LINUX JOURNAL

Since 1994: The Original Magazine of the Linux Community

MARCH 2014 | ISSUE 239 | www.linuxjournal.com

20

CELEBRATING 20 YEARS

WRITER FROM *LJ*'S FIRST ISSUE LOOKS BACK AND AHEAD

DIY TWO-FACTOR AUTHENTICATION FOR APACHE AND SSH

Use the Tails Live Distribution for Extra Privacy

A Look at the Role of UEFI Secure Boot

HOW-TO: Give Irssi a GUI

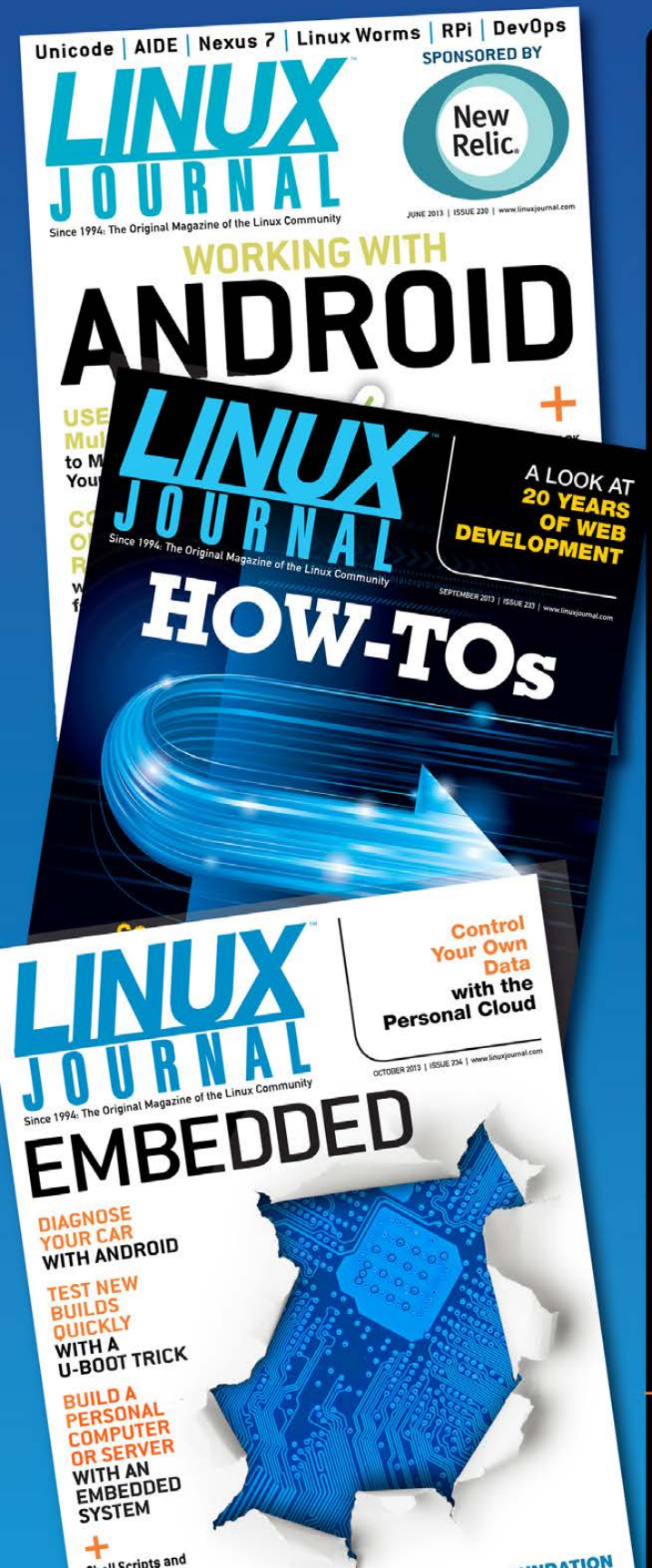
Conquer Dependency Hell with Docker

EOF

Doc Presents Readers an Assignment for the Future

WATCH: ISSUE OVERVIEW

If You Use Linux, You Should Be Reading LINUX JOURNAL™



- In-depth information providing a full 360-degree look at featured topics relating to Linux
- Tools, tips and tricks you will use today as well as relevant information for the future
- Advice and inspiration for getting the most out of your Linux system
- Instructional how-tos will save you time and money

Subscribe now for instant access! For only $29.50 per year—less than $2.50 per issue—you'll have access to *Linux Journal* each month as a PDF, in ePub format, in Mobi format, on-line and through our Android and iOS apps. Wherever you go, *Linux Journal* goes with you.

SUBSCRIBE NOW AT:

WWW.LINUXJOURNAL.COM/SUBSCRIBE

SILICON MECHANICS

SERVERS • STORAGE • HPC

Four-Layer Recipe for Success

1. Consult
   - Discussing Needs
   - Determining Solutions
   - Driving Value
2. Architect
   - Servers & Storage
   - Cloud & Virtualization
   - HPC Clusters
3. Build
   - Custom Configurations
   - Skilled Technicians
   - Thorough Testing
4. Service & Support
   - Onsite & Remote Integration
   - Guaranteed 3+ Year Warranty
   - Diagnostics & Replacement

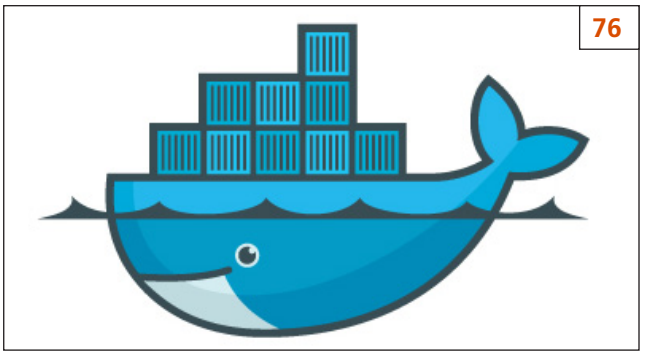
CONTENTS

MARCH 2014
ISSUE 239

FEATURE

INDEPTH

IN EVERY ISSUE



COLUMNS

KNOWLEDGE HUB

ON THE COVER

- **Writer from *LJ*'s First Issue Looks Back and Ahead, p. 62**
- **DIY Two-Factor Authentication for Apache and SSH, p. 100**
- **Use the Tails Live Distribution for Extra Privacy, p. 44**
- **A Look at the Role of UEFI Secure Boot, p. 92**
- **How-To: Give Irssi a GUI, p. 52**
- **Conquer Dependency Hell with Docker, p. 76**
- **EOF: Doc Presents Readers and Assignment for the Future, p. 108**

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

Publisher	Carlie Fairchild publisher@linuxjournal.com
Director of Sales	John Grogan john@linuxjournal.com
Associate Publisher	Mark Irgang mark@linuxjournal.com
Webmistress	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

Become a Big Data Master!

Over 45 HOW-TO, practical classes and tutorials to choose from!

Attend The 3rd Big Data TechCon!

The HOW-TO technical conference for professionals implementing Big Data



Come to Big Data TechCon to learn the best ways to:

- Process and analyze the real-time data pouring into your organization.
- Learn HOW TO integrate data collection technologies with data analytics and predictive analysis tools to produce the kind of workable information and reports your organization needs.
- Understand HOW TO leverage Big Data to help your organization today.
- Master Big Data tools and technologies like Hadoop, MapReduce, HBase, Cassandra, NoSQL databases, and more!
- Looking for Hadoop training? We have several Hadoop tutorials and dozens of Hadoop classes to get you started — or advanced classes to take you to the next level!

BigData TECHCON Boston

March 31-April 2, 2014

www.BigDataTechCon.com

A BZ Media Event Big Data TechCon

SHAWN POWERS

7,305 Days!

Personally, I thought it was strange for everyone to make a big deal about such an arbitrary number of days. Then I was told it was the years that were of particular note (20 of them, to be exact), and suddenly thought it seemed insignificant. I mean, 7,305 is a *much* bigger number! Then people rolled their eyes and left the room.

We're excited for our 20th year! And to celebrate, we figured we'd give you a month of Linux-related information, up-to-date news, fun articles and boat loads of tips. It's been our thing for 20 years, and it seemed like a silly time to stop! Reuven M. Lerner kicks off this System Administrator-focused issue with togetherness. Specifically, togetherness supported by TogetherJS. With TogetherJS, you can add real-time collaboration to your Web apps. If you need to write apps that allow remote individuals to collaborate on particular projects, this article is for you.

Dave Taylor steps back into the gaming world as he starts us off on a quest for *Zombie Dice*. I don't think we'll actually script brain-munching game pieces, but Dave proves that story problems really were important in school. Let's dive in and figure out the game as Dave describes how to create it. Kyle Rankin has a game of his own this month, and that game is security. Okay, maybe it's not a game, but it was a good segue, so I'm going to keep it. Kyle demonstrates Tails, which is an entire Linux distribution designed to route all traffic through the TOR network. It does even more, but I'll let Kyle explain the rest.

I decided to open up my personal laptop a bit and explain how I use a GUI notification system on my remote Irssi IRC program. Using Irssi in a screen session is such an incredible way to chat that I'm unwilling to move to another client. Unfortunately, I can't always see my terminal window when working, so

VIDEO:
Shawn Powers runs through the latest issue.

I miss important notifications. This month, I show you my solution. Hint: it's nerdy.

Then we have Bernie Thompson back to celebrate our 20 years of Linux. You may remember Bernie wrote in the very first issue of *Linux Journal*, comparing Linux to Windows and OS/2. In this issue, he looks at where things have gone during the past two decades, and where things are going in the future. Linux was cutting edge 20 years ago, and today? Still on the forefront of technology.

Virtualization has changed the way we think of computers. As with any incredible idea, it has evolved and even sparked new technologies like LXC, or Linux Containers. Dirk Merkel shows us Docker this month. If you need lightweight Linux containers, and want them to be consistent and easy to deploy, you'll want to read this article. Dirk not only shows us the why, but also the how.

Every system administrator needs to be familiar with the latest security-related features of Linux and the hardware it supports. Mark Doran discusses UEFI Secure Boot this month, as it's becoming more and more prevalent and important in our corner of the tech world. Whether you prefer simply to disable Secure Boot or want to leverage a distribution that supports it, Mark's article will arm you with knowledge. And like G.I. Joe told us all those years ago, knowing is half the battle!

Last, but certainly not least, James Litton shows how to implement two-factor authentication on Web sites and SSH servers. If you thought implementing two-factor auth was too complicated for your own purposes, think again. James demonstrates that with a little bit of scripting, it's as easy as 1-2-3. (But please don't use "123" as your password, even with two-factor auth.)

Like every other issue during the past 20 years, this one is full of tech tips, product announcements and thought-provoking content. I've only been part of the staff here at *Linux Journal* for a little less than half the past 20 years, but I've been a reader for almost the entire 20. As a community, we've grown more and more passionate through the past two decades, so I very much look forward to the next 20 years—or 10,519,200 minutes, whichever you prefer.■

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

SSDs, I

Regarding Brian Trapp's interesting article on SSDs ["Solid-State Drives—Get One Already" in the January 2014 issue], there's one thing I would like to know. Mr Trapp should have been more elaborate on this: how does he reconcile his remark on TRIM and RAID setups with this: **http://serverfault.com/questions/508459/implementing-linux-fstrim-on-ssd-with-software-md-raid?**?

I have a RAID 1 setup running on an SSD, which works perfectly well with fstrim.

—jouthuis

Brian Trapp replies: *Thanks for your note. My assertion that SSDs, TRIM and RAID are a bad combination was indeed incomplete. Never bet on Linux not having a feature for long!*

If you're using hardware-based RAID (using the RAID feature built in to your motherboard's BIOS, a standalone RAID card or a RAID on a NAS device), TRIM support still is probably not going to work. Intel seems to be working hard on the RST chipsets, but check the notes for your specific driver carefully.

*On the other hand, if you're using software-based RAID, the picture is much prettier! According to the 3.7 Linux kernel notes (***http://kernelnewbies.org/Linux_3.7***, December 2012), TRIM was added to mdadm for modes 0, 1 and 10!*

Thanks for the pointer. I'm going to try a RAID 1 on my workstation the first chance I get!

SSDs, II

I did a lot of research on SSDs and found "The SSD Endurance Experiment" by Tech Report, Geoff Gasior. There are several parts to the report with the last one on

300TB testing on six SSDs. The results are impressive.
—Roman

Brian Trapp replies: *Thanks for the link Roman! While the 200TB update came out after I had already written the article, it is a good resource for data on how the extreme upper end of write loads affect SSD performance and endurance.*

Usability, I

The article "It's about the User: Applying Usability in Open-Source Software" by Jim Hall in the December 2013 issue was fantastic! I work for the US government, and usability is a significant issue that has plagued many of our software projects.

The GNOME Human Interface Guide (HIG) provides on-target usability responsibilities, and I even used it as the foundation for a government software HIG several years ago. While the GNOME HIG is great, it lacks many of the key points made by Jim. Maybe GNOME 3 would have ranked higher in the 2013 Best Desktop Environment category if the HIG had a stronger emphasis on user testing.

