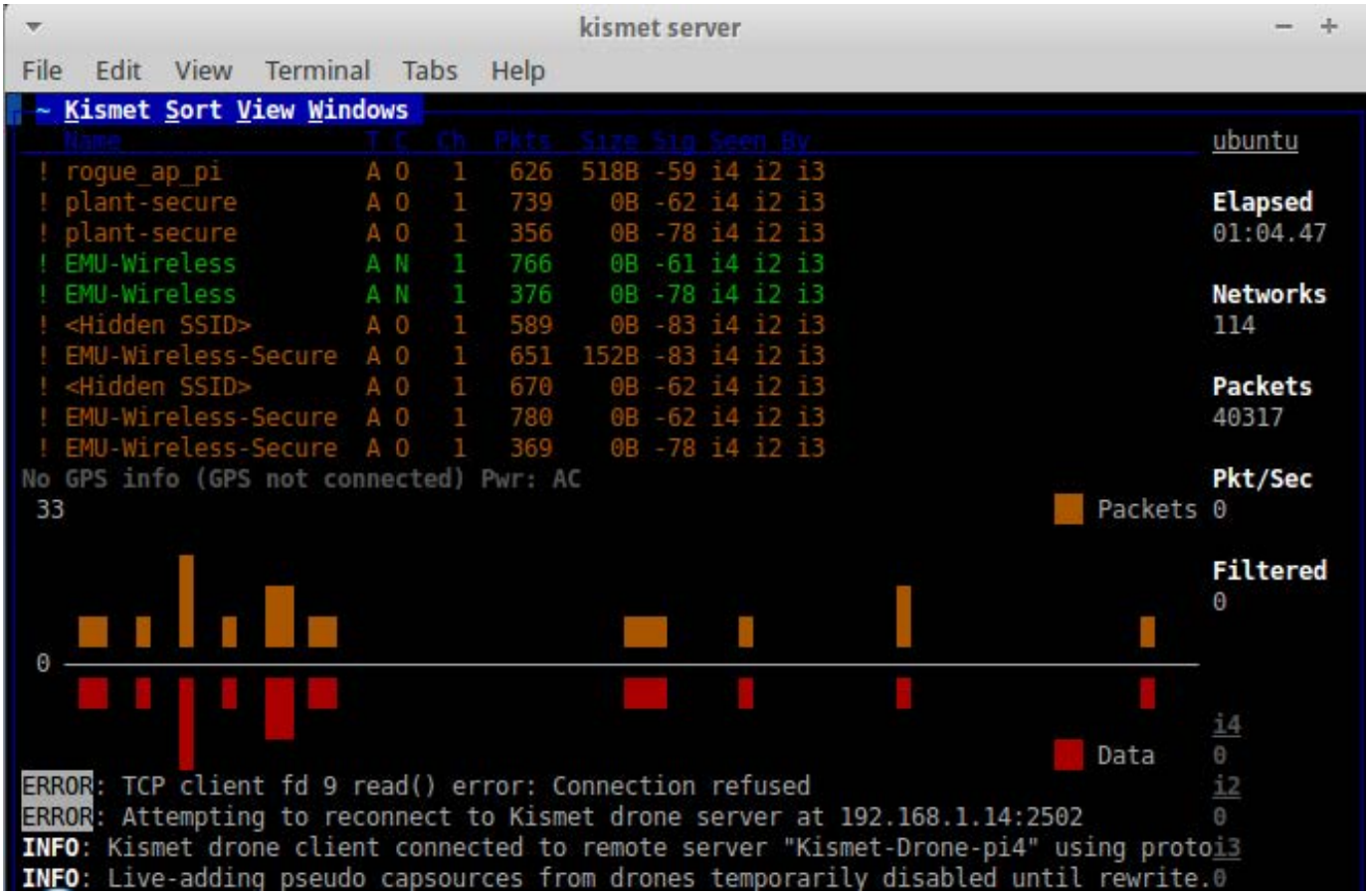


Figure 1. Kismet Network List

Kismet calls the remote sensors drones. They're configured through the kismet drone configuration file in /etc/kismet/kismet_drone.conf. I found the documentation for setting up this part rather sparse. Everything I found spanned multiple years and didn't go into too much detail.

When I configured my drones, I set one up first and then cloned the SD card with the `dd` command. I copied the cloned image to the other SD cards, again using `dd`. To speed up `dd`, set the block size to about half the computer's memory.

Making the drones this way did cause a problem with the wireless network cards. Use `ifconfig` to see what cards the system lists. As you can see from the screenshot shown in Figure 2, my drone02 has the wireless card listed as wlan1, and it is already in monitor mode. After the drives were all cloned, I just had to go in and make minor configuration changes—besides the wireless card change.

The Raspberry Pi uses about 750 mAh, and a 5-volt 1-amp power

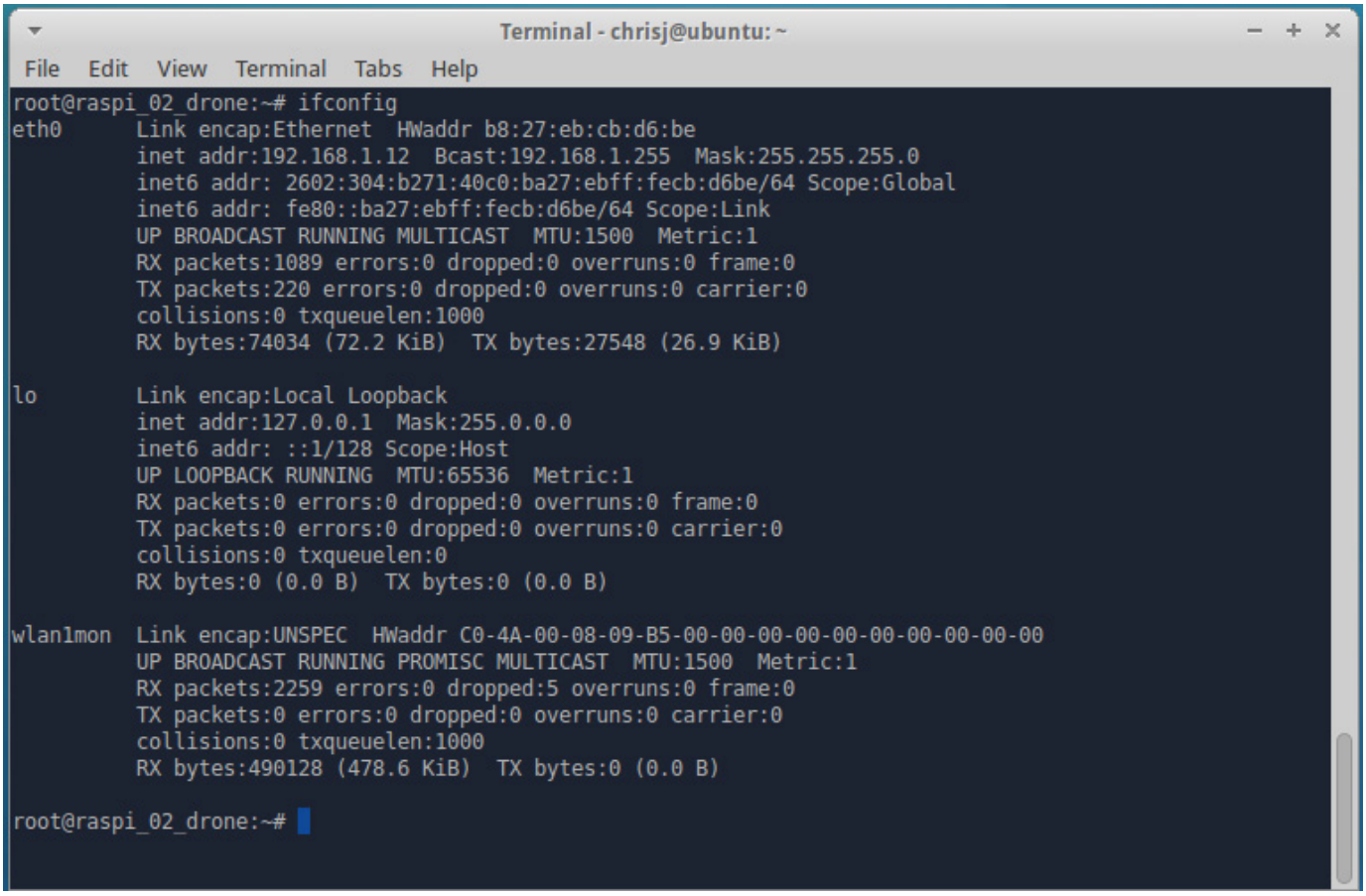


Figure 2. iconfig Screenshot

supply doesn't put enough power out to start the wireless card after the Raspberry Pi is booted. Many of the forums I read said that you need something that puts out 1.5–2.1 amps. I found that plugging in the card first prevents the extra draw, and I didn't have a problem.

In the steps below, if you plug in the Wi-Fi card after booting, you risk a power drop to the Raspberry Pi. The loss of power will crash the RPi, and the SD card could be corrupted.

Configuring the Raspberry Pi with Kali

First, download the Kali Raspberry Pi distro from the Kali Linux Web site.

Copy the image to the SD card with the `dd` command or a tool like Win32DiskImage in Windows. This creates a bit-for-bit copy of the image on the SD card. It is similar to burning an ISO to a DVD or CD.

Put the SD card into the RPi. Then, attach the Wi-Fi card you're using to a USB port.

Attach a Cat5, Cat5e or Cat6 cable

to the Ethernet port. Wired is used to communicate data to the network to prevent problems with the wireless card in monitor mode.

Plug in the micro USB cable to turn on the RPi. Next, `ssh` to the device. You may need to do a port scan with nmap to find the RPi. Alternatively, you can use a monitor and keyboard to access the console directly. Again, have all peripherals plugged in prior to plugging in the power.

The login is "root", and the password is "toor".

Configure the Kismet Drone

Configure eth0 with a static IP address like in the static IP address screenshot (Figure 3). This is done under /etc/network/interfaces.

Restart the networking with `service networking restart` or `reboot`. If you connected via SSH, you'll have to reconnect.

Next, edit /etc/kismet/kismet_drone.conf. I have included a screenshot (Figure 4), but below are settings I used, and the fields that need to be changed.

Set the following either to something that makes sense to you or the right information for your network and device. I used what I have configured for the examples.

Name the drone with `servername`. This will show up in the bottom of the Kismet UI and logs when connecting.

Use `dronelisten` to set the protocol, interface's IP address and port for the drone is to listen on for servers' connection.

List what servers can talk to this drone with `allowedhosts`. This can be a whole network using CIDR notation or just individual boxes on the network, and also allow the drone to talk to itself with

```
root@raspi_02_drone:~# cat /etc/network/interfaces
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
  address 192.168.1.12
  netmask 255.255.255.0
root@raspi_02_drone:~#
```

Figure 3. Static IP Address Screenshot

```
root@raspi_02_drone:~# cat /etc/kismet/kismet_drone.conf
# Kismet drone config file

version=newcore.1

# Name of drone server (informational)
servername=Kismet-Drone-pi2

# Drone configuration
# Protocol, interface, and port to listen on
dronelisten=tcp://192.168.1.12:2502
# Hosts allowed to connect, comma separated.  May include netmasks.
# allowedhosts=127.0.0.1,10.10.10.0/255.255.255.0
allowedhosts=192.168.1.65
droneallowedhosts=127.0.0.1
# Maximum number of drone clients
dronemaxclients=10
droneringlen=65535

# Do we have a GPS?
# gps=true
gps=false

# See the README for full information on the new source format
# ncsource=interface:options
# ncsource=null
# for example:
 ncsource=wlan1:type=ath5k
# ncsource=wifi0:type=madwifi
# ncsource=wlan0:name=intel,hop=false,channel=11

# Special per-source options
# sourceopts=[sourcename|*]:opt1,opt2
# sourceopts=*:fuzzycrypt,weakvalidate

# Comma-separated list of sources to enable, if you don't want to enable all
# the sources you defined.
# enablesource=source1,source2

# How many channels per second do we hop?  (1-10)
channelvelocity=5

# By setting the dwell time for channel hopping we override the channelvelocity
# setting above and dwell on each channel for the given number of seconds.
#channeldwell=10

# Users outside the US might want to use this list:
# channellist=IEEE80211b:1,7,13,2,8,3,14,9,4,10,5,11,6,12
channellist=IEEE80211b:1:3,6:3,11:3,2,7,3,8,4,9,5,10

# US IEEE 80211a
channellist=IEEE80211a:36,40,44,48,52,56,60,64,149,153,157,161,165

# Combo
channellist=IEEE80211ab:1:3,6:3,11:3,2,7,3,8,4,9,5,10,36,40,44,48,52,56,60,64,149,153,157,161,165

root@raspi_02_drone:~#
```

Figure 4. drone.conf

dronealloweddhosts.

Set the maximum number of servers that can talk to the drone with dronemaxclients.

Set the max backlog of packets for the kismet drone with droneringlen. Smaller than what I have might be better. I had problems with drone04 crashing. It also was the one that saw the most networks.

Turn off GPS with gps=false. You don't need it since these are stationary devices and you should know where they are.

Set the capture source, ncsource. This tells the system what interface to use and driver to use for that card:

```
servername=Kismet-Drone-pi2
dronelisten=tcp://192.168.1.12:2502
allowedhosts=192.168.1.65
dronealloweddhosts==127.0.0.1
dronemaxclients=10
droneringlen=65535
gps=false
ncsource=wlan1:type=ath5k
```

The rest of the options I left set to default. Unless you're in a country that uses more B/G channels than the US, there is nothing that needs to be modified.

The last thing to configure on the drones is the /etc/rc.local file. This will start the kismet drone program in the background when the RPi powers on. Add these two lines before the exit 0 so yours looks like the code below:

```
# start kistmet
 /usr/bin/kismet_drone --daemonize

exit 0
```

Configure the Kismet Server on a PC or Server for the Drone Sensors

There are two settings to change in /etc/kismet/kismet.conf. The first is in the ncsource. The second is the filter_tracking.

The line below and related screen capture (Figure 5) tell Kismet what its capture sources are. In this case, nothing local is being used, just the drones. Repeat this line for each drone, with the proper information:

```
ncsource=drone:host=192.168.1.12,port=2502,name=i2
```

The line says the source is a drone, the drone's IP address, what port to connect to on the drone, and what the drone should show as in the Kismet UI. The name is seen "network list" view and "network detail" view. I went with the two-character name of "i#" because the "Last Seen By" field in the network list has a hard-coded limit of ten characters. I wanted that field to

```
# See the README for full information on the new source format
# ncsource=interface:options
# for example:
# ncsource=wlan0
# ncsource=wifi0:type=madwifi
# ncsource=wlan0:name=intel,hop=false,channel=11

ncsource=drone:host=192.168.1.12,port=2502,name=i2
ncsource=drone:host=192.168.1.13,port=2502,name=i3
ncsource=drone:host=192.168.1.14,port=2502,name=i4
#ncsource=drone:host=192.168.1.15,port=2502,name=i5
#ncsource=drone:host=192.168.1.16,port=2502,name=i6
```

Figure 5. Kismet Server Sources

show as many drones as it could.

Next, filter out the known network BSSIDs. Previously, I mentioned that a wireless survey or network documentation should be able to provide this information. As you can see in the network list screenshot, several devices are listed. If you have devices you don't want to see, you'll need to filter them out in the Kismet Server configuration file /etc/kismet/kismet.conf.

In the configuration file, it has an example of:

```
filter_tracker=ANY(!"00:00:DE:AD:BE:EF")
```

In my version of Kismet, that did not work. I had to remove the quotes so the line looked like this:

```
filter_tracker=BSSID(!00:00:DE:AD:BE:EF)
```

The bang (!) ignores that MAC address. This shows everything but ignored WAPs. Without the bang (!), Kismet would show only the WAP with that BSSID in the network list. The choices are ANY, BSSID, SOURCE and DEST. Although the documentation says you can use ANY with a bang (!), trying it fails. The error said to use one of the other three options. The MAC address can be stacked using a comma-separated list:

```
filter_tracker=BSSID(!00:00:DE:AD:BE:EF,!00:0:DE:AD:BE:EE,
➥!00:00:DE:AD:BE:ED)
```

With the Drone Sensor Network running, the network detail screen for an access point will show which drones see the WAP (Figure 6). But, this is a limitation of the system. This screen provides only the signal