Speaking of user testing, I can't figure out one element of Jim's Gedit test—that is, how do you change the default font in Gedit?
—Peter Cook

Jim Hall replies: *Thanks for your comments! Your question about how to change the default font highlights a usability issue in Gedit. From my usability test, all*

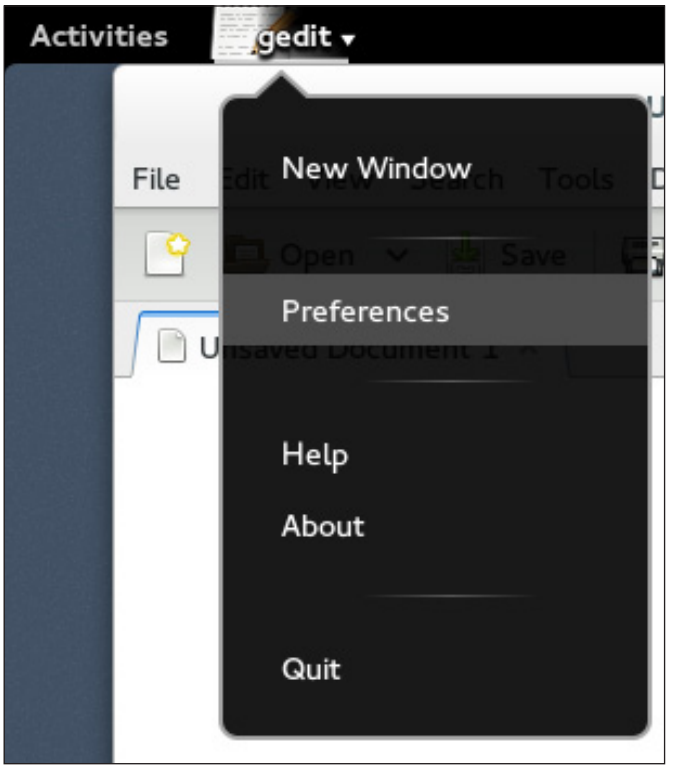


Figure 1. Gedit Preferences

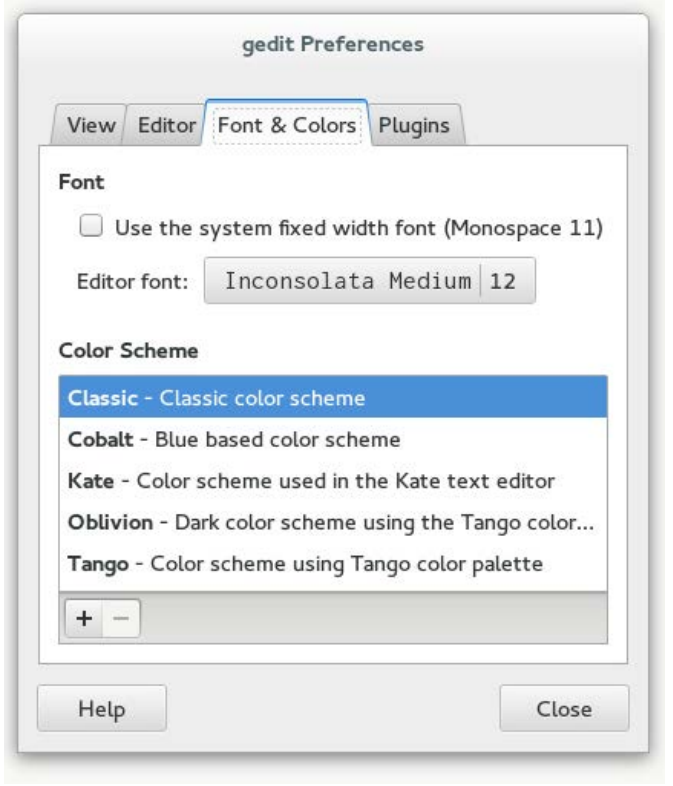


Figure 2. Font Tab

participants either were unable to change the default font in Gedit, or they experienced extreme difficulty in doing so.

I conducted my usability test with Fedora 17 and GNOME 3.4, and in that version of Gedit, there was a Preferences menu item under the Edit menu. In the current version of GNOME (3.10), the Preferences menu item is located under the gedit application menu in the top bar. From the Font & Colors tab, you can set the default font for Gedit. See Figures 1 and 2 for screenshots using Fedora 20 and GNOME 3.10.

Usability, II

Jim Hall's article on usability testing in the December 2013 issue was excellent! I have done some of the things he mentions, and one thing in particular is really important: when you have a user struggling during the testing, don't jump in to "fix" it. The problem is not the user; the problem is the program, particularly when a number of people face the same issues.

Another example I found that agreed with my experience is regarding the use of "hot corners". I have always found that the biggest annoyance, and I cannot imagine it got in with

the help of any usability testing. As an experienced user, I know I can find the place to turn it off (and I always do), but it exemplifies to me the problem with software designed by developers for developers. Open source has a culture that puts the developer at the top of the pyramid, with users at the bottom, and hot corners is a result. We are at the point where often the open-source alternative is every bit as functional as the proprietary software, but cannot get traction because of usability issues. This is the next frontier for open source if we really want to take over. And since Microsoft seems to have committed a huge usability blunder with Windows 8, we have a shot, but only if we can provide an alternative that is easier for people to use.

—Kevin O'Brien

Jim Hall responds: *While GNOME was not a focus of my usability test, almost every participant experienced problems with the GNOME hot corner. This usually occurred right away in the first task, when testers first tried to use the menus. They would often "overshoot" the menu and activate the hot corner instead. Although testers were able to recover from this pretty quickly, it definitely caused unexpected problems during the test. During the usability tests, I kept a running log of comments made and actions performed by the participants. When they accidentally activated the GNOME hot corner, participants were immediately confused, asking "What just happened?" or exclaiming "Whoa!" or "Oops!" One user became so frustrated, he started swearing at the computer whenever he hit the hot corner.*

I'd like to make 2014 the year of usability in open-source software. Even proprietary software has usability issues (the dramatic change to the Windows 8 desktop is one obvious example), but I want to focus on open-source software. I'm glad you share this enthusiasm! Please bring this energy to your favorite open-source software project to help find and fix usability issues. This will help everyone who uses open-source software, and together we can make Linux even better!

Automating Passphrases in Encrypted Volumes

Thanks to Shawn Powers for his "Encrypting Your Cat Photos" article in the January 2014 issue. I'd like to share a solution to a problem I face while working with encrypted volumes in the cloud. The problem is how to automate entering passphrases when accessing encrypted volumes; in my case, access is

done from a script, and users cannot be prompted.

I'm using this command to solve the problem:

```
cryptsetup --key-file $pass_file luksOpen
 ➥$vol_name my_volume
```

Before calling it, I copy `$pass_file`, which contains the passphrase. Then I delete the file.

This is not very secure method. I wonder if there is a better way to do it.
—Evgeni Stavinov

That is, unfortunately, an issue with encrypting a filesystem. The only solution I know for non-interactive decryption is what you are doing, which is to have a password file. Usually, that's a file stored with protected permissions somewhere on a system, but of course, you see the problem, the password is in a file! I'm not sure how you're copying the password file, but if it were me, I'd temporarily copy the file (with secure permissions of course) to a ramdisk so the file isn't recoverable from the hard disk. There's no great answer I'm aware of, however.—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!

FLOURISH! 2014

OPEN SOURCE, OPEN FUTURE

LINUX JOURNAL

SPANTREE technology group

UIC University of Illinois at Chicago Department of Computer Science College of Engineering

diff -u
WHAT'S NEW IN KERNEL DEVELOPMENT

Linus Torvalds is planning to put out **Linux version 4.0** in a year or so, and he mentioned it by way of explaining why he wanted to do it and what he hoped would happen when he did.

Apparently, his main reason to go from 3.x to 4.x is to avoid the 3.x minor numbers going up above the teens. He said, "I'm ok with 3.<low teens>, but I don't want us to get to the kinds of crazy numbers we had in the 2.x series." That's no joke. The 2.1.y series got up to 2.1.132, and there were complaints.

So, why announce it now, if 4.0 is a year or two away? Linus said he hoped the 4.0 kernel could be a "stabilization" release, where everyone contributed bug-fixes only. By announcing it early, he hoped everyone gradually would get used to the idea, and that perhaps there would be at least a very slim chance of success. He said, "I may be pessimistic, but I'd expect many developers would go 'Let's hunt bugs....Wait. Oooh, shiny' and go off doing some new feature after all instead. Or just take that release off. But I do wonder...maybe it would be possible."

Clearly his thinking has come a long way from the old 2.x days, when long periods of stabilization were official policy—and the source of much developer angst.

Even now, most responses were opposed to making 4.0 a "stabilization" release. But, that could have less to do with a general dislike of bug-fixing and more with the fact that a large culture of stabilization has emerged in the wake of the 2.6 tree's abandonment of stabilization efforts nearly altogether.

Originally Linus was supposed to hand off the 2.6 tree to **Andrew Morton**, who would maintain it as a stable tree while Linus forked off a 2.7 development series. Instead, Linus stayed in control of 2.6, never forked a development tree, and just started taking development patches in 2.6 instead. Thus endeth the x.even/x.odd development paradigm for stable and development kernels.

In place of this, in those long-ago days, Linus said that really the Linux distribution folks were the natural ones

to handle stabilization. They typically heavily patched the Linux kernel anyway, so they had a process in place and the motivation to keep their users from hitting bugs.

But that didn't really seem like enough, so Andrew converted his beloved and popular **-mm tree** into what is now called the **linux-next** tree. Before the switch, the -mm tree was simply the collection of patches Andrew intended to feed up to Linus. But he was so prolific, and collected so many patches from other developers, that the -mm tree already had started to become a natural staging area for many patches that were on their way to the official tree. And, a lot of people ran the -mm kernel on their home systems because they liked it better. Renaming it linux-next simply formalized that identity and gave code a place to live, and more important, it allowed it to be tested by a large number of users before heading up to Linus.

Greg Kroah-Hartman and some others also started their own stabilization effort, in which they would fork off each official release and put out a few more official versions of that release consisting of bug-fixes

only. For example, after Linus put out 2.6.39, Greg and his fellows put out 2.6.39.1 through 2.6.39.4, all with stabilization fixes and security patches. When Greg started this work, Linus' nickname for it was the "sucker's tree" because of the unfathomable amount of work required.

Both Andrew's linux-next and Greg's stable tree persisted into the 3.0 time frame and continue to be a primary means of stabilizing the official kernel tree. Any patches going into Greg's stable tree, for example, must be accepted into Linus' tree first. That may sound odd, but it actually prevents the same bug from cropping up again and again in new official releases. Even with the patches going into Linus' tree, Greg's stable series is still valuable, because it has the fixes, the whole fixes and nothing but the fixes, which means Greg's tree becomes more and more stable while Linus' continues to gallivant off in development-land.

Greg also occasionally designates one of his forks as "long-term stable", which means he intends to put out more than just a few stabilization releases, but will continue to maintain it for maybe a year or longer. The choice of which official release will become "long-term stable" is a source of much ongoing speculation, because distributions like to organize their ongoing work around a particular kernel as early as possible—in fact, earlier than Greg typically picks out a new "long-term stable" candidate.

But, that is another story. The point is, stabilization efforts have risen up to fill the gaps left by Linus' decision to abandon stabilization efforts in the main kernel tree. And as a result, his concept of 4.0 as a "stabilization" release doesn't seem to strike anyone as very important.

Actually, Linus didn't completely abandon stabilization as a goal. Each release cycle includes a final release-candidate process that focuses on stabilization for a couple weeks before the official release comes out.

But, this brief stabilization window was exactly why certain people didn't like his 4.0 stabilization idea. **Olof Johansson**, for example, said that instead of dedicating a whole release to stabilization, why not just extend the stabilization window of the previous release?

There were other objections. **Ingo Molnar** and Greg both felt that the release before 4.0, "would also probably be a super-stressful release to maintainers, as everyone would try to cram their feature in there". And Greg remarked, "I see it today when people are trying to guess as to what the next

long-term-stable kernel is going to be, and they throw things in that are half-baked just because they know they can 'fix it up' later."

Ingo also said that a better way to make the official kernel more stable would be for maintainers to put more pressure on developers to submit fixes to known bugs before sending in patches implementing new features.

Ingo pointed out that Linus rarely got patches directly from the developer who created them, and that these patches typically went through at least a maintainer and possibly another person above them, before reaching Linus. So Ingo didn't feel that Linus could have much direct influence over the stabilization process anyway. He concluded, "why not just do what worked so well for v3.0 and afterward? Keep the existing process in place."

And Greg pointed out that his stabilization forks were actually very successful. He pointed out that 4,000 bug-fixes went into the 3.0.x stable releases, which he felt indicated that developers in general already took stabilization very seriously. He suggested that trying to focus on stabilization for a single 4.0 kernel release would not be as useful as evangelizing good practices for submitting code to Greg's stable trees.

The discussion continued for a bit, but for the moment, Linus had nothing to add. It seems as though he does want to improve the stabilization process for the official kernel in some way. Maybe he'll do it with a push for stability in 4.0, and maybe he'll try something else that runs more along the lines of supporting existing stability efforts.—**ZACK BROWN**

They Said It

We would spend many hours taking turns trying to rescue the princess and overthrow Bowser. We bonded over 8-bit technology and room temperature fizzy drink.
—B. T. Hogan

Nothing of me is original. I am the combined effort of everybody I've ever known.
—Chuck Palahniuk

Wanting to be someone you're not is a waste of the person you are.
—Kurt Cobain

No one can be right all of the time, but it helps to be right most of the time.
—Robert Half

[Television is] the triumph of machine over people.
—Fred Allen

Non-Linux FOSS: Wampserver

I'll be honest, the first thing that drew me to Wampserver was the name. Although it's just a continuation of the LAMP acronym for Linux, Apache, MySQL and PHP, there's something about that name that makes me smile. Anyway, Wampserver is an application for Windows that provides Apache, PHP and MySQL in a single manageable package.

Wampserver includes a single installer for all the components and an interface for controlling them (Figure 1). Although it certainly makes sense to use Linux for hosting an Apache-based Web server, if you're in a situation that requires you use Windows, Wampserver is a perfect way to install what is normally a fairly complex set of applications.