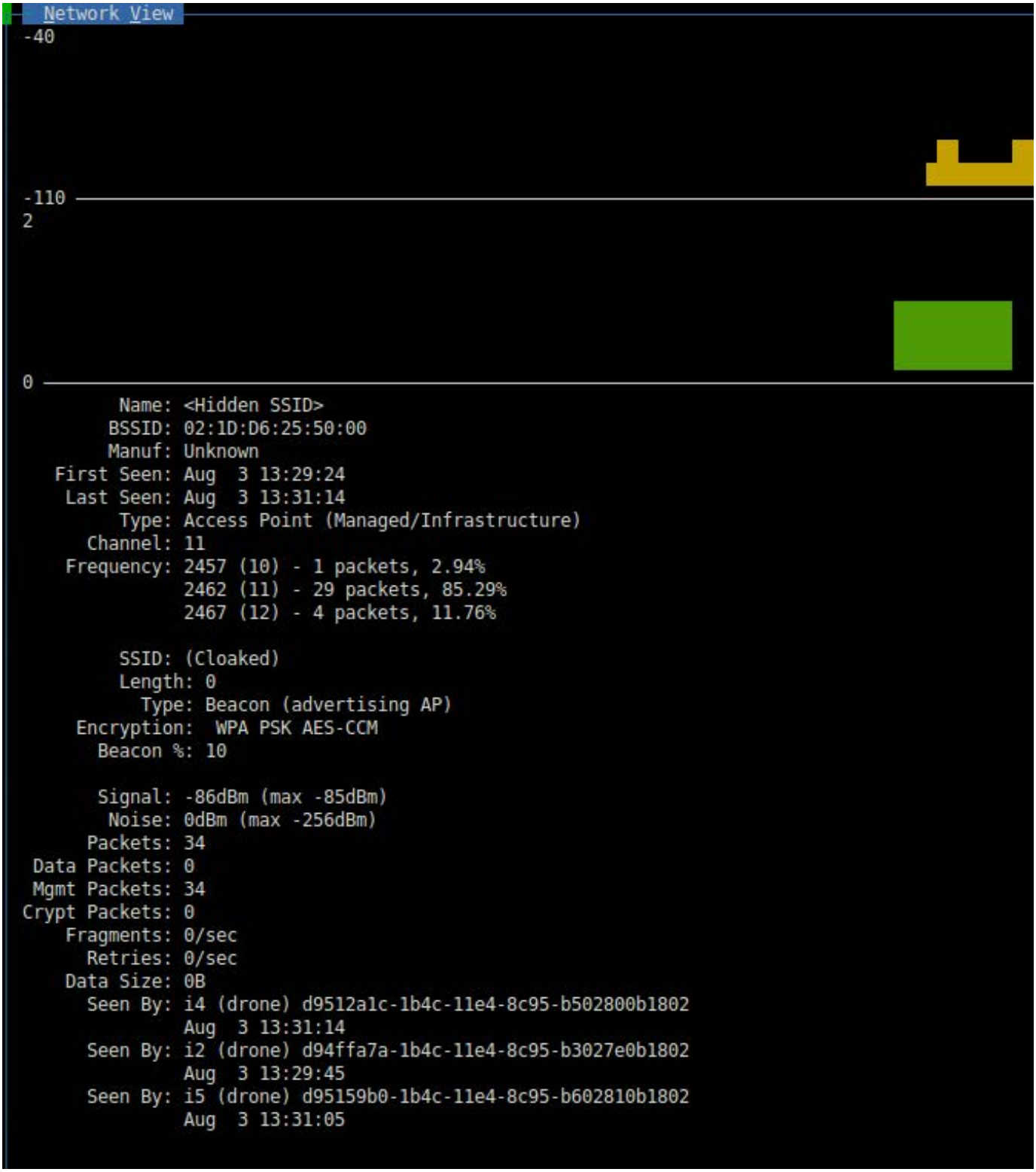


Figure 6. Network Detail Screen

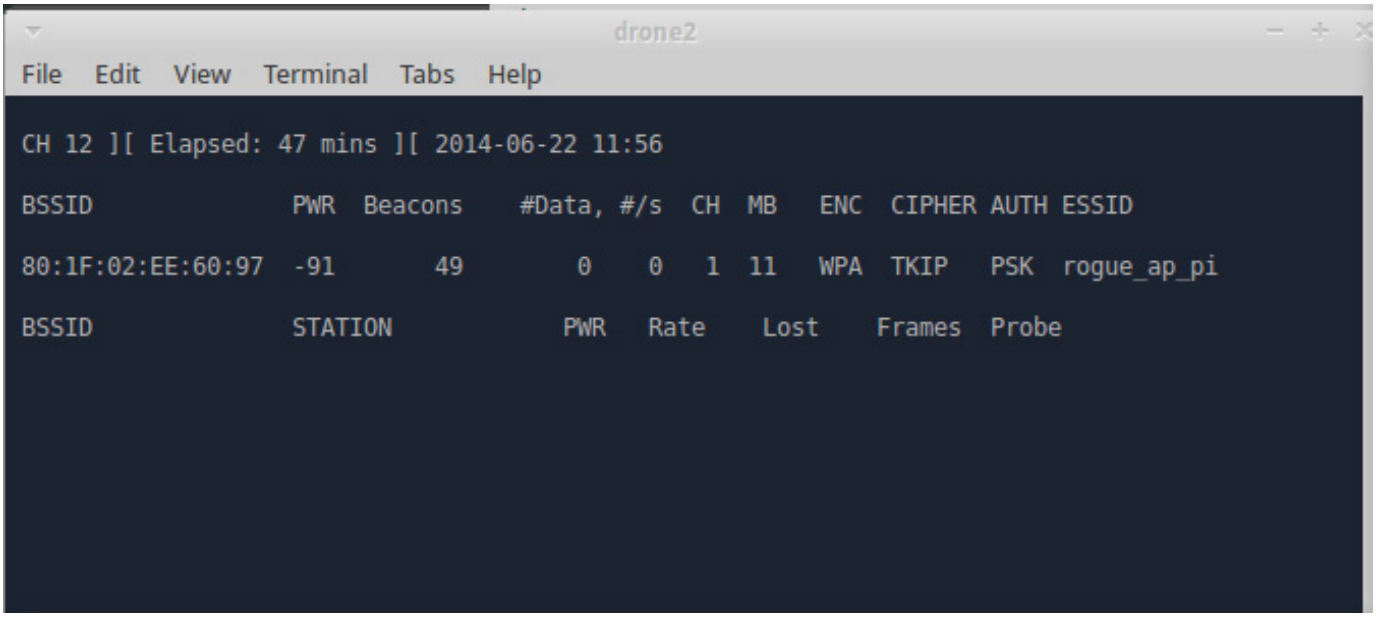


Figure 7. Drone2 airodump

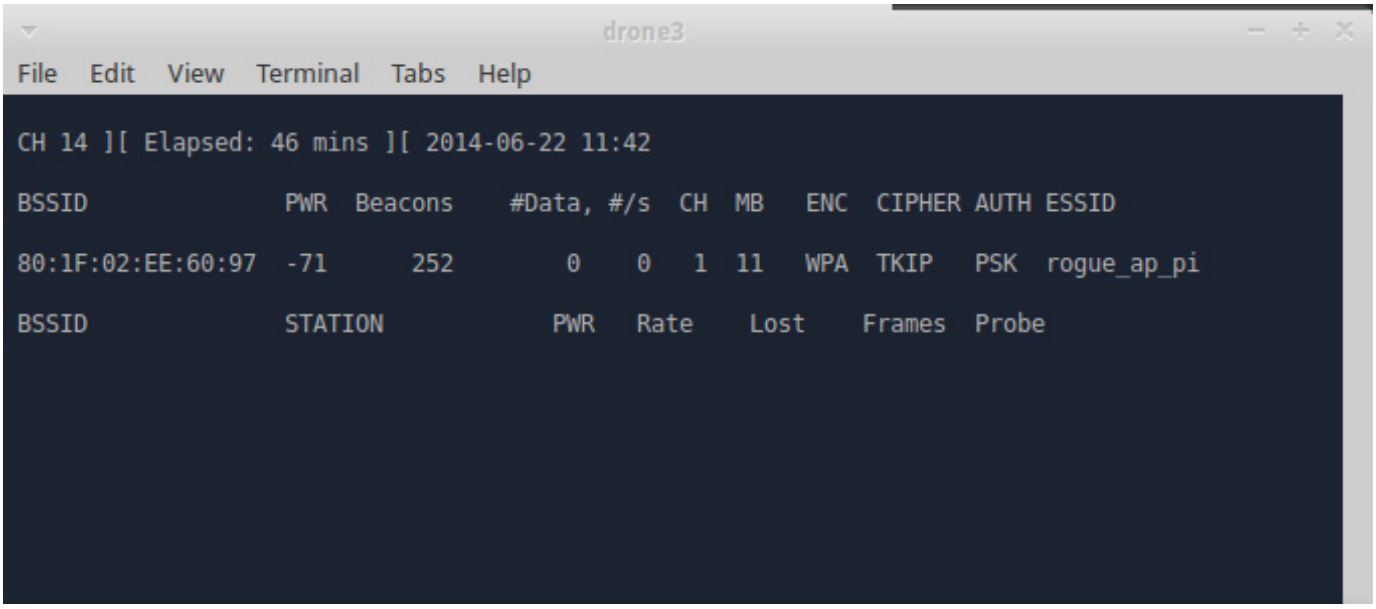


Figure 8. Drone3 airodump

strength for the drone with the strongest signal. This was the limit of Mr Pesce's WRT54GL option.

In Mr Pesce's model, once the rogue WAP was detected, someone had to go out and search. The search area was around all the drones that saw the rouge WAP. Although his design makes the search area smaller than a whole

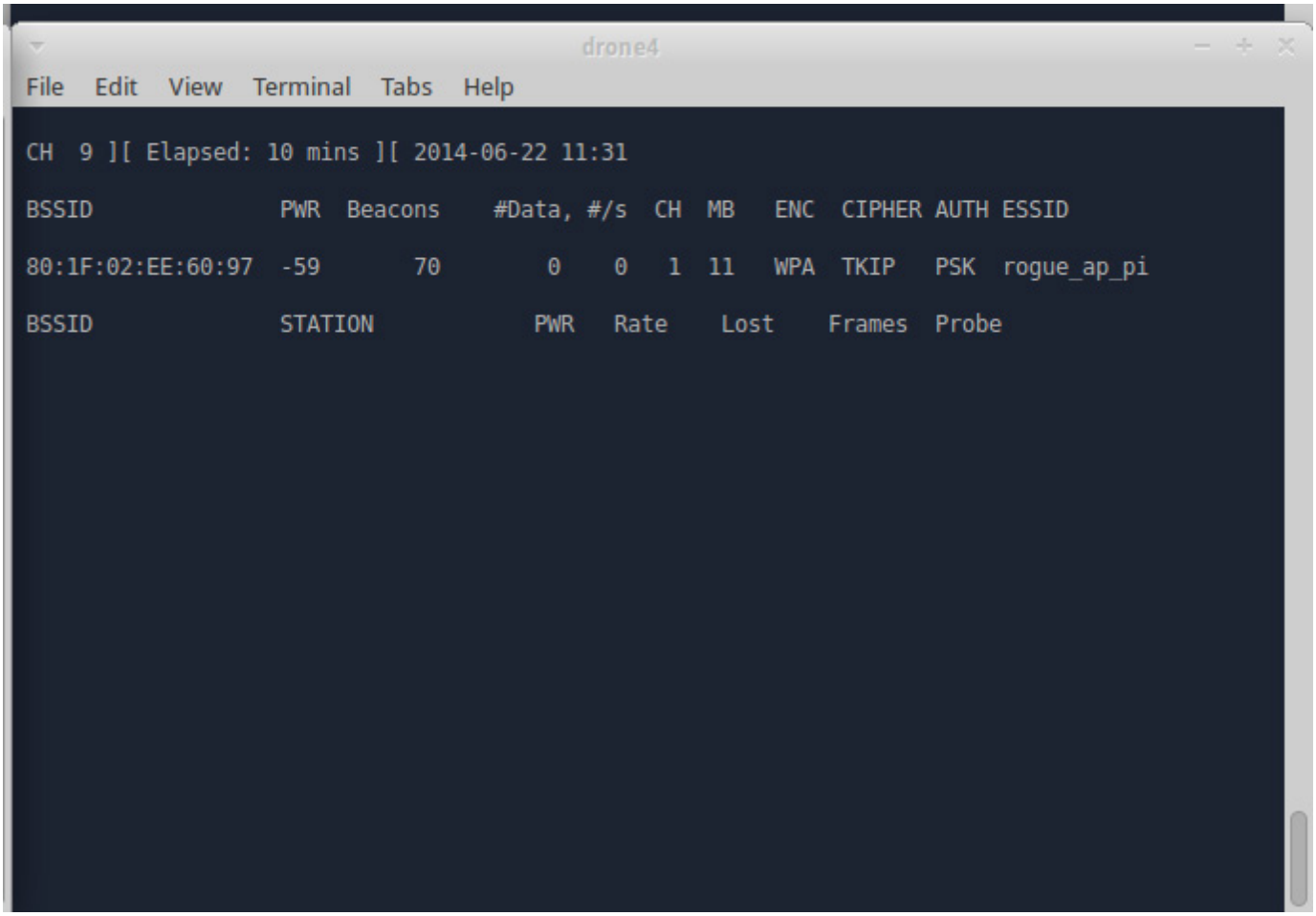


Figure 9. Drone4 airodump

building, it doesn't triangulate very well. By using the RPis as drones, there is a second program you can use for triangulation.

Airodump-NG, as I mentioned before, is for capturing packets on over Wi-Fi. The user interface, when running Airodump-NG, provides several bits of information. The ones you want are BSSID, PWR (power measured in negative DB), Channel and ESSID (Figures 7–9: each image shows a different power level, which when used with the Roosevelt picture, shows how to use it for triangulation).

`ssh` to each RPi drone, and run this command. Don't forget to replace the `bssid` with the MAC address of the WAP you are looking for and the interface for what that drone is monitoring with:

```
airodump-ng --bssid <MAC ADDRESS> <interface>
```

Note: I did not use the channel

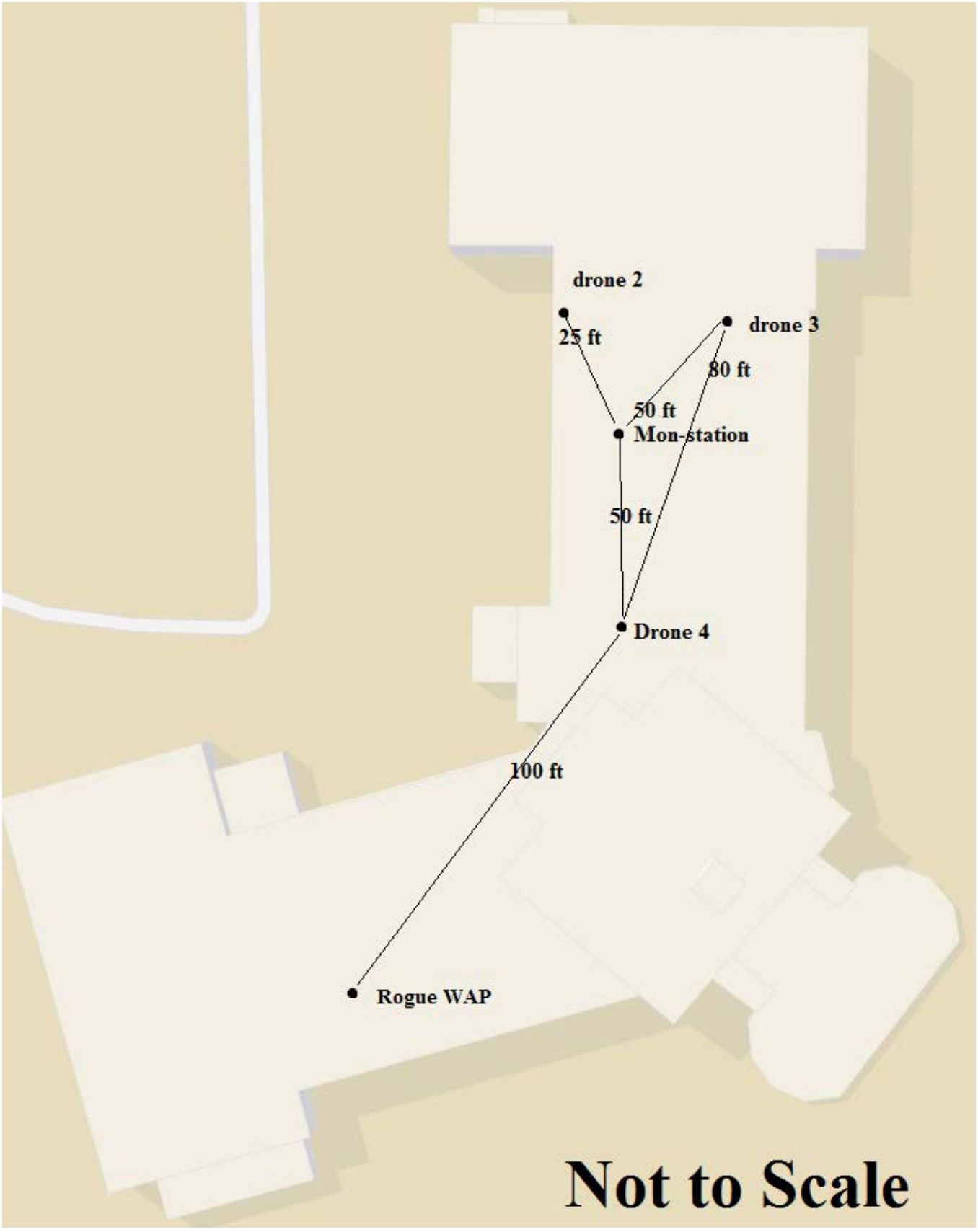


Figure 10. Map of Roosevelt Hall at Eastern Michigan University (Google Maps)

command in the above line to lock a channel. This would interfere with data going back to the Kismet Server and lose the device if the WAP is configured for channel hopping.