Wampserver installs all open-source software on your Windows machine, and is itself open-source software as well. It's available from SourceForge or directly from the Wampserver Web site: www.wampserver.com.—**SHAWN POWERS**

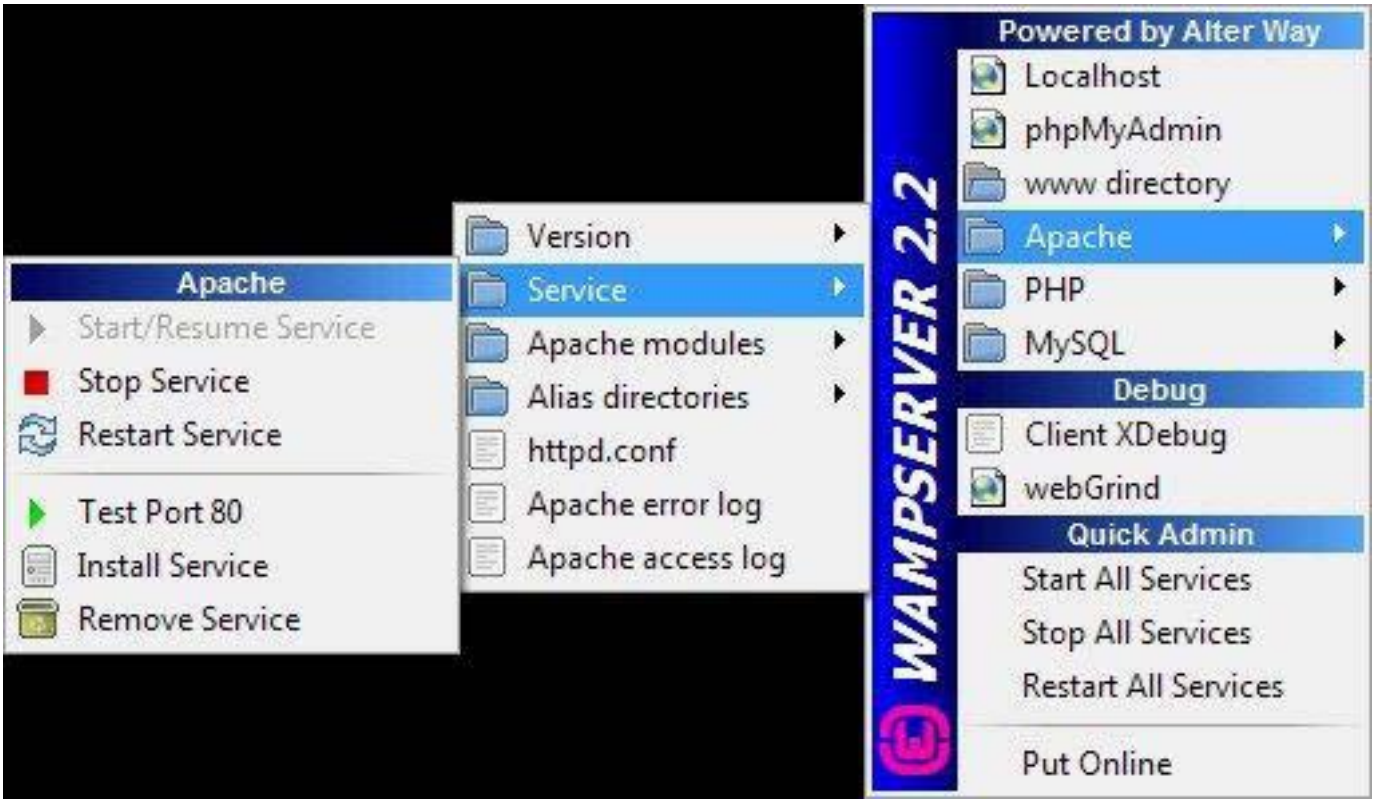
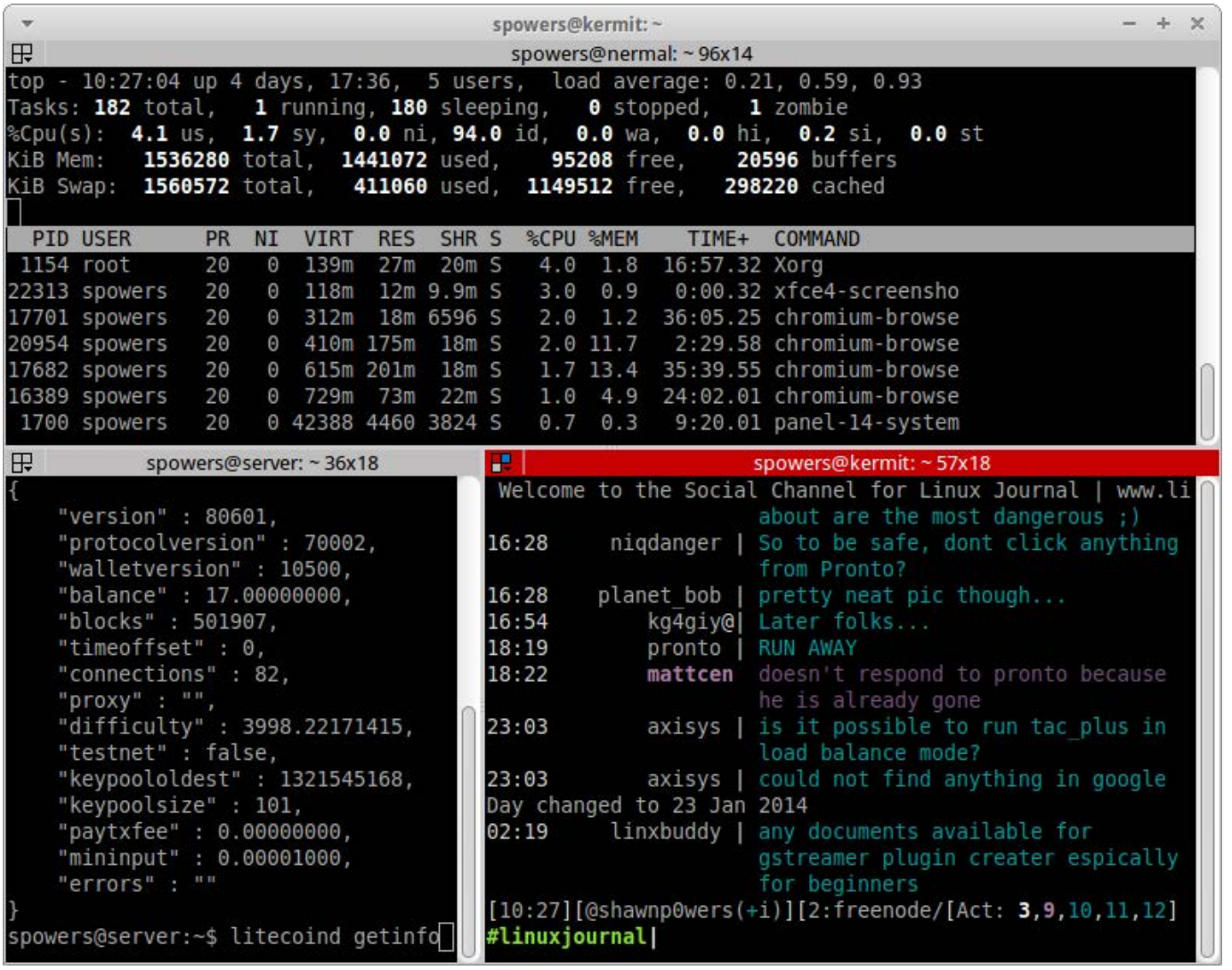


Figure 1. Wampserver Interface



Terminator

Kyle Rankin demonstrated the advantage of splitting terminal windows in previous issues. If you'd like a simple, graphical way to split your terminal window, look no further than Terminator (**http://gnometerminator.blogspot.com**). Split screens and keybindings can be saved for quick setups, and it's simple to create a grid of terminal windows or a combination of large and small screens.

If you've ever opened multiple terminal windows and tried to arrange them so you could see them all, Terminator will make your life a lot easier. Even if you've never tried a split-window setup, the ease and simplicity Terminator offers might make you a believer. It did for me!

—SHAWN POWERS

Android Candy: Humans, Run!

Whether you're a fan of the shambling brain-munchers or you prefer your undead to sprint from victim to victim, zombies are amazingly popular. In an ironic twist, the most unhealthy members of humanity, or former members, can help you become the healthiest!

"Zombies, Run!" is an application for your Android phone that uses GPS and your music collection to help motivate you to exercise. A slim waist and a great beach body might be enough to get you running, but how about the sounds of an approaching zombie horde? It's a twisted motivator, but it really works!

The application isn't free, but at $7.99 (or less if you find it on sale), it costs about as much as an extra-large frozen mocha latte. Both investments will affect your health, but in this case, the zombies are far better for you. Check it out at **https://www.zombiesrungame.com**.

—SHAWN POWERS

Spelunking with Linux

Mapping is ordinarily a complex task—just consider all the available software to help maintain and generate maps. And, that's only for mapping in two dimensions. Maintaining and generating maps of caves bumps this complexity up a notch. Caves exist in all three dimensions. So any mapping software needs to keep track of volumes rather than areas. There are not very many options that can handle this task. Luckily, one available package can do this job very well: Therion (**http://therion.speleo.sk/index.php**). Therion is an open-source program, released under the GPL and available for Windows, Mac OS X and Linux.

Most distributions should have a package available, or you always can download the source and build it from scratch. In Debian-based distributions, you can grab it with the command:

```
sudo apt-get install therion
```

Once it is installed, you can find it in your desktop environment's menu system, or run `xtherion` from a terminal window.

When you first start up Therion, you essentially get a blank canvas. You can open an existing file or create a new one to begin working. Let's start by creating a new project. You can click the File→New menu entry or press Alt-N. Therion then asks you to enter a name for your new project. This name should end with .th.

You end up with two black panes as the main part of the interface. On the far right, there are a number of sub-panes, with the settings pane opened. Here you can set the working directory for your current project and the related configuration file. You also can set command-line options, and there is a compile button, which I'll discuss shortly.

The main interface is actually composed of three separate windows. You start in the first one, the "Text Editor". There is also a "Map Editor" window and a "Compiler" window. They are accessible two ways. The first is by using the menu. The window entries are located under the main menu entry Window. There also are keyboard shortcuts for each of them. You can bring up the text editor with F1, the map editor with F2 and the compiler with F3.

Let's start with the text editor. The first step is to define the survey

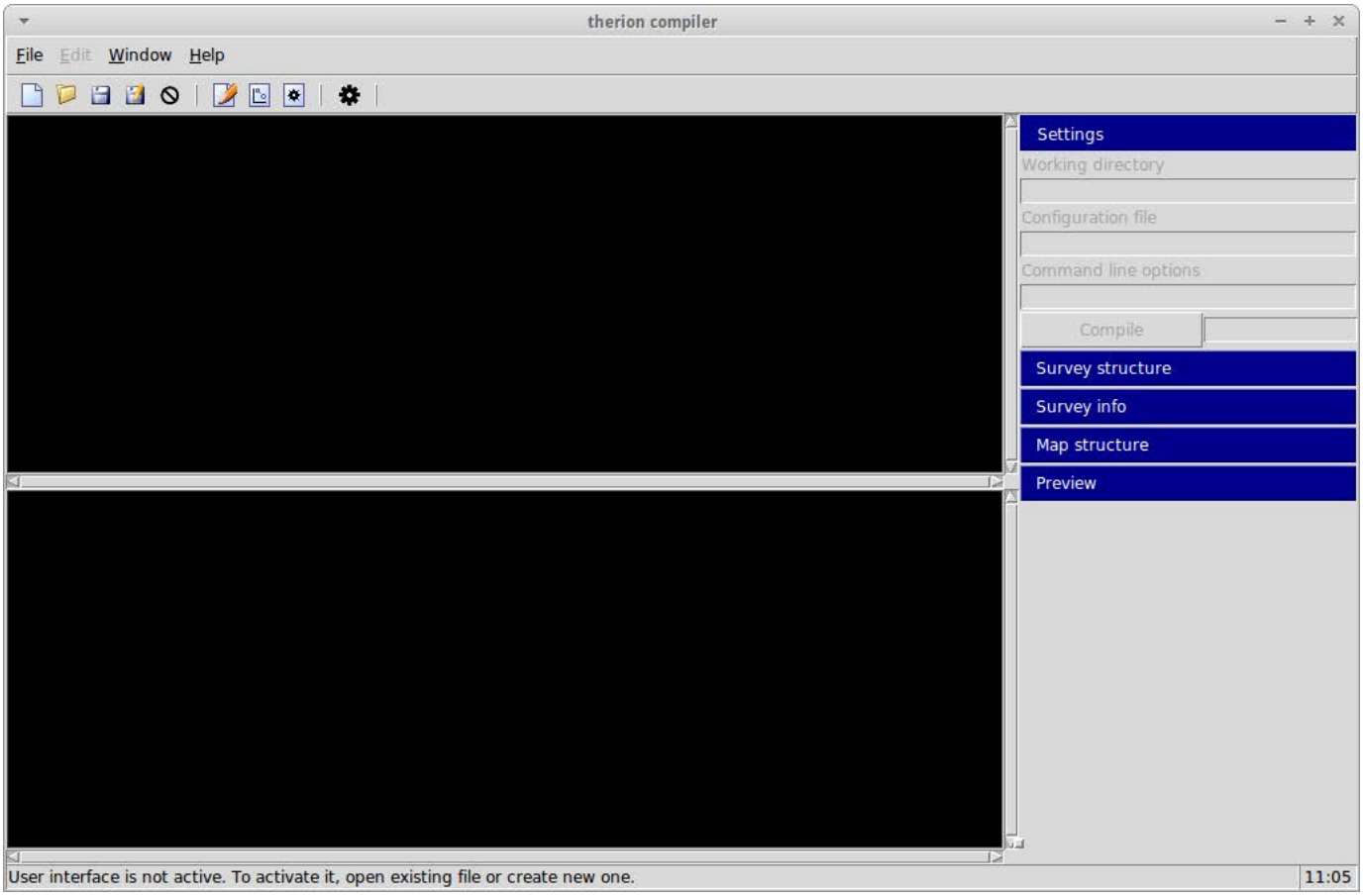


Figure 1. The main window that comes up when you first start Therion.

for this project. Since this is just a quick tutorial, let's just use the name "example1". In the text editor window (press F1 to bring it up), enter the following text:

```
survey example1
endsurvey
```

Between these two lines is where you need to enter your survey data. You could enter this data manually, but usually, it would have been collected using some other system. One example of this is using PocketTopo to go and collect the survey data directly in the field.