Proof of Concept

I did my proof of concept at Eastern Michigan University's Roosevelt Hall, which is where my degree program's labs are. In the map of Roosevelt Hall (Figure 10), there are three drones. This was due to power and Ethernet cable limitations. There is also a rogue WAP (rogue_ap_pi), hidden by my professor. Kismet showed me all the networks in the area, because I didn't have BSSIDs to filter them out. Again, this is where having network documentation or a wireless survey would be helpful.

Drone3 and drone4 are in a hallway. Drone2 is in one of the lab rooms, with the Linksys network and my laptop running the Kismet Server and Kismet UI. When drone4 was just inside the lab's door, there was a 10DB signal loss. Again, a wireless survey would have helped, because it would show how much signal the walls blocked.

Once I had the system up and running and the drones where I wanted them, my professor hid the rogue WAP somewhere on the same floor. By looking at the power levels, I was able to figure out where to go to find the rouge WAP. I knew that drone4 was the closest and that the rogue WAP was on the other side of that drone. I walked down that hallway and found the rogue WAP in less than two minutes. It was hidden under a bench in the hall, outside a classroom and the other lab.

Limitations of the Wi-Fi Card

The last limitation I want to cover is the TP-Link TL-WN722N card. I went with this card because of the cost, the external antenna, the power draw when plugged in to the RPi and its availability at a local store. The card can talk only on the 2.4GHz range, meaning that it picks up only 802.11 B, G and N networks. It does not have the ability to detect or use the 5GHz range used by parts of N, A or the new AC networks.

Although I have a couple ALFA wireless cards, and one that should be able to detect A and AC, I do not know if I could run them on the RPi drone without a separate powered USB hub.

This setup also does not detect Zigbee/Xbee or Bluetooth. Xbee uses both 900MHz and 2.4GHz. Bluetooth uses 2.4GHz. Although both devices use 2.4GHz, frequencies (channels) are outside the Wi-Fi card's range. Mike Kershaw (aka Dragorn, Kismet's developer) is working on a hardware and software Xbee detector called kisbee. An Ubertooth One should work with Kismet to detect Bluetooth.

Cell phones and related cellular network cards also would be missed. Phones operate outside the Wi-Fi card's range, unless the phone is a Wi-Fi hotspot. The new HackRF One card might be able to detect the cellular networks, as well as A/B/G/N/AC Wi-Fi, Xbee and Bluetooth, but I haven't gotten one to play with, and they would drive up the cost to about $300 USD per sensor.■

Chris Jenks is a senior in Information Assurance at Eastern Michigan University. He has more than 15 years of experience in networking and system administration. He has presented on Linux Hardening at DebryCon, GrrCON, and Bsides Detroit; and on Open Source Intelligence at Circle City Con.

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

Resources

Kismet: **http://kismetwireless.net**

Kali Linux: **http://www.kali.org**

The Usability of GNOME

Jim Hall writes about his latest usability study of open-source software, doing a "deep dive" into the usability of the GNOME desktop.

JIM HALL

I work at a university, and one of our faculty members often repeats to me, "Software needs to be like a rock; it needs to be that easy to use." And, she's right. Because if software is too hard to use, no one will want to use it.

I recently spoke at GUADEC, the GNOME Users And Developers European Conference, and I opened my presentation with a reminder that GNOME is competing for mind share with other systems that are fairly easy for most people to use: Mac, iPad, Windows and Chromebook. So for GNOME to continue to be successful, it needs to be easy for everyone to use—experts and newcomers alike. And, that's where usability comes in.

So, what is usability? Usability is about the users. Users often are busy people who are trying to get things done. They use programs to be productive, so the users decide when a program is easy to use. Generally, a program has good usability if it is easy for new users to learn, easy for them to use, and easy for them to remember when they use the program again.

In a more practical view, average users with typical knowledge should be able to use the software to perform real tasks. The purpose of usability testing, therefore, is to uncover issues that prevent general users from employing the software successfully. As such, usability testing differs from quality assurance testing or unit testing, the purpose of which is to uncover errors in the program. Usability testing is not a functional evaluation of the program's features, but rather a practical determination of the program's operability.

Usability testing does not rely on

Usability cannot be addressed only at the end of a software development lifecycle. If you wait until the end, it is usually too late to make changes.

a single method. There are multiple approaches to implement usability practices, from interviews and focus groups to formal usability testing. Whatever method you use, the value of usability testing lies in performing the evaluation during development, not after the program enters functional testing when user interface changes become more difficult to implement. In open-source software development, the community of developers must apply usability testing iteratively throughout development. Developers do not require extensive usability experience to apply these usability practices in open-source software.

I prefer the formal usability test, and it's not hard. You can gain significant insight just by gathering a few testers and watching them use the software. With each iteration, usability testing identifies a number of issues to resolve and uncovers additional issues that, when addressed, will further improve the program's ease of use. Usability cannot be addressed only at the end of a software development lifecycle. If you wait until the end, it is usually too late to make changes.

How to Run a Usability Test—Usability Testing in GNOME

I recently worked with the GNOME Design Team to examine the usability of GNOME. This was an opportune moment to do usability testing in GNOME. The project was in the process of updating the GNOME design patterns: the gear menu, the application menu, selection mode and search, just to list a few. I assembled a usability test to understand how well users understand and navigate the new design patterns in GNOME. Here is how I did that.

The first step in planning a usability test is to understand the users. Who are they, and what tasks do they typically want to perform? With GNOME, that answer was easy. The GNOME Web site explains that "GNOME 3 is an easy and elegant way to use your computer. It is designed to put you in control and bring freedom to everybody." GNOME is for everyone, of all ages, for casual users

and software developers.

From there, I worked with the GNOME Design Team to build a usability test for these users. The test also needed to exercise the GNOME design patterns. I compared the design patterns used by each GNOME program and decided five GNOME applications provided a reasonable representation of the design patterns:

1. gedit (text editor)
2. Web (Web browser)
3. Nautilus (file manager)
4. Software (similar to an app store)
5. Notes (a simple note-taking program)

Having decided on the programs, I wove a set of test scenarios that exercised the design patterns around tasks that real people would likely do. Designing the test scenarios is an important part of any usability test. You need to be very careful in the wording. It is too easy to "give away" accidentally how to do something just by using a particular turn of phrase. For example, one of my scenarios for Web asked testers to "Please make the text bigger" on a Web site—not to "Increase the font size", which would have hinted at the menu action they would need to use to accomplish the task.

Because I work on a university campus, I invited students, faculty and staff to participate in a usability study. Volunteers were selected without preference for gender, age group or level of experience. This reflects GNOME's preference to target a broad range of users. Each tester was given a $5 gift card to the campus coffee shop as a "thank you" for participating in the usability test.

In total, 12 testers participated in the usability test, representing a mix of genders and an age range spanning 18–74. Participants self-identified their level of computer expertise on a scale from 1 to 5, where 1 indicated "No knowledge" and 5 meant "Computer expert". No testers self-identified as either 1 or 5. Instead they filled a range between "2: I know some things, but not a lot", and "4: I am better than most", with an average rating of 3.25 and a mode of "3: I am pretty average."