If you don't have any data available, you can get some on-line. There is a very good tutorial at the Cave Surveying Group (**http://cp.cavesurveying.org.uk/index.php/articles/3-therion-tutorial**). It also includes some example data you can use while you are learning to use Therion. Let's use the example file bpwth.txt.

In the text editor window, add

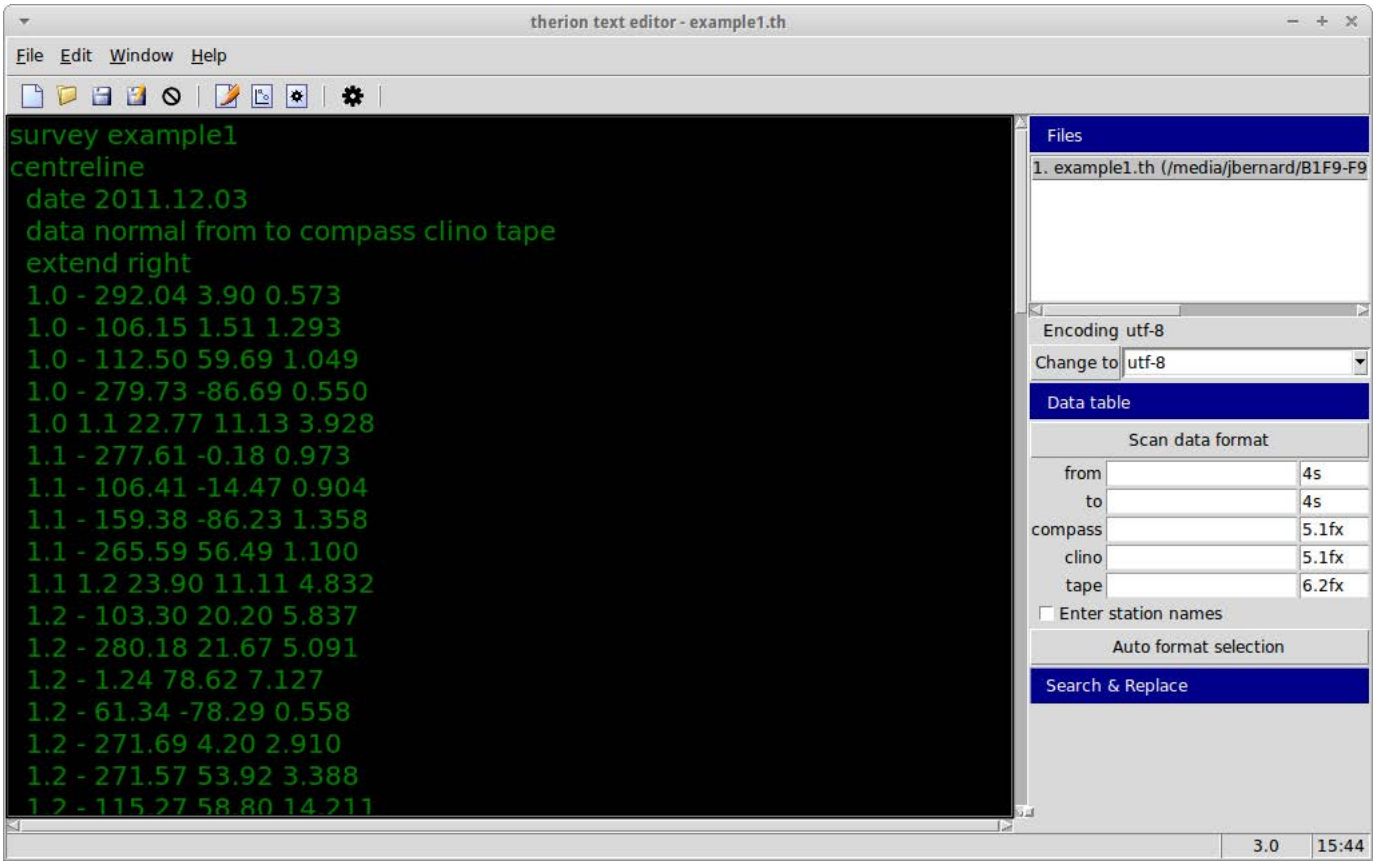


Figure 2. The survey data is simply a list of data points.

an extra blank line between the two lines you already entered, and place the cursor on this new blank line. Now you can click on the menu entry File→Import and select the file bpwth.txt. This will import the data you are going to plot.

Once you have all of your data imported, you need to save it to a file. Because the project's name is example1, name the text file example1.th.

The next step is to run the compile stage to process all of this data. Press F3 to switch to the compile window. You need to give the compiler some options so that it knows what to do, so open a new configuration file by either clicking the menu item File→New or pressing Alt-N. The default filename that comes up is thconfig. To keep things clean, name the file thconfig.thc, and save it in the same location as the text file you created earlier. Click in the top pane to set the focus there, and enter the following text:

```
source example1.th export model -fmt survex -o example1.3d
```

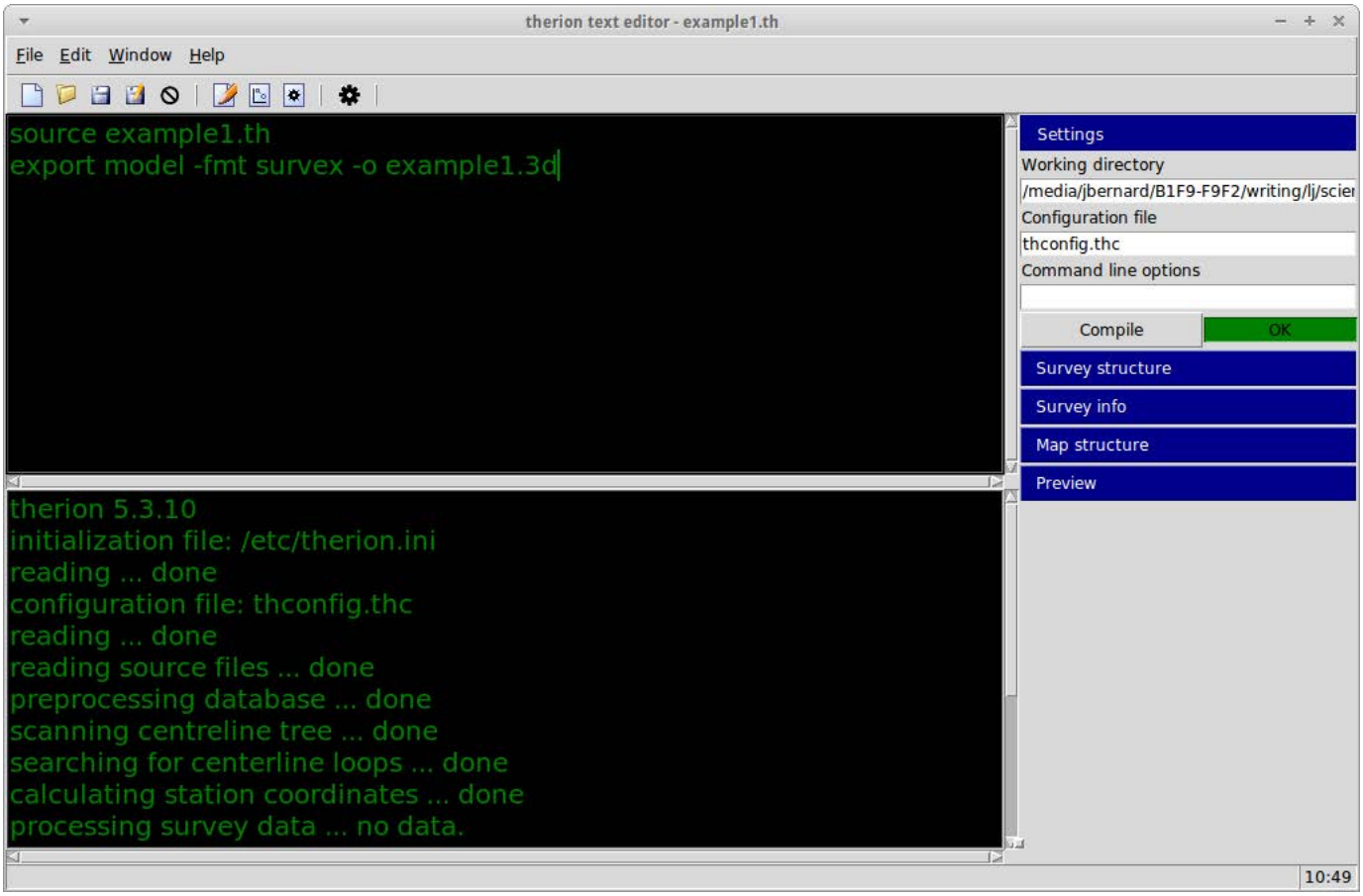


Figure 3. The log of results from the compile shows up in the lower pane of the main window.

The first line tells Therion where the data is to compile. The second line tells Therion what file format to use to save the compiled model. In this particular case, the model is being saved in Survex 3D format. Therion supports several different file formats.

With this information in the system, you now can go ahead and compile your model. You can press F9 or click the menu entry File→Compile. Therion will process your model and write out details in a log, which is displayed in the lower pane.

So, what does this look like? Luckily, Therion includes a viewer called Loch that can handle Survex 3D files. You should have an entry for it in the menu system for your desktop environment. You also can start it up from a terminal window with the command `loch`. It uses OpenGL to draw the three-dimensional model of your cave data. There is also another program called Aven that you can use to visualize the cave data.

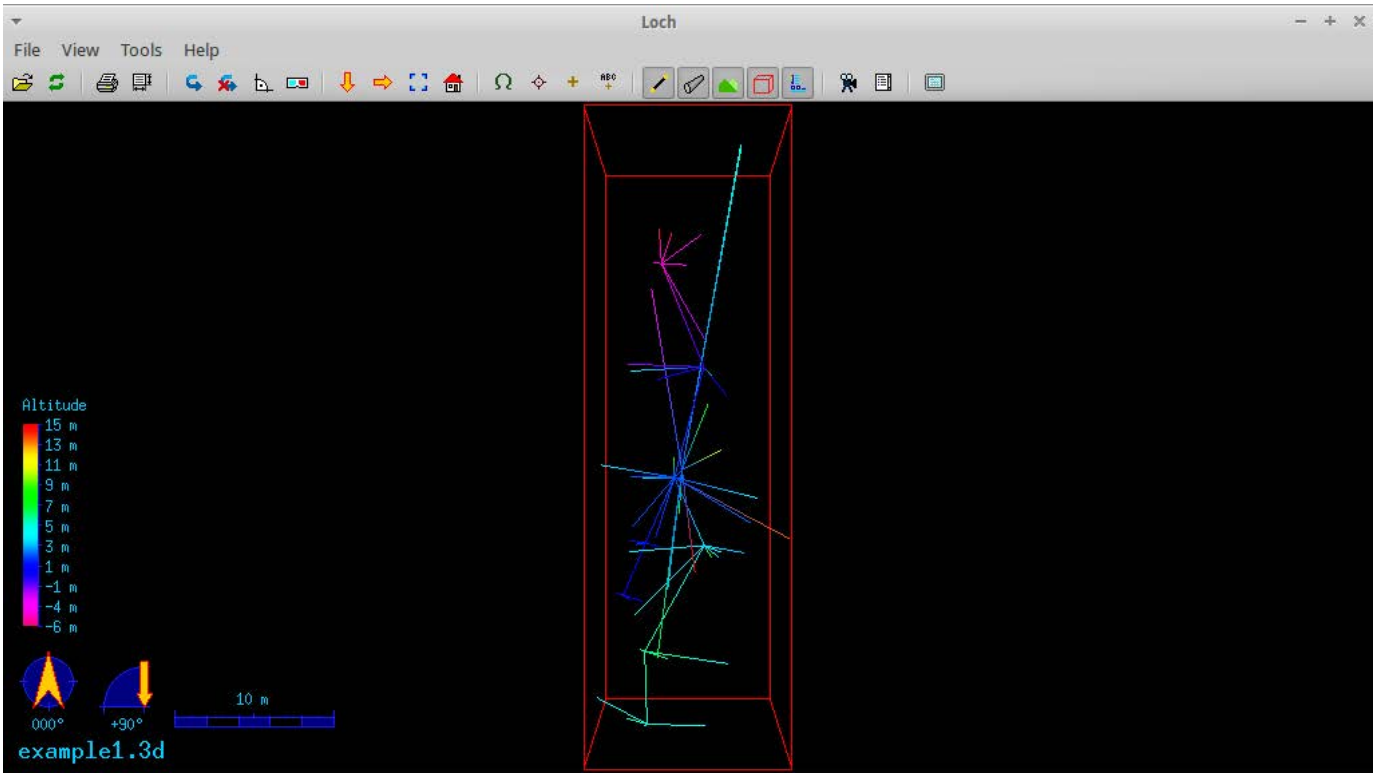


Figure 4. Loch is one of the viewers for 3-D files available from Therion.

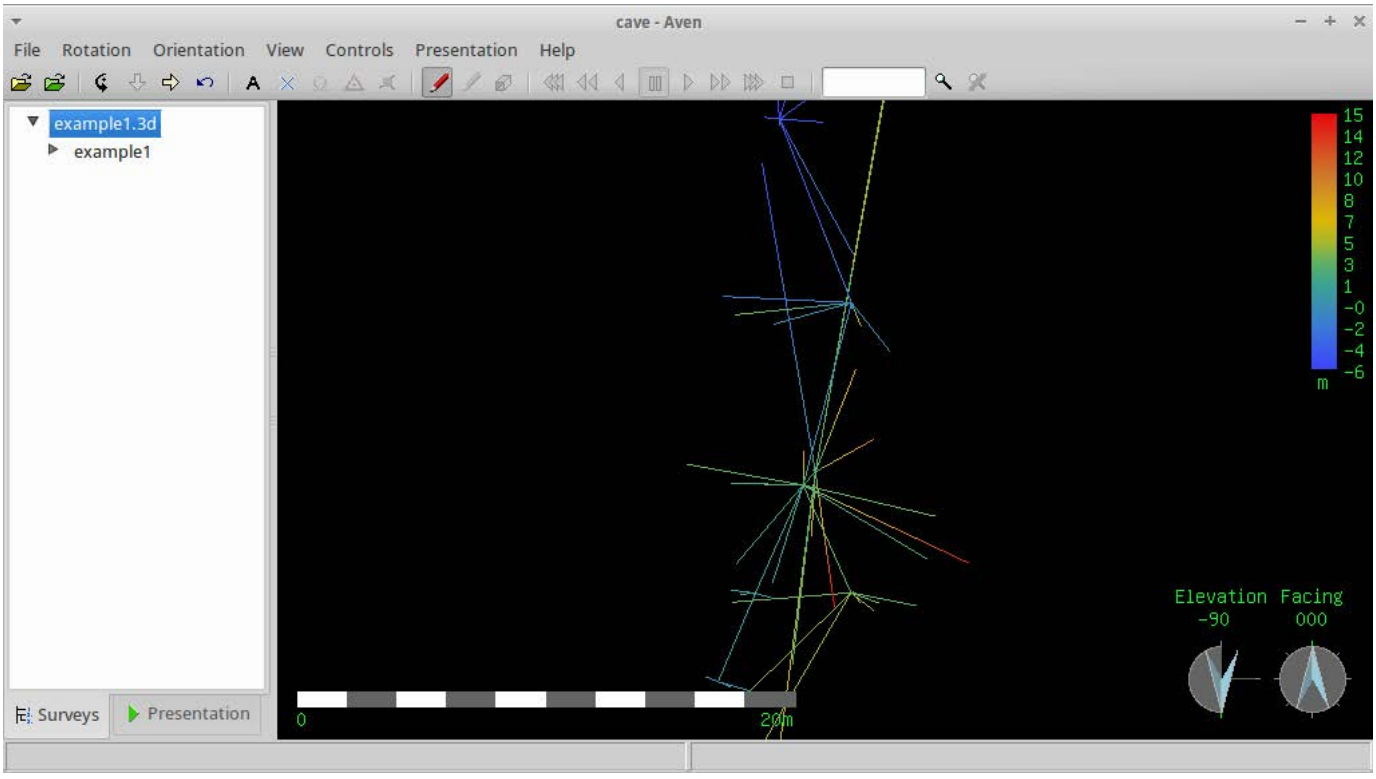


Figure 5. You also can use Aven to view 3-D files generated from Therion.

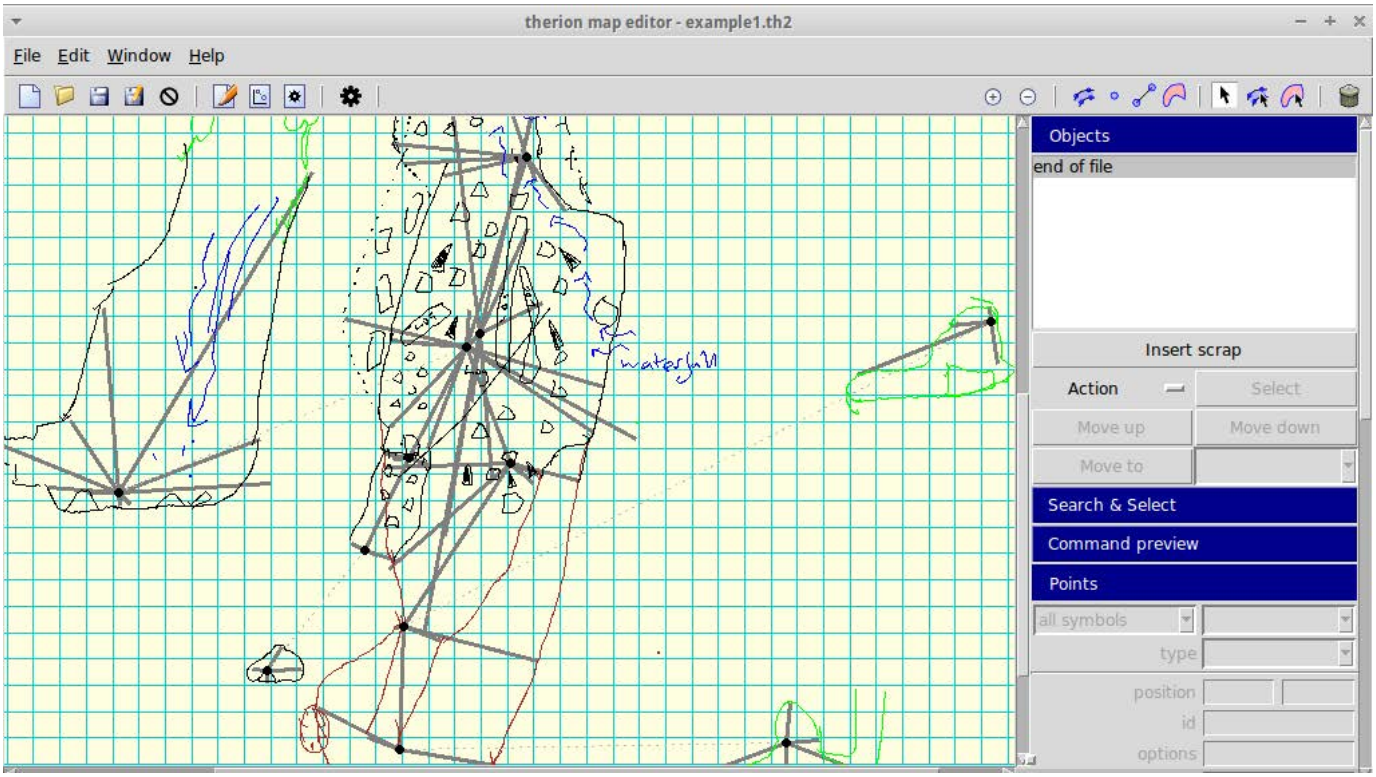


Figure 6. You can import map data from PocketTopo into Therion.

The last window to look at here is the map editor. For this example, let's switch to the map editor by pressing F2. Add a new file by clicking the menu item File→New and save it with the filename example1.th2. The file ending .th2 is the default for mapping files in Therion.

You can add images from several different file formats to your map, but in this case, let's add data from the PocketTopo export data. Click the menu entry Edit→Insert Image, and change the file type to "PocketTopo therion export". Select the same sample data file as before, bpwth.txt. You will end up with a very nice layout of the cave that was surveyed with PocketTopo. You then can edit this map, adding points, lines and areas.

Hopefully, you now will be able to plan safe and rewarding trips into your local caves with Therion. With more and more publicly accessible caves using Therion, you will have a growing set of information available before you ever leave home. Don't forget to check out the other tutorials available on-line to learn all of the other great things you can do with Therion.—**JOEY BERNARD**

11th Annual

2014 High Performance Computing Linux for Wall Street Show and Conference

April 7, 2014 (Monday) **Roosevelt Hotel, NYC**

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

Register and Save $100 on Conference, only $295. $395 Onsite.

HPC, Data Centers, Networks, Low Latency, Big Data, Cloud, Optimization, Linux, at 2014 HPC, April 7, Roosevelt, NYC. Wall Street will be coming to see these systems live at the show.

Wall Street and New York will be coming to the 11th Annual Wall Street IT marketplace at one time and one place in New York, April 7, Monday.

Register and attend this major IT event covering HPC, Data Centers, Networks, Switches, Low Latency, Big Data, Cloud, Optimization, Linux, Cost Savings, and Operational Efficiencies.

Our Show is an efficient one-day showcase and networking opportunity.

Register in advance and save $100. Includes general sessions, drill-down sessions, industry luncheon, exclusive show viewing times, post-show receptions. In advance: $295. On-site: $395.

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

Wall Street IT speakers and Gold Sponsors will lead drill-down sessions in the Grand Ballroom of the convenient Roosevelt Hotel.

Leading vendors will be introducing their latest financial services systems at this One-Day Conference for Wall Street registrants.

Show Hours: Mon, April 7	8:00 - 4:00
Conference Hours: Mon, April 7	8:30 - 4:50

2014 Sponsors

CISCO, IBM, redhat, HP, ARISTA, SAP, DataDirect Networks, SanDisk

LINUX NEW MEDIA The Pulse of Open Source, LINUX Pro Magazine, ADMIN Network & Security, TABOR COMMUNICATIONS, MARKETSMEDIA

LINUX JOURNAL, HPC In the Cloud, datanami, HPCwire, SourceMedia, scientific computing world

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

www.flaggmgmt.com/linux

Alice, the Turtle of the Modern Age

(Image from http://www.alice.org)

Many of us grew up with LOGO, the kid-friendly programming language that guided a little turtle around a screen. Yes, it was simplistic. Yes, it taught only the very basics of programming concepts, but it also inspired an entire generation of programmers. The applications you run every day were written by people who steered a digital turtle around a screen in third grade.

Alice is a project from Carnegie Mellon University that allows new programmers to use a drag-and-drop interface to create 3-D scenes and perform programmatic results without typing any code. The Alice Project has evolved through the years, and it's currently on version 3. The code is freely downloadable and is available for Linux, Mac and Windows.

Although the LOGO programming language allowed for some lengthy instructions for the turtle, it was limited. Alice, on the other hand, uses the animation environment to teach amazingly complex programming concepts. By utilizing an environment where syntax is dragged as opposed to typed, it takes "typos" out of the equation. It's hard to describe just how complex the programming can be with Alice, so I urge you to download it or at least visit the Alice Project at **http://www.alice.org**.

For doing its part in producing the next generation of programmers, while (at least in my mind) continuing the legacy of a small digital turtle from my youth, Alice gets this month's Editors' Choice Award.—**SHAWN POWERS**

O'REILLY®

Fluent CONFERENCE

The Web Platform

San Francisco, CA | March 11–13, 2014

Does Your Future Depend on the Web Platform?

You bet it does. If you're building web applications, designing for mobile devices, or working with the web's evolving infrastructure, you need to keep up with the enormous proliferation of web technologies. At Fluent, you'll find workshops, tutorials, and sessions on all aspects of the Web Platform, including JavaScript, HTML5, WebGL, CSS3, mobile APIs, Node.js, AngularJS, ECMAScript 6, and more. We're even partnering with WebPlatform.org and hosting a Document Sprint. It's everything you need to stay competitive.

Don't miss your chance to be a part of the web's evolution. Build your custom Fluent schedule around exciting talks on topics like:

- Front End Frameworks and Libraries
- HTML5, CSS3, and Browser Technologies
- Node.js
- Pure Code and JavaScript
- The Leading Edge
- The Server Side
- Tools, Platforms, and APIs
- User Interface / User Experience
- HTML5 Gaming

Fluent is closer than you think.

Save 20% with code **LINUX20**

fluentconf.com

REUVEN M. LERNER

TogetherJS

Want to add real-time collaboration to your Web application? Mozilla's TogetherJS is worth a look.

When Tim Berners-Lee invented the World Wide Web more than 20 years ago, he did it in the hopes that physicists would be able to collaborate easily with one another over the Internet. Since then, the Web has grown and morphed into a new medium that handles everything from newspapers to finance to supermarkets.

And yet, although we can marvel at the large number of things we can do on the Web nowadays, the original idea that drove it all, of collaboration, is still a bit of a dream. Sure, we have sites like GitHub, which provide a Web interface to the Git version-control system. And of course, we have plenty of writing systems, such as WordPress, that allow a number of people to create (and publish) documents. And there's also Facebook, which sometimes can be seen as collaborative.

But if you really think about it, we still don't have the sort of seamless collaboration we originally thought might be possible via the Web. Sure, I can work on something, hand it to others, and then work on it again when they are done with it, but it's still relatively rare to have collaborative tools on-line.

Perhaps the most sophisticated and widespread example of real-time, Web-based collaboration is Google Docs (now known, I think, as Google Drive). It's true that Google's applications make it possible for you to store your documents in the cloud, as people now like to say. And it's certainly convenient to be able to read and write your documents from anywhere, so long as you have access to a Web browser. But for me, the real power of Google Docs is in the collaboration. Many different people can work on the same document, and they even can do so at the same time. I found this sort of collaboration to be invaluable when I had to work with several other people to put together a budget on a project several years ago. The fact that we could all, from our own computers,

edit the same spreadsheet in real time was quite useful.

There are a number of open-source alternatives to Google's word processor as well. Etherpad was released as an open-source project after its authors were acquired by Google several years ago. You can download and install Etherpad on your own, or you can take advantage of one of the existing Etherpad servers on-line. Another interesting application is Ace, a browser-based programming editor with impressive collaborative abilities.

Now, I never would claim that all collaboration needs to be in real time. There are many examples in the open-source world of people communicating and collaborating asynchronously, using e-mail and Git to work together—often quite effectively, without the bells and whistles that real-time collaboration seems to offer.

However, for many of us, collaboration without a real-time component is always missing something. It would be great for me not only to be able to talk to someone about a Web site, but also to look at it (and edit its content) along with them, in real time. Yes, there are screen-sharing systems, such as VNC, Join.me and ScreenHero, but they require that you install something on your computer and that you activate it.

That's why I have become interested in a project sponsored by the Mozilla Foundation known as TogetherJS. As the name implies, TogetherJS is a JavaScript-based, real-time collaboration system. The most impressive thing, in my opinion, is how much TogetherJS provides out of the box, with little or no configuration. It allows you to make your site more collaborative by adding some simple elements to each page.