Before each usability test session, every participant received a brief description of the usability study, explaining that this was a usability test of the software, not of them. This introduction also encouraged

testers to communicate their thought process. If searching for a print action, for example, the participant should state "I am looking for a 'Print' button." Testers were provided a laptop running a "liveUSB" image of GNOME 3.12 containing a set of example files, including the text file used in the gedit scenario tasks. However, the USB image proved unstable for most programs in the usability test. To mitigate the stability issues, I rebooted the laptop into Fedora 20 and GNOME 3.10 to complete the scenario tasks for Web, Nautilus, Software and Notes. In testing GNOME, each participant used a separate guest account that had been pre-loaded with the same example files. These example files also included items unrelated to the usability test, similar to how real users keep a variety of documents, photos and other files on their computers, allowing participants to navigate these contents to locate files and folders required for the scenario tasks.

At the end of each usability test, I briefly interviewed the testers to probe their responses to the tasks and programs, to better understand what they were thinking during certain tasks that seemed particularly easy or difficult. Overall, the usability test included 23 scenario tasks, which most testers completed in 50–55 minutes.

My Usability Test Results

The value of a usability test is finding areas where the program is difficult to use, so developers can improve the interface in the next version. To do that, you need to identify the tasks that were difficult for most users to perform.

You easily can see the usability test results by using a heat map. This kind of visualization neatly summarizes the results of the usability test. In the heat map, each scenario task is represented in a separate row, and each block in the row represents a tester's experience with that task (Figure 1). Green blocks indicate tasks that testers completed with little or no difficulty, and yellow blocks signify tasks that presented moderate difficulty. Red boxes denote tasks where testers experienced extreme difficulty or where testers completed tasks incorrectly. Black blocks indicate tasks the tester was unable to complete, while white boxes indicate tasks omitted from the test, usually for lack of time.

The "hot spots" in the heat map show tasks that were difficult for testers.

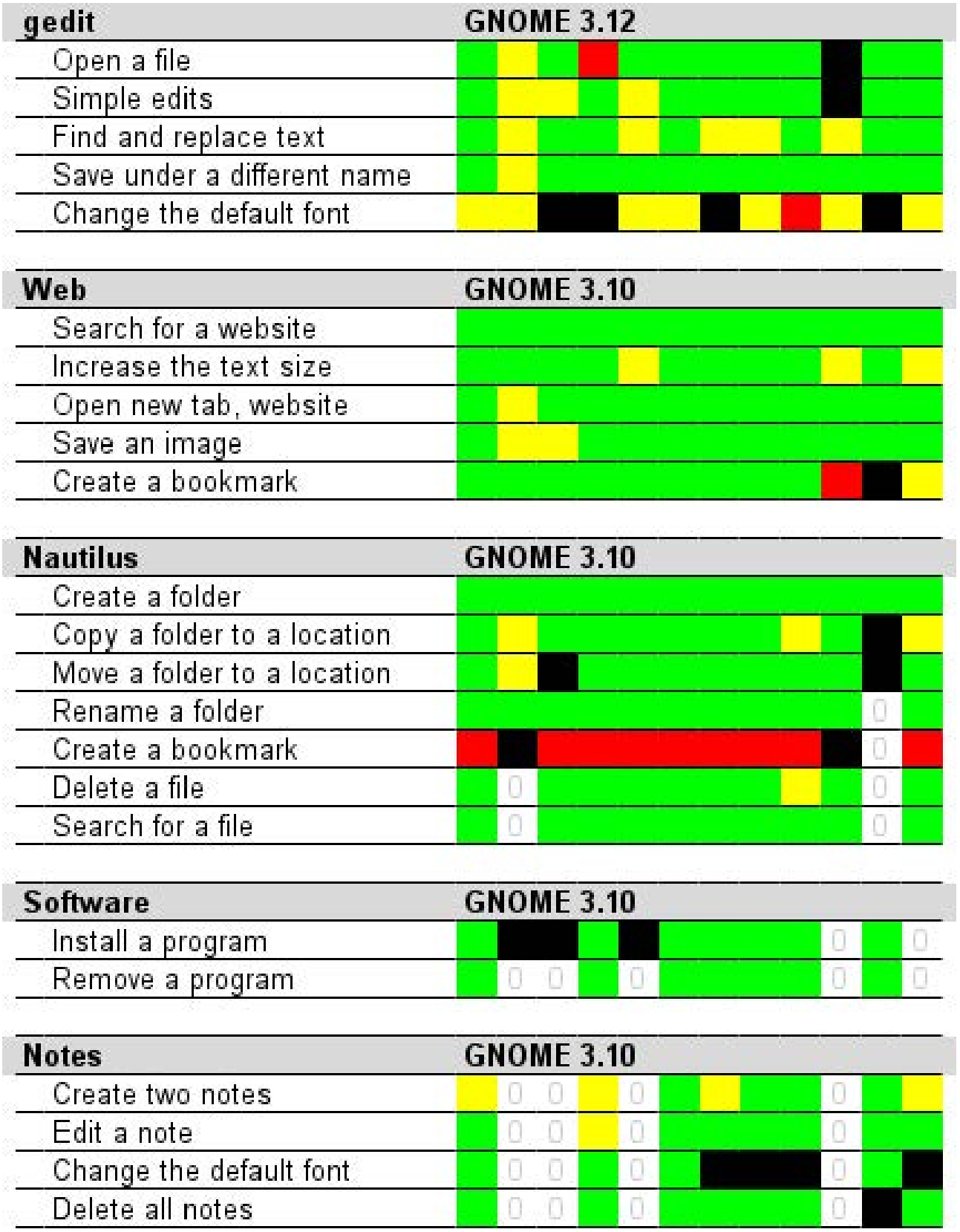


Figure 1. Usability Heat Map

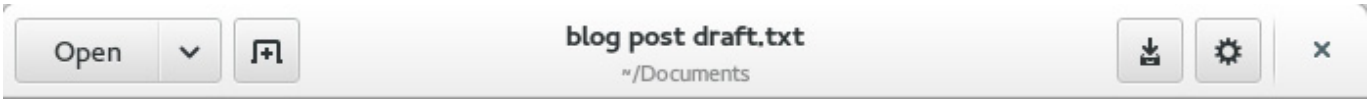


Figure 2. Header Bar for gedit, Showing the Gear Menu

1) Interestingly, all participants experienced significant issues with changing the default font in gedit (GNOME 3.12). A significant number of testers were unable to accomplish a similar task in Notes. In observing the tests, the testers typically looked for a "font" or "text" action under the gear menu. Many participants referred to the gear menu as the "options" or "settings" menu because of a previous affiliation with the gear icon and "settings" in other Mac OS X and Windows applications (Figure 2).

These participants expected that changing the font was an option in the program, and therefore searched for a "font" action under the gear or "options" menu. Part of this confusion stemmed from thinking of the text editor as though it were a word processor, such as Microsoft Word, which uses items in menus or on a toolbar to set the document font. This behavior often was exhibited by first highlighting all the text in the document before searching for a "font" action.

2) In the Nautilus file manager, testers also experienced serious difficulty in creating a bookmark to a folder. In GNOME, this task is usually achieved by clicking into the target folder then selecting "Bookmark this Location" from the gear menu, or by

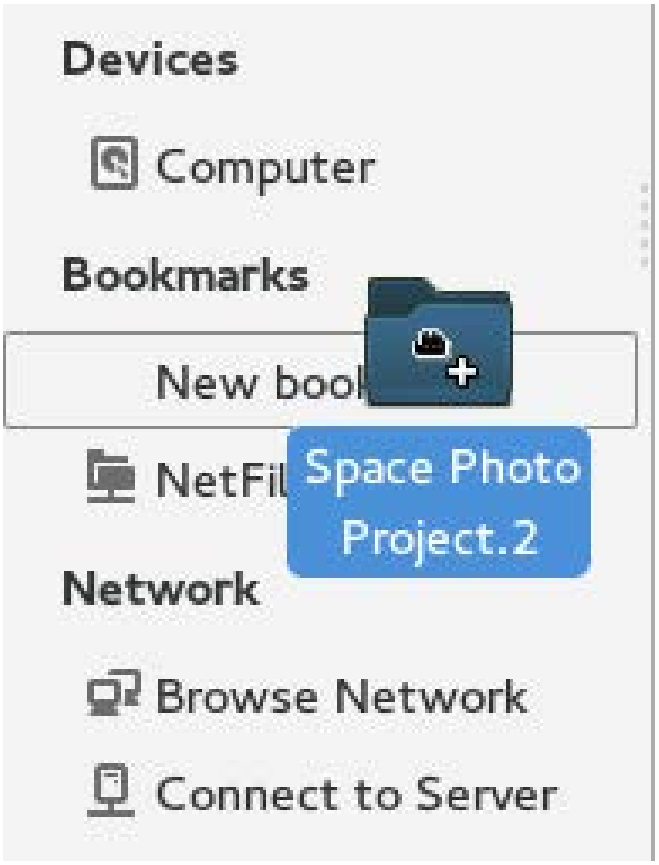


Figure 3. Dragging a Folder to Create a Bookmark in Nautilus

clicking and dragging the intended folder onto "Bookmarks" in the left pane (Figure 3).

However, testers initially addressed this task by attempting to drag the folder onto the GNOME desktop. When interviewed about this response, almost all participants indicated that they prefer to keep frequently accessed folders on the desktop for easier access. Most testers eventually moved the target folder into the Desktop folder in their Home directory, and believed they successfully completed the task even though the target folder did not appear on the desktop.

3) Testers also experienced difficulty when attempting "find and replace text" in gedit. In this task, testers employed the "Find" feature in gedit to search for text in the document. Experimenting with "Find", testers said they expected to replace at the same time they searched for text. After several failed attempts, testers usually were able to invoke the "Find and Replace" action successfully under the gear menu.

Although the overall GNOME desktop was not part of the usability test, many testers experienced difficulty with the GNOME "Activities" hot corner. In the GNOME desktop environment, the Activities menu reveals a view of currently running programs and a selection of available programs. Users can trigger the Activities menu either by clicking the "Activities" word button in the upper-left corner of the screen or by moving the mouse into that corner (the "hot corner"). Although testers generally recovered quickly from the unexpected "hot corner" action, this feature caused significant issues during the usability test.

General Issues

The open-source software usability challenge is cultural. To date, usability has been antithetical to open-source software philosophy, where most projects start by solving a problem that is interesting to the developer, and new features are incorporated based on need. Crafting new functionality takes priority, and open-source developers rarely consider how end users will access those features.

However, a key strength of open-source software development is the developer-user relationship. In open-source software projects, the user community plays a strong role in testing new releases. Unfortunately, testers cannot rely on the typical user-testing cycle to provide sufficient user interface feedback. Developers

can learn a great deal simply by observing a few users interacting with the program and making note where testers have problems. This is the essence of a usability test.

Usability tests need not be performed in a formal laboratory environment, and developers do not need to be experts in order to apply usability testing methodology to their projects. Developers need only observe users interacting with the program for usability issues to become clear. A handful of usability testers operating against a prototype provides sufficient feedback to make informed usability improvements. And with good usability, everyone wins.■

Jim Hall is an advocate for free and open-source software, best known for his work on the FreeDOS Project. At work, Jim is the IT Director and IT Executive at the University of Minnesota Morris. In May, Jim received his Master's in Scientific and Technical Communication, studying the usability of open-source software.

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

DOC SEARLS

How Can We Get Business to Care about Freedom, Openness and Interoperability?

They use our stuff. Why not our values too?

At this point in history, arguments for using Linux, FOSS (free and open-source software) and the Internet make themselves. Yet the virtues behind those things—freedom, openness, compatibility, interoperability, substitutability—still tend to be ignored by commercial builders of new stuff.

For example, US health care, like pretty much every business category, is full of Linux and FOSS, and is to some degree connected on the Net. Yet, it remains a vast feudal system of suppliers that nearly all work to lock doctors, hospitals and labs into dependency on closed, proprietary, incompatible, non-interoperable and non-substitutable systems. I've witnessed these up close as a patient. In one case, diagnostic scans by one machine and software system couldn't be read by computers with software designed to read the output of a different company's scans. In

another case, records kept by one specialty failed to inform another specialty in the same hospital. The first one gave me a case of pancreatitis, and the second one gave my mother a fatal stroke.

We are seeing the same thing start to happen already with the Internet of Things (IoT), about which Bruce Sterling (**http://en.wikipedia.org/wiki/Bruce_Sterling**) has written a brilliant essay titled *The Epic Struggle of The Internet of Things* (**http://www.amazon.com/The-Epic-Struggle-Internet-Things-ebook/dp/B00N8AIFYC**). "The first thing to understand about the 'Internet of Things'", he says, "is that it's not about Things on the Internet. It's a code term that powerful stakeholders have settled on for their own purposes. They like the slogan 'Internet of Things' because it sounds peaceable and progressive. It disguises the epic struggle over power, money and influence that is about to ensue. There is genuine Internet technology involved in the 'Internet of Things'. However, the legacy Internet of yesterday is a shrinking part of what is at stake now."

It's actually worse than that:

> An Internet of Things is not a consumer society. It's a materialized network society. It's like a Google or Facebook writ large on the landscape.
>
> Google and Facebook don't have "users" or "customers". Instead, they have participants under machine surveillance, whose activities are algorithmically combined within Big Data silos. They don't need the reader to be the hero. He's not some rational, autonomous, economic actor who decides to encourage the Internet of Things with his purchasing dollars. They're much better off when those decisions are not his to make.
>
> The reader may be allowed to choose the casing of his smartphone and the brand of his vacuum cleaner, but the digital relation between the two of them is not his decision. He still has a role of sorts, but it's much like the role he has within Google and Facebook. He gets fantastic services free of charge, and he responds mostly with dropdown menus and check boxes, while generating data whose uses and values are invisible to him.
>
> The reader didn't build the

If we're geeks like most *Linux Journal* readers, we're already doing our part, laboring away on free and open-source stuff, and using as much free and open stuff as we can.

> phone or the vacuum cleaner. He can't repair or modify them. He doesn't understand their technical workings, and when the two of them interact (by various adroit forms of wireless communication), he's not in charge of that, or of where the data goes....
>
> The reader is not a "customer" of Facebook because he never paid for Facebook. Facebook's genuine customers are the marketers—those who pay Facebook for the hard labour of surveilling the billion people on Facebook. Facebook is one of the "Big Five" of Facebook, Amazon, Google, Microsoft and Apple....
>
> The Big Five are the genuine heroes of the Internet of Things. The epic drama of the Internet of Things is really their story. It's not a popular uprising—except in the sense that the Big Five are really, really, really "popular"—because billions of people are willingly involved in their systems. The Internet of Things is basically a recognition by other power-players that the methods of the Big Five have won, and that they should be emulated.
>
> The Big Five are smart, profitable, capable and colossal. They are as entirely free of political constraint as the railroads or Standard Oil were in their own heyday....

Phil Windley calls this The Compuserve of Things (**http://www.windley.com/archives/2014/04/the_compuserve_of_things.shtml**), and says:

> On the Net today we face a choice between freedom and captivity, independence and dependence. How we build the Internet of Things has far-reaching consequences for the humans who will use—or be used by—it. Will we push forward, connecting things using forests of silos that are

> reminiscent the online services of the 1980s, or will we learn the lessons of the Internet and build a true Internet of Things?

Well, it depends on who "we" are. If we're geeks like most *Linux Journal* readers, we're already doing our part, laboring away on free and open-source stuff, and using as much free and open stuff as we can. But we're not the ones running the companies building closed and silo'd systems, including proprietary networks of things. And those folks aren't getting the lessons we've been teaching for decades. Why is that?

One dividing line is between standards and platforms built on them. This is also the line between infrastructure and commerce in the "layers of time"

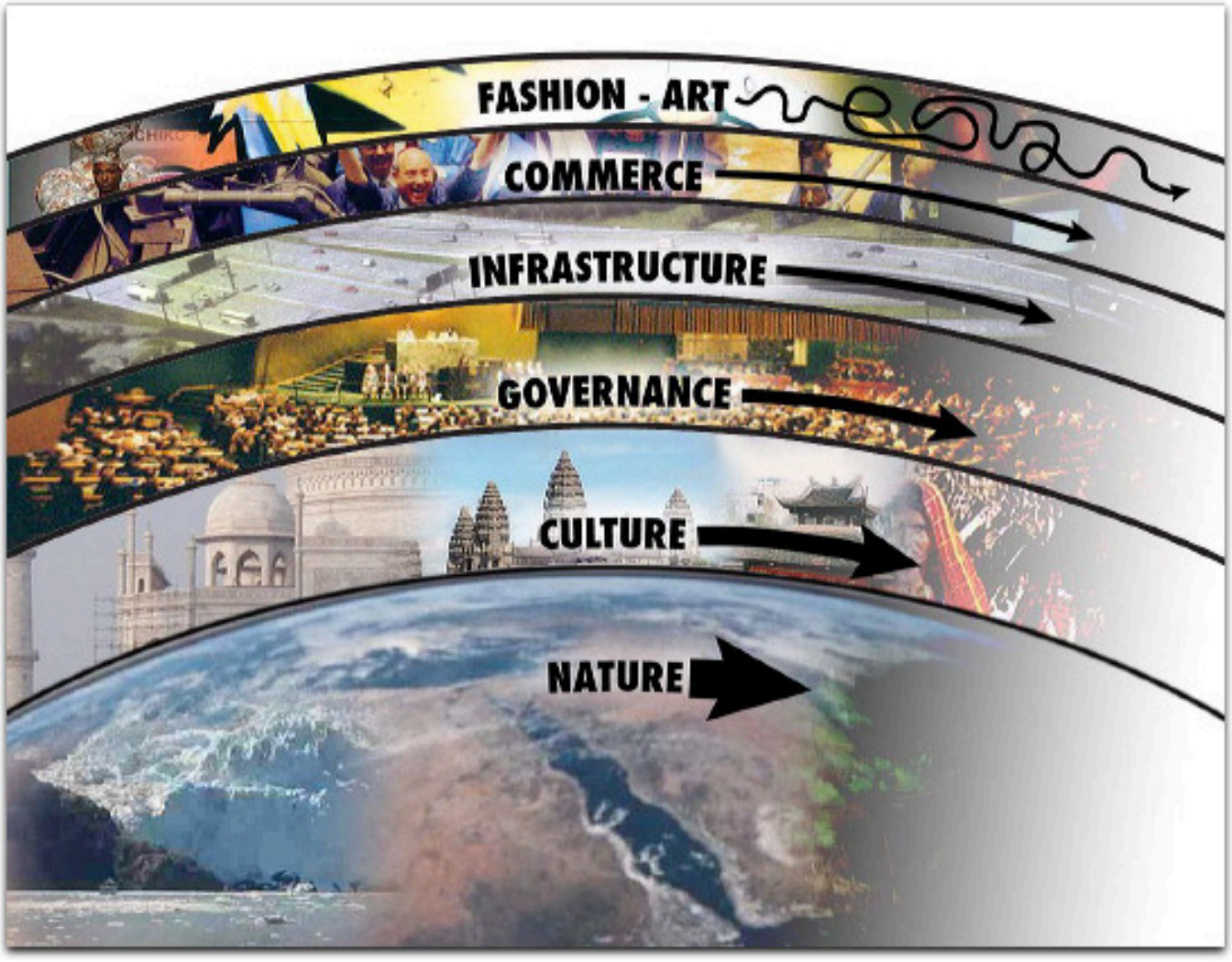


Figure 1. Layers of Time (from The Long Now Foundation)

Geeks working down in the lower geology layers tend to be as oblivious to what happens at the Commerce and Fashion layers as the core of the Earth is to civilization.

graphic from The Long Now Foundation (**http://longnow.org**)—see Figure 1.

FOSS building materials are all at the Infrastructure layer. So are the standards that create the Net and the Web: TCP/IP, HTTP and the rest. These and countless thousands (millions?) of standards and code bases support boundless freedom and generativity for everything that's built on them and with them, up at the Commerce and Fashion-Art levels.

But all the Big Five, while founded on those standards, and packed with FOSS code, need to compete at the commerce level. This tends to subordinate freedom, interop, generativity and the rest—except within their own feudal empires. There they can do a pretty good job.

For example, most of the big platforms come with SDKs (software development kits), APIs (application programming interfaces) and other means for encouraging, producing and supporting lots of code—and dependencies. This is why there are more than a billion apps each on iOS and Android.

But platforms can also change fast (**http://scripting.com/2014/07/07/comparingApis.html**) and put dependent developers and users at their mercy, as many discovered when Twitter pulled the rug out from under them in 2012 by changing its API (**http://www.techradar.com/us/news/world-of-tech/twitter-to-developers-screw-you-guys-1092570**). This was a huge interop fail for many code bases, some of which died.

Infrastructural standards and code bases (such as Linux and Apache) come up from lowest level—Nature—and bubble up through their cultures and governance norms. None of those norms arise from government policies, in spite of what the Governance layer suggests in that graphic. As David Clark (**http://en.wikipedia.org/wiki/David_D._Clark**) famously put it to (and for) the IETF in 1992, "We reject: kings, presidents and voting. We believe in: rough consensus and

running code" (Lessig, *Code 2.0*, p. 4: **http://www.codev2.cc/download+remix/Lessig-Codev2.pdf**).

Geeks working down in the lower geology layers tend to be as oblivious to what happens at the Commerce and Fashion layers as the core of the Earth is to civilization. Some of us have heard Linus say "I do kernel space" (**http://www.linfo.org/kernel_space.html**). "I don't do user space" (**http://www.linfo.org/user_space.html**). That's because Linux developers make a sharp distinction between those two spaces, by name. The deep nature of Linux is in its kernel, whose job (as working infrastructure) is to support everything above it without prejudice (**http://www.linuxjournal.com/article/8664**). In other words, it tries to be as neutral as possible.

But up at the Commerce and Fashion-Art levels, where most people acquire goods and live with them, few are concerned that a billion+ iOS apps run only on Apple hardware, or that a billion+ Android apps run on hardware from makers required to privilege Google's apps first. Or that the walls of those feudal empires block views toward negative externalities, one of which is restricted (or absent) inter-empire interoperability. As a result we have a marketplace where the benefits of freedom and openness—taught by the Net, the Web, open standards, open code and open hardware—aren't seen, because their first causes are out of sight: buried down at the infrastructure layer and below.

But some geeks aren't forgetting. Cory Doctorow (**http://craphound.com**), for example, warns of a coming "civil war" (**http://boingboing.net/2012/08/23/civilwar.html**) between general-purpose and special-purpose computing. The white-box PC, for example, is general-purpose. Meanwhile, we still have no equivalent with smartphones and tablets. Instead we have closed and locked things that foreclose on

The Net and the Web were both built on software and hardware that leveraged standard, open and general-purpose equipment, protocols and code bases.

many freedoms.

The Net and the Web were both built on software and hardware that leveraged standard, open and general-purpose equipment, protocols and code bases. Smartphones and tablets take advantage of those things, but limit what people can do with them. Apps are available only in stores, which are closed and private. Apps themselves also tend to be silo'd. They may operate on your phone, but don't easily interoperate with each other. Try gathering data from all your fitness and health apps into a place of your own. Even if you can, such as with DigiFit (**http://www.digifit.com**), it's in yet another company's silo.

Research on the questions raised by these issues is remarkably thin. Toward relieving that, Urs Gasser (**http://cyber.law.harvard.edu/people/ugasser**) and John Palfrey (**http://cyber.law.harvard.edu/people/jpalfrey**), colleagues of mine at the Berkman Center (**http://cyber.law.harvard.edu**), published *Interop: The Promise and Perils of Highly Interconnected Systems*, in 2012 (**http://www.amazon.com/Interop-Promise-Perils-Interconnected-Systems/dp/0465021972**). In it they argue for "a nuanced and stable theory of interoperability" that can resolve issues at the four layers at which they see interoperability playing: technological, data, human and institutional.

I've been watching problems at all four layers for the past eight years through ProjectVRM (**http://cyber.law.harvard.edu/projectvrm/Main_Page**), which seeks to provide individuals with tools for both independence and engagement in the commercial world. All the projects and companies on our developers list (**http://cyber.law.harvard.edu/projectvrm/VRM_Development_Work**) do their best to reconcile the need for interop and substitutability with commercial imperatives: attracting investment and partners, competing and so on.

Some efforts are formal and involve a number of players that compete in a cooperative framework. The Respect Network (**https://www.respectnetwork.com**), for example, includes dozens of partner organizations committed to the Respect Trust Framework (**http://openidentityexchange.org/trust-frameworks/respect-trust-framework**), which assures both interoperability and substitutability between service providers.

But it's an uphill battle, and has

been for a long time.

As I see it (and I've been looking at this for *Linux Journal* since the mid-1990s), the biggest conflict for Interop is freedom vs. captivity. Developers that value freedom tend to maximize interop, while developers that value captivity tend to minimize interop. And most of what we get are compromises.

I believe the main problem is simply ignorance of consequences, which is an old problem with business. But I also believe in research. Urs recently wrote to one of the Berkman lists, asking for "the most challenging, fascinating, puzzling...interop stories, questions, problems, opportunities...that you've been thinking about recently"—toward fresh research on interop.

Let's give him some of those. Comment below (when this goes up on the Web) or write me (doc at linuxjournal dot com).■

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.

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

Advertiser Index

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