So in this article, I want to look into TogetherJS—what it does, how you can add it to your own sites, and how you even can connect your application to it, creating your own, custom collaborative experience.

What Is TogetherJS?

TogetherJS is a project sponsored by the Mozilla Foundation (best known for the Firefox browser and the Thunderbird e-mail client). Mozilla has been developing and releasing a growing number of interesting open-source tools during the past few years, of which TogetherJS is one of the most recent examples. (In recent months, for instance, Mozilla also released Persona, which attempts to let you sign in to multiple sites using a single identity, without

tying it to a for-profit company.) TogetherJS was released by "Mozilla Labs", which, from the name and description, suggests this is where Mozilla experiments with new ideas and technologies.

On a technical level, TogetherJS is a client-server system. The client is a JavaScript library—or more accurately, a set of JavaScript libraries—loaded onto a Web page, which then communicates back to a server. The server to which things are sent, known in TogetherJS parlance as the "hub", runs under node.js, the JavaScript-powered server system that has become quite popular during the past few years. The hub acts as a simple switchboard operator, running WebSockets, a low-overhead protocol designed for real-time communication. Thus, if there are ten people using TogetherJS, divided into five pairs of collaborators, they can all be using the same hub, but the hub will make sure to pass messages solely to the appropriate collaborators.

Installing TogetherJS on a Web site is surprisingly easy. You first need to load the TogetherJS library into your page. This is done by adding the following line into your Web application:

```
<script src="https://togetherjs.com/togetherjs-min.js"></script>
```

Of course, you also can host the JavaScript file on your own server, either because you want to keep it private or internal, or if you are modifying it, or if you simply prefer not to have it upgrade each time Mozilla releases a new version.

That JavaScript file doesn't actually do much on its own. Rather, it checks to see whether you want to download a minimized version of the code. Based on that, it decides whether to download all of the code in a single file or in separate ones. Regardless of how you download TogetherJS, the above line ensures that the JavaScript component is ready to work.

But of course, it's not enough to install the JavaScript. Someone needs to activate and then use it, which means installing a button on your Web site that will do so. Once again, because of the way TogetherJS is constructed, it's pretty straightforward. You can install the following:

```
<button id="start-togetherjs" type="button"
        onclick="TogetherJS(this);
    ➥return false">Start TogetherJS</button>
```

In other words, you create a button that, when clicked, invokes the TogetherJS function. When users click on that button, they will be added to

TogetherJS. Now, this is actually pretty boring; if you're running TogetherJS by yourself, it's not going to seem to be doing very much. Once you click on the TogetherJS button, and after you click through the first-time introduction, you'll immediately be given a link, labeled "invite a friend". In my particular case, running this on port 8000 of my server, I get http://lerner.co.il:8000/togetherjs.html#&togetherjs=oTtEp6wmoF.

As you can see, the TogetherJS special-invitation URL combines the URL of the page you own, along with a token that uniquely identifies your session. This allows multiple sets of people to collaborate on the same page, with each set existing in its own, isolated environment.

For example, togetherjs.html (Listing 1) is a file that I put up on my server. I opened two separate browsers onto that page, one via the direct URL and the second by using the full URL shown above, with a specific confirmation token. Once both browsers were pointing to the site,

Listing 1. togetherjs.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Collaborate!</title>
  </head>
  <body>
    <h1>Collaborate!</h1>
    <textarea>Hello out there!</textarea>
    <script src="https://togetherjs.com/togetherjs-min.js">
    ➥</script>
    <div id="togetherjs-div">
      <button id="start-togetherjs" type="button"
              onclick="TogetherJS(this); return false"
              data-end-togetherjs-html="End TogetherJS">Start
               ➥TogetherJS</button>
    </div>
  </body>
</html>
```

and once both users had confirmed their interest in collaborating and using TogetherJS, I found that either user could modify the content of the "textarea" tag, and that no matter who was typing, the changes were reflected immediately on the other person's computer. In addition, each click of the mouse is displayed graphically. And if one user goes to a different page on the site (assuming that the TogetherJS library is on the other page as well), TogetherJS will ask the other user if he or she wants to follow along.

Configuration and Customization

It's great and amazing that TogetherJS works so well, immediately upon installing it. However, there also are ways you can configure TogetherJS, so it'll reflect your needs. Because of the way TogetherJS is loaded, it's recommended that you make these configuration settings before TogetherJS has been loaded. For example:

```
<script>
    var TogetherJSConfig_siteName = "Reuven's page";
</script>
<script src="https://togetherjs.com/togetherjs-min.js"></script>
```

All of the configuration variables begin with `TogetherJSConfig_` and have names that are fairly descriptive. A full list is on the TogetherJS site at **https://togetherjs.com/docs/#configuring-togetherjs**, but you also can look through the together.js code, which contains comments describing what the configuration variables do.

For example, if you decide you want to run your own hub (that is, a message-passing, WebSockets-based server), you must tell TogetherJS to look in a different location, with `TogetherJSConfig_hubBase`.

Other useful configuration variables are:

- `TogetherJSConfig_useMinimizedCode`: downloads the minimized JavaScript files for the rest of TogetherJS.
- `TogetherJSConfig_inviteFromRoom`: allows you to think about collaboration on a site-wide basis.
- `TogetherJSConfig_youtube`: when set, this means that if one person views a YouTube video, everyone will, and they will be synchronized.

But, TogetherJS provides more than just the ability to configure and use it. You also can extend it.

The same message-passing system that TogetherJS uses for itself is available to developers. Thus, you can send arbitrary messages, for arbitrary events, between all of the people currently collaborating on this system.

In order to send an arbitrary JSON message, all of the parties involved need to agree on the message "type"—that is, the string that you will use to identify your message. All parties also need to register their interest in receiving messages of this type and to define a callback function that will fire when the message is sent. Although it makes sense to do this within the HTML or JavaScript in your application, it's also possible to do it in your favorite browser's JavaScript console.

First, you register interest in your object by telling TogetherJS that whenever it receives a message of a certain type from a hub (using the `TogetherJS.hub.on` methods), it should fire a particular callback:

```
TogetherJS.hub.on("reuvenTest",
        function (msg) {
            console.log("message received: " + msg)
 }
);
```

Now, it's true that because your message is an object, it will be printed as "[Object object]" in the Web console, which is really too bad. If you prefer, you can choose individual fields from the object, but be sure that you know what fields you will be receiving.

To send a message, just invoke `TogetherJS.send` along with a JSON object that will be sent to all of the other TogetherJS subscribers on this channel. There is no way to send a message to one particular

computer; in this way, the paradigm is similar to what you probably saw with real-time Web updates with Pusher and PubNub last year.

To send a message, you use the following:

```
TogetherJS.send({type: "reuvenTest", foo:
➥'foofoo', bar: 'barbar'});
```

I should note that both of these methods return undefined to the caller. Normally, this isn't a problem, but if you're working in the console, you might not expect such a response.

Of course, the power of this communication channel is in deciding what you want to communicate, and what should happen when communication comes in. When a message arrives, do you want to change the way something is displayed on the receivers' screens? Do you want to display a message for the user?

Conclusion

TogetherJS is relatively new, and I am still in the early stages of using it in my applications that require real-time collaboration. That said, TogetherJS is an interesting application in and of itself, and it was designed as a platform that will continue to grow and expand. With TogetherJS, we are one additional step closer to using the Web for new and different forms of collaboration.■

Reuven M. Lerner, a longtime Web developer, consultant and trainer, is completing his PhD in learning sciences at Northwestern University. You can learn about his on-line programming courses, subscribe to his newsletter or contact him at http://lerner.co.il.

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

Resources

The home page for TogetherJS is **http://togetherjs.com**, but the GitHub page is **https://github.com/mozilla/togetherjs**. You can use TogetherJS with any modern browser, which basically means that it does not support Internet Explorer. I was able to get it to work with Firefox, Chrome and Safari (on OS X) without any problems. Note: TogetherJS explicitly states that it doesn't support Internet Explorer very well, including the most recent versions, which means that if you're working in a mixed environment, it might not be a good choice.

usenix
THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

UPCOMING CONFERENCES

For a complete list of USENIX and USENIX co-sponsored events, see www.usenix.org/conferences

FAST '14: 12th USENIX Conference on File and Storage Technologies

February 17–20, 2014, Santa Clara, CA, USA
www.usenix.org/conference/fast14

2014 USENIX Research in Linux File and Storage Technologies Summit
In conjunction with FAST '14
February 20, 2014, Mountain View, CA, USA
Submissions due: January 17, 2014

NSDI '14: 11th USENIX Symposium on Networked Systems Design and Implementation

April 2–4, 2014, Seattle, WA, USA
www.usenix.org/conference/nsdi14

2014 USENIX Federated Conferences Week

June 17–20, 2014, Philadelphia, PA, USA

USENIX ATC '14: 2014 USENIX Annual Technical Conference
www.usenix.org/conference/atc14
Paper titles and abstracts due January 28, 2014

HotCloud '14: 6th USENIX Workshop on Hot Topics in Cloud Computing

WiAC '14: 2014 USENIX Women in Advanced Computing Summit

HotStorage '14: 6th USENIX Workshop on Hot Topics in Storage and File Systems

UCMS '14: 2014 USENIX Configuration Management Summit

ICAC '14: 11th International Conference on Autonomic Computing

USRE '14: 2014 USENIX Summit on Release Engineering

Do you know about the USENIX Open Access Policy?

USENIX is the first computing association to offer free and open access to all of our conferences proceedings and videos. We stand by our mission to foster excellence and innovation while supporting research with a practical bias. Your membership fees play a major role in making this endeavor successful.

Please help us support open access.
Renew your USENIX membership and ask your colleagues to join or renew today!

www.usenix.org/membership

23rd USENIX Security Symposium

August 20–22, 2014, San Diego, CA, USA
www.usenix.org/conference/usenixsecurity14
Submissions due: Thursday, February 27, 2014

Workshops Co-located with USENIX Security '14

EVT/WOTE '14: 2014 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections

USENIX Journal of Election Technology and Systems (JETS)
Published in conjunction with EVT/WOTE
www.usenix.org/jets
Submissions for Volume 2, Issue 2, due: December 5, 2013
Submissions for Volume 2, Issue 3, due: April 8, 2014

HotSec '14: 2014 USENIX Summit on Hot Topics in Security

FOCI '14: 4th USENIX Workshop on Free and Open Communications on the Internet

HealthTech '14: 2014 USENIX Workshop on Health Information Technologies
Safety, Security, Privacy, and Interoperability of Health Information Technologies

CSET '14: 7th Workshop on Cyber Security Experimentation and Test

WOOT '14: 8th USENIX Workshop on Offensive Technologies

OSDI '14: 11th USENIX Symposium on Operating Systems Design and Implementation

October 6–8, 2014, Broomfield, CO, USA
www.usenix.org/conference/osdi14
Abstract registration due April 24, 2014

Co-located with OSDI '14:

Diversity '14: 2014 Workshop on Diversity in Systems Research

LISA '14: 28th Large Installation System Administration Conference

November 9–14, 2014, Seattle, WA, USA
https://www.usenix.org/conference/lisa14
Submissions due: April 14, 2014

Stay Connected...

twitter.com/usenix
www.usenix.org/youtube
www.usenix.org/gplus
www.usenix.org/facebook
www.usenix.org/linkedin
www.usenix.org/blog

DAVE TAYLOR

Simulating Dice Rolls with *Zombie Dice*

Dave rolls the dice to see if he's getting brains for lunch or a gunshot as he tries to flee—well, at least within his shell script programming project, that is.

I've spent the past few articles describing how to work with ImageMagick, watermarking, resizing and adding 3-D frames to images from within shell scripts. Interesting stuff, but I feel the need to move back to some game programming, because it's more fun to write than something practical and actually useful. See? That's how this year's going to go in my column. I'm going to start out by enjoying programming rather than carrying it as a burden while we Accomplish Important Things. You with me? I thought you would be.

So in this article, I'm going to start writing a multi-dice-rolling game program based on the fun, lightweight game *Zombie Dice*. In the game, you're a zombie trying to accumulate brains without getting shot down by vengeful humans. The game comes with 13 dice: six green, four yellow and three red. Each die has three possible elements: brains (good), gunshots (bad) and feet, which denote the person represented by that particular die has "done a runner" and you'll need to roll that particular die again next turn.

Out of the 13 dice, you randomly pick three each time, then roll them, accumulating both brains and gunshots. If you get to three gunshots, you're dead. If you get to 13 brains, you win. That's about it. In the real game, you take turns with someone else, but I'll get into the nuances a bit further in the program. For now, let's start by looking at how to simulate each individual die.

In the game, you're a zombie trying to accumulate brains without getting shot down by vengeful humans.

Remember, there are three different colors, the green being the "easiest" humans to defeat, and the red being the "hardest".

In programmatic terms, and using some additional variables to make the notation easier to understand, here's how I am going to simulate the different sides of these dice:

```
BRAIN=1; SHOT=2; RUNNER=3
gdie[1]=$BRAIN; gdie[2]=$BRAIN; gdie[3]=$BRAIN;
gdie[4]=$SHOT; gdie[5]=$RUNNER; gdie[6]=$RUNNER;
ydie[1]=$BRAIN; ydie[2]=$BRAIN; ydie[3]=$SHOT;
ydie[4]=$SHOT; ydie[5]=$RUNNER; ydie[6]=$RUNNER;
rdie[1]=$BRAIN; rdie[2]=$SHOT;  rdie[3]=$SHOT;
rdie[4]=$SHOT; rdie[5]=$RUNNER; rdie[6]=$RUNNER;
```

As you can see, each color die has six options—since they're all six-sided dice—and the differences in color are reflected in the number of brains, shots and runners each has specified.

Using all-caps variables as placeholders for numeric values is a bit of an old-school programming technique, but the script will run 99% as fast, and the value of the mnemonics makes writing a lot of the later code far easier and helps avoid errors.

I also start my indexing at 1, not zero. Why, when Bash and other shells use zero-index arrays? Um, I don't know, but likely it's just because it makes more sense to me to refer to "faces one through six" on a die than "zero through five". Since I'm using the mnemonic names rather than just numeric values, however, I ostensibly could change some of this as desired if it really bugs you. The first actual script segment requires more mnemonic variables so I can stick with the desire to write a very clear script:

```
GREEN=1; YELLOW=2; RED=3
function pick_color
{
  # returns 0 for green, 1 for yellow and 2 for red
  case $( expr $RANDOM % 13 + 1 ) in
    1|2|3|4|5|6  ) color=$GREEN  ;;
    7|8|9|10     ) color=$YELLOW ;;
    11|12|13     ) color=$RED    ;;
  esac
}
```

As with just about every game, randomness is important. That's why so many games have decks to shuffle, dice to roll and so on. Totally deterministic games are generally a lot less fun (although not always—neither Chess nor Go have any randomness to them).

Bash makes working with random numbers super easy. The special variable `$RANDOM` is assigned a random integer value between 1-MAXINT every time you reference it—no need to "seed" it.

Here's an easy and interesting little test:

```
$ while [ $i -lt 10000 ] ; do
  echo $RANDOM ; i=$(( $i + 1 ));
done | sort | uniq -c | sort -rn | head -5
   4 7281
   4 6947
   4 6309
   4 31328
   4 28955
```

This spits out 10,000 random numbers, then analyzes how frequently each occurs, displaying the five most common. In this case, you can see that the most commonly occurring number appears only four times out of 10,000 numbers. Try this on your system: if you see a number appearing significantly more, you might have a less random number system than you want.

Now where was I? Oh yeah, rolling dice. Since I'm working in a shell script, the idea of a function "returning" a value is pretty non-existent, so instead I'll be setting global variables within local function scope. It's sloppy programming, but a constraint of shell scripting, so bear with it and know that if you want to write this in Ruby, Perl or similar, you'll be able to have a more rigorous policy with your own variables.

Six out of 13 dice are green, four are yellow, and three are red. You can see how that's easily implemented with the case statement.

In a very similar spirit, rolling an individual die is easily accomplished, particularly because of the array of dice sides by color set up earlier in the script. Here's all that's needed:

```
function roll_die
{
  # returns 1 for a brain, 2 for gunshot, 3 for a runner

  dieroll=$( expr $RANDOM % 6 + 1 )

  case $1 in
    $GREEN  ) roll=${gdie[$dieroll]} ;;
    $YELLOW ) roll=${ydie[$dieroll]} ;;
    $RED    ) roll=${rdie[$dieroll]} ;;
  esac
}
```

A two-dimensional array would make this even easier, with a single line:

```
roll=${die[$COLOR][$dieroll]}
```

But in the interest of legibility, I've slightly complicated things with the three different arrays. Still, the function's delightfully small and easily understood.

Armed with both of these, it's easy to pick a color randomly and then roll that die:

```
pick_color
roll_die $color
```

Notice that the function `roll_die` is written to expect the die color to be specified as its parameter. Since we are using global variables, I suppose I should just refer to `$color` in the function, but that makes me cringe just a little bit. Know what I mean?

Let's add some code to make it easy to display what I've "rolled" too. This is easily done with arrays again:

```
colorname[$GREEN]="green"; colorname[$YELLOW]="yellow";
 ➥colorname[$RED]="red";
nameof[$BRAIN]="brain"; nameof[$SHOT]="gunshot";
 ➥nameof[$RUNNER]="runner";
```

Now it's this easy to show what's been rolled:

```
echo "Rolled ${colorname[$color]} die: ${nameof[$roll]}"
```

So let's roll a few dice:

```
$ ./zdice.sh
    rolled green die: runner
    rolled green die: brain
    rolled red die: runner
$ ./zdice.sh
    rolled green die: brain
    rolled red die: runner
    rolled red die: brain
```

That's a good start. Next month, I'll look at how to accumulate brains and track those dangerous gunshots. Meanwhile, why not pick up the actual game, for sale at Target and a zillion other places. It's from Steve Jackson Games: http://www.sjgames.com. ■

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 can be found on Twitter as @DaveTaylor and more generally at http://www.DaveTaylorOnline.com.

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

KYLE RANKIN

Tails above the Rest: the Installation

Ready for extra privacy on top of the Tor Browser Bundle? Try out the Tails live disk.

Two columns ago, I started a series aimed at helping everyone improve their privacy and security on the Internet. The first column in this series was an updated version of a Tor column I wrote a few years ago. While the new column talked about how to get up and running with Tor using the Tor Browser Bundle, in my original Tor column, I talked about how to browse even more anonymously by using a Knoppix live CD and live-installing Tor on top of it. That way, when you were done with your session, you simply could reboot and remove the Knoppix disc, and the host computer would be back to normal with no trace of your steps. Although the Tor Browser Bundle is incredibly useful, it doesn't cover the case where you would like to use Tor on a computer you don't own, much less using Tor without leaving a trace.

These days, there is a much better option than a Knoppix live CD—a live DVD/USB distribution called Tails, which does so much more than provide a live CD with Tor installed. Think of it like a live DVD version of the Tor Browser Bundle, only for the entire OS. With Tails, you boot in to a live environment either from a DVD or USB stick that is set with incredibly secure defaults. Tails takes great lengths to provide you with a secure, anonymous environment. Among other things, Tor automatically launches and connects once you start the desktop, and all communications are routed over it. By default, nothing you do in your desktop session persists—everything is not only erased when you reboot,

In addition to downloading the ISO image, instead of downloading an md5sum to validate it, you also will download a cryptographic signature.

Tails actually goes further and attempts to scrub the contents of RAM before it does reboot. Tails includes a Web browser configured much like in the Tor Browser Bundle with a number of privacy-enhancing plugins and settings already in place. Beyond that, it includes a password manager (Keepassx), GPG encryption software, an e-mail program (Claws), LUKS disk encryption software, secure chat via Pidgin with the OTR (Off the Record) plugin already installed, and it even includes an on-screen keyboard you can use to type in passwords if you are concerned a keylogger might be installed on your host machine.

In the next few columns, I'm going to discuss how to get and validate the Tails distribution and install it. I will follow up with what Tails can and can't do to protect your privacy, and how to use Tails in a way that minimizes your risk. Then I will finish with some more advanced features of Tails, including the use of a persistent volume (with this feature, depending on your needs, you could conceivably use Tails as your main Linux distribution).

Get Tails

If there is someone you trust who already has Tails, installation is easy. Either have them make a copy of their DVD, or in the case of a USB install, have them boot a computer into Tails, insert your USB disk, and then click Applications→Tails→Tails Installer. Finally, click Clone & Install in the GUI window that appears, and Tails will wipe out any data on your USB disk and clone itself there.

If you don't already know someone with Tails, installing Tails is a little bit more involved than with a normal live DVD. This is because if you truly want to use Tails for its intended purpose—security and anonymity—you will want to perform a few additional steps to confirm that the version of Tails you downloaded hasn't been tampered with.

If you have read my column about the Tor Browser Bundle, these steps will seem familiar. In addition to downloading the ISO image, instead

of downloading an md5sum to validate it, you also will download a cryptographic signature. Then you will use that signature to validate that ISO image is the legitimate unmodified version. Finally, if you burn to a DVD, you will follow the same procedure you would use for any other DVD image, or if you want to install to a USB disk, there are a couple extra steps. I'll go over all of this below.

First, let's get Tails. The official Tails site is at **https://tails.boum.org**. (It's HTTPS, so be sure to check whether you get a certificate warning.) From the main page, you should see an image you can click to take you to the download page for the latest version of Tails (at the time of this writing, the latest version is 0.22). At the download page, you should see two main options: either download the ISO image and the corresponding signature as separate downloads, or use the torrent download, which will pull both files down together. The ISO is almost 1GB, so depending on your Internet connection, it may take some time. Once you have both the .iso file and the corresponding .sig signature file, you are ready to move on to validating the image.

Validate the ISO Image

Normally when you download an ISO or other large file, there is a .md5 checksum file that corresponds to it. The purpose of the .md5 file is just to validate that the large ISO file wasn't corrupted during the download and that the file that is on your desktop matched the file on the server. In the case of security software like Tails, unfortunately that kind of verification isn't enough. In particular, if attackers were able to position themselves between you and the Tails server (a Man in the Middle or MITM attack), the attackers potentially could send you a modified version of the Tails ISO that contained a backdoor. If they could do that, they also could send you a matching .md5 checksum file that matches their ISO.

To counter this threat, instead of a .md5 checksum, Tails has you download a cryptographic signature file created by the private key of the Tails maintainer. Much like cryptographic checksums on your distribution's packages, this signature file could be created only by a person with access to a specific GPG private key. In a MITM attack scenario, if attackers try to send you a modified version of the ISO, they would not be able to send you a modified version of the .sig signature file that would

Note: if you are truly paranoid, you also can use GPG to validate further that this signature was created with the actual Tails signing key by taking advantage of the fact that the Tails maintainer has gotten the signing key signed by a number of Debian maintainers.

match without themselves having access to the official private key.

Validate the Signature with sha256sum

Since Tails users need to care a bit more about security than the average user, you will need to go through the extra step of validating this signature. Depending on how paranoid you are, there are a few ways you can go about this. The simplest way is to attempt to download the signature file from multiple computers that are in different locations (and even in different countries if you can swing that; see my "Raspberry Strudel: My Raspberry Pi in Austria" article in the February 2013 issue about one method of colocating a Raspberry Pi in another country). Then, confirm that all of the checksums match. The idea here is that even if someone were able to perform a MITM attack or otherwise compromise your home computer or home Internet connection, it would be much more difficult also to compromise the connection at a public computer at a library, all the computers your friends use and the computer you have at work. With this in mind, simply download as many different copies of the signature file from as many different locations you can, and then use a tool like sha256sum (like md5sum, just using a different algorithm) to compare the checksum of all the files to make sure they are all the same:

```
$ sha256sum tails-i386-0.22.iso.sig
4578929f419d7f4bc99b99ec17a6c0ff3936c5bb02938d3940bac2b93580383b
➥tails-i386-0.22.iso.sig
```

In fact, if you are downloading the same version of Tails as I'm mentioning in this article, you even could use the signature published here as an extra point to compare against.

Note: if you are truly paranoid, you

also can use GPG to validate further that this signature was created with the actual Tails signing key by taking advantage of the fact that the Tails maintainer has gotten the signing key signed by a number of Debian maintainers. This process is a little more involved, but if you want to go that route, it is well-documented at **https://tails.boum.org/doc/get/trusting_tails_signing_key/index.en.html#index3h1**.

Validate the ISO with GPG

Once you have validated the signature, you can use it to validate the ISO. First, you need to download the public part of the signing key that was used for this signature from **https://tails.boum.org/tails-signing.key**. Once you have that signing key, import it into your GPG keyring:

```
$ cat tails-signing.key | gpg --keyid-format long --import
gpg: key 1202821CBE2CD9C1: public key "Tails developers
 ➥(signing key) <tails@boum.org>" imported
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)
```

With the signing key imported, you now can verify the ISO image with GPG:

```
$ gpg --keyid-format long --verify tails-i386-0.22.iso.sig
 ➥tails-i386-0.22.iso
```

If you have added the signing key to your keyring, you will get a reply like:

```
gpg: Signature made Mon 09 Dec 2013 02:50:48 PM PST
gpg:                using RSA key 1202821CBE2CD9C1
gpg: Good signature from "Tails developers (signing key)
     ➥<tails@boum.org>"
gpg:                 aka "T(A)ILS developers (signing key)
     ➥<amnesia@boum.org>"
Primary key fingerprint: 0D24 B36A A9A2 A651 7878  7645
 ➥1202 821C BE2C D9C1
```

Otherwise, you will more likely see the following output:

```
gpg: Signature made Mon 09 Dec 2013 02:50:48 PM PST
gpg:                using RSA key 1202821CBE2CD9C1
gpg: Good signature from "Tails developers (signing key)
     ➥<tails@boum.org>"
gpg:                 aka "T(A)ILS developers (signing key)
     ➥<amnesia@boum.org>"
gpg: WARNING: This key is not certified with a trusted
              ➥signature!
gpg:          There is no indication that the signature
              ➥belongs to the owner.
Primary key fingerprint: 0D24 B36A A9A2 A651 7878  7645
 ➥1202 821C BE2C D9C1
```

Either output means the signature matched, and you have the legitimate ISO. The warning in the second reply simply means you haven't personally signed the Tails signing key with your own key, so it's not part of your web of trust.

The slightly tricky scenario is the case where you need to create a Tails USB disk from the ISO, as that requires a few specific commands.

The following reply is one to look out for. If you see this, it means the ISO was not correct and either downloaded incorrectly or was tampered with and can't be trusted:

```
gpg: Signature made Mon 09 Dec 2013 02:50:48 PM PST
gpg:                using RSA key 1202821CBE2CD9C1
gpg: BAD signature from "Tails developers (signing key)
     ➥<tails@boum.org>"
```

Install Tails to a Disk

I already mentioned the simplest and safest way to install Tails (via someone else's install) at the beginning of this article. Also, these days I'm assuming if you have a DVD burner, you know how to burn a new DVD from an ISO file. The slightly tricky scenario is the case where you need to create a Tails USB disk from the ISO, as that requires a few specific commands. The one caveat to this technique is that you will not be allowed to create a persistent volume on this particular USB Tails install if you use this method. Instead, you will use this process to bootstrap a first Tails USB key and then use that to install Tails to a second USB disk using the official installer built in to the Tails desktop.

The first step is to modify the ISO so that it can function as a bootable USB image. To do this, you need the isohybrid utility, which is part of the syslinux software suite. On a Debian-based system, you can run `apt-get install syslinux`, and on other distributions, your package will likely also simply be called syslinux.

Once the application is installed, plug in your USB stick and attempt to identify which device it shows up as. It's important you get the device right, as you are going to overwrite the device with a Tails install, and if you pick the wrong device name, you could end up wiping out the wrong drive—possibly including the main OS on your computer! If you already have a partition on the device and your desktop environment automatically mounts USB drives, you can use the `df` utility in a terminal to confirm the device:

```
$ df
Filesystem           1K-blocks      Used Available Use% Mounted on
rootfs               146410652 25812772 120597880  18% /
```

```
udev                        10240        0     10240   0% /dev
tmpfs                      398856      728    398128   1% /run
/dev/mapper/sda2_crypt 146410652 25812772 120597880  18% /
tmpfs                        5120        0      5120   0% /run/lock
tmpfs                      797700      120    797580   1% /run/shm
/dev/sda1                  188403    24916    153759  14% /boot
/dev/sdb1                 7503668   148036   6974464   3% /media/data
```

The drive likely will be the last device to show up on the list. In this case, my drive was mounted at /media/data, and the drive's name is /dev/sdb (/dev/sdb1 is the first partition on that drive). Be sure to unmount the device before you proceed. If your desktop environment doesn't automatically mount USB disks or if you have no partition on the drive, you may have to use a tool like dmesg to see the last disk device it mentions:

```
$ dmesg | grep sd
. . .
[291588.322874] sd 5:0:0:0: Attached scsi generic sg1 type 0
[291589.768931] sd 5:0:0:0: [sdb] 15248832 512-byte logical
 ➥blocks: (7.80 GB/7.27 GiB)
[291589.769424] sd 5:0:0:0: [sdb] Write Protect is off
[291589.769433] sd 5:0:0:0: [sdb] Mode Sense: 23 00 00 00
[291589.769910] sd 5:0:0:0: [sdb] No Caching mode page present
[291589.769920] sd 5:0:0:0: [sdb] Assuming drive cache: write through
[291589.773642] sd 5:0:0:0: [sdb] No Caching mode page present
[291589.773646] sd 5:0:0:0: [sdb] Assuming drive cache: write through
[291589.791319]  sdb: sdb1
[291589.793656] sd 5:0:0:0: [sdb] No Caching mode page present
[291589.793662] sd 5:0:0:0: [sdb] Assuming drive cache: write through
[291589.793666] sd 5:0:0:0: [sdb] Attached SCSI removable disk
[291590.178671] EXT3-fs (sdb1): using internal journal
[291590.178679] EXT3-fs (sdb1): mounted filesystem with ordered data
 ➥mode
```

Again, the last disk that is mentioned in the output should correspond with the drive you inserted. In this case, I see the device was assigned sdb, and the kernel detected one partition: sdb1.

Now that I know that the device is /dev/sdb, I can modify the Tails ISO with `isohybrid` and then use the `dd` utility to write it to the disk. Instead of modifying the ISO I downloaded and verified, I like to make a copy of it first and then use `isohybrid` on the copy:

```
$ cp tails-i386-0.22.iso tails-i386-0.22-isohybrid.iso
$ isohybrid tails-i386-0.22-isohybrid.iso --entry 4 --type 0x1c
$ sudo dd if=tails-i386-0.22-isohybrid.iso of=YOURDEVICE bs=1M
```

Note that in the `dd` command, you will need to replace `YOURDEVICE` with your actual USB disk, such as /dev/sdb. I didn't put an actual device name in there in case someone accidentally copies and pastes the above lines into a terminal and presses Enter without reading the commands carefully. The `dd` command might take some time to complete, but provided you don't see any error messages, the image

should have been copied correctly. Now all you need to do is reboot into Tails, and if you do want to take advantage of a persistent disk (which I will cover in a future column), you will want to use the Tails installer from within Tails to clone and install to a second USB disk.

How to use Tails and all of the software it includes is a big enough topic that I will cover it in my next column. If you can't wait until next month and do boot in to the environment, just click Login. Tails should connect to your network automatically, and once the Tor network is set up, it will launch a safe Web browser for you to use. I'll talk more about how best to use Tails next month.■

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

A Little GUI for Your CLI

Irssi has everything but a GUI, until today.

I've tried pretty much every IRC client available for both Linux and OS X. (I use both platforms during my day job.) No matter how many times I try to find a GUI application that meets my needs, I always turn back to Irssi. It works so well, and I can access it from anywhere (Figure 1). Thanks to my Raspberry Pi colocation in Austria, I always can stay logged in and never miss a message. Unfortunately, the one thing I wish Irssi had is a pop-up notification when someone sends me a message or mentions me in a channel. So, I decided to give Irssi a little GUI. It was fun, and as it turns out, it works very well.

If you're running Irssi locally on the Linux machine you're sitting in front of, this process is much simpler. Since my Irssi-connected machine is on another continent, the process is a little more complex, but also a lot more fun. In order to get local GUI notifications for remote mentions of my user name, I need to accomplish a few things:

1. I have to get Irssi to generate some sort of event and/or log when I'm mentioned in an IRC channel or query.

2. I have to parse that information and transfer it to my local machine over the Internet in real time.

3. I need to display a GUI pop-up on my local machine when the remote events occur, again in real time.

4. I need to be able to disconnect and reconnect to a screen session and have the GUI notifications follow me, regardless of what computer I'm actually on. (In my case, this means Linux or OS X.)

Thankfully, Linux supplies all the tools I need, and with a little bit of scripting, I can get the system working.

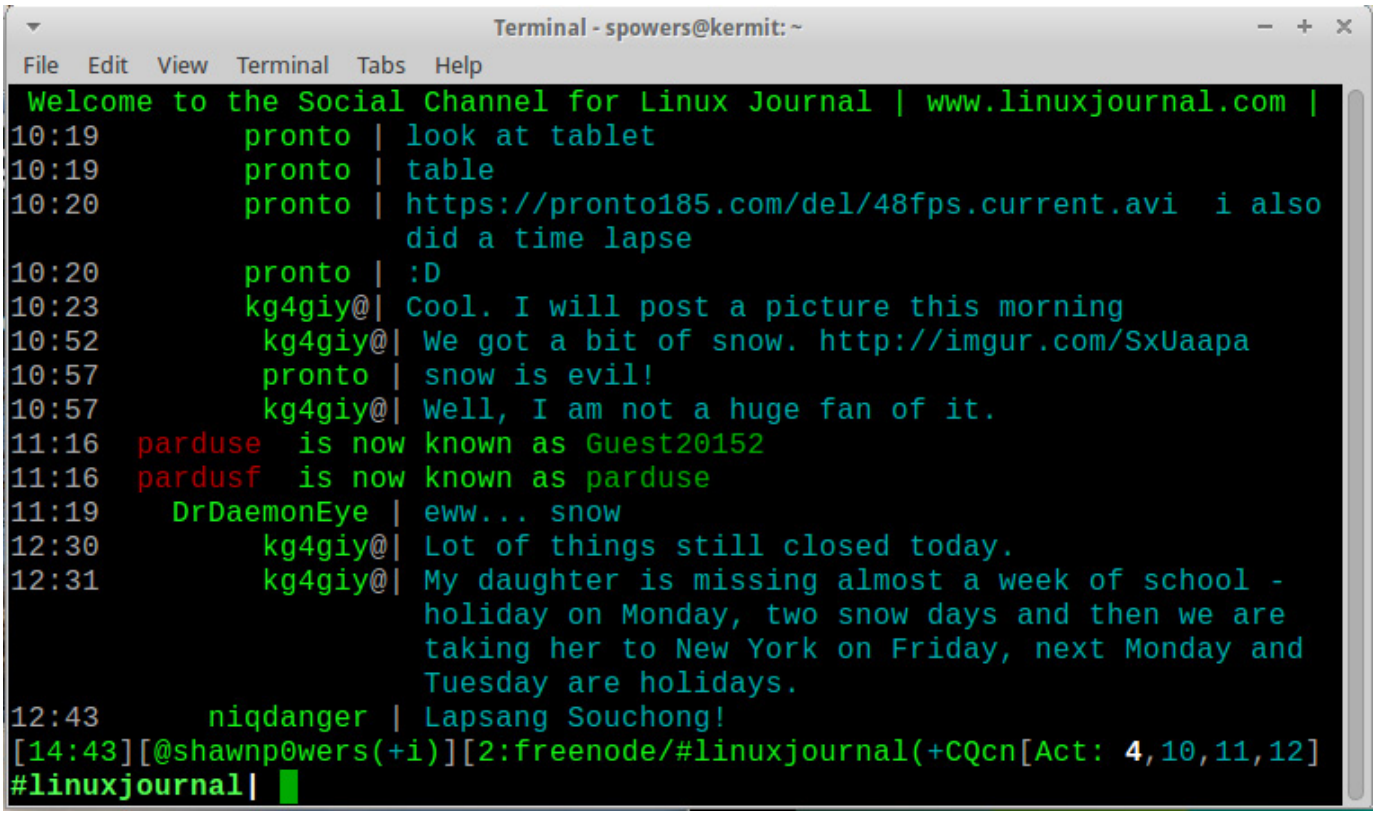


Figure 1. I actually had to scroll up a bit to find something harmless enough to post in the magazine!

Step 1: Irssi Events

I'm obviously not the first person to want a GUI notification system for Irssi, and thankfully, Thorsten Leemhuis has shared his Irssi plugin for everyone to use. You can download the plugin at **http://www.leemhuis.info/files/fnotify/fnotify**, and since it's released under the GPL, you can share it as you see fit.

In order to use the script, save it as fnotify.pl in your ~/.irssi/scripts/ folder where Irssi is running. For me, that's on my remote server in Austria. Once saved, you can load the script by typing:

```
/script load fnotify.pl
```

from inside the Irssi application. If you want to make fnotify.pl be loaded automatically every time Irssi starts (my recommendation), create a symbolic link into the autorun folder inside the scripts folder. To do that, type:

```
mkdir ~/.irssi/scripts/autorun (if it doesn't exist already)
ln -s ~/.irssi/scripts/fnotify.pl
➥~/.irssi/scripts/autorun/fnotify.pl
```

Then fnotify should be loaded every time Irssi starts. To make sure it's working correctly, have someone mention you in an IRC channel, and check to see that the message is written to the file ~/.irssi/fnotify.

Once you're certain the plugin is working, and mentions of your name are written to the fnotify file, it's time to get that information to your local computer. But first, you need to set up the local computer for GUI pop-up messages, so you have somewhere to *send* the information when you transfer it.

A Local GUI Notification

This part of the equation is fairly simple. Most Linux distributions come with a notification system of some sort. I prefer a GNOME environment, so I choose to use libnotify, or more specifically the `notify-send` command that creates a GUI pop-up when invoked from the command line or from a script. KDE users can use the `kdialog` program for a similar effect, and OS X users will want to check out Terminal Notifier, hosted at **https://github.com/alloy/terminal-notifier**.

From your Linux command line, test `notify-send` by typing:

```
notify-send "Test Title" "Sample message..."
```

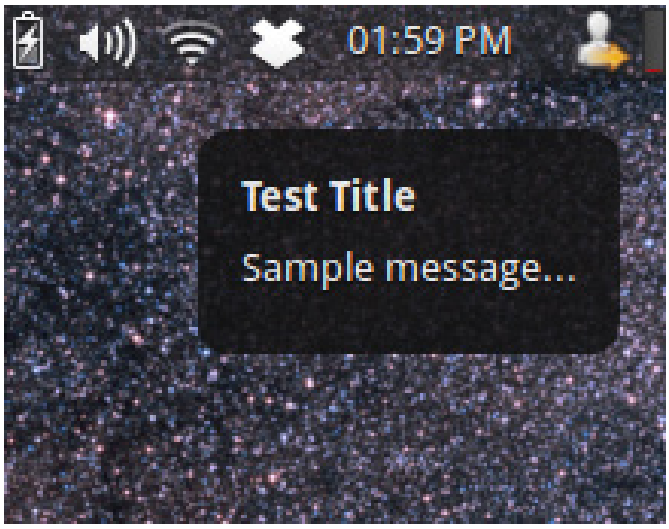


Figure 2. The pop-up system is really pretty slick.

That should bring up a libnotify box with your title and message (Figure 2). If you're using KDE (or OS X), the command will be similar, but you'll need to check for the exact syntax. It's also possible to tweak the program to get custom icons, change the duration of the pop-up and even change the system urgency for the notification. I actually use a Pidgin icon for the notification box, only because I've used Pidgin for so long, the icon makes me think "chat".

Remote fnotify, Local libnotify

This is the part of the equation that stumped me for a long time. I'd done remote X tunneling over SSH, but since Irssi didn't have a native GUI pop-up, there was no X stuff to

tunnel. I also knew that if I SSH'd to my remote server, any commands I ran would execute on the remote server. Thankfully, with enough googling and head-scratching, I learned something: it's possible to run a remote program *other than a shell* over SSH! I had never considered that as a possibility, but it certainly makes sense.

Knowing that I could run a program over SSH on a remote server and have its output appear on my local machine, it turned into a simple scripting matter to get the fnotify text piped in to libnotify. I should point out that it's much easier to do remote SSH stuff if you have a keypair set up to avoid the need to type in a password. It's possible to run the commands interactively, but you lose a lot of the automation aspect. I have a simple tutorial video for setting up SSH keypairs here: **https://www.youtube.com/watch?v=R65HTJeObkI**.

Here is my script; I'll explain it afterward:

```
#!/bin/bash
# Location: /usr/local/bin/irc_notify
(ssh spowers@my.remote.server.com \
-o PermitLocalCommand=no -o ServerAliveInterval=30 \
"> .irssi/fnotify; tail -f .irssi/fnotify" |
while read title message; do
notify-send -u critical -i pidgin "${title}" "${message}";
done)
```

I must confess again that the code above has been scrounged from other people's work and tweaked by me a bit. I certainly don't claim to be the owner of the concept. The script assumes keypairs are installed, but it probably will work interactively if there are no keypairs set up.

Here's what the script does:

1. The script launches SSH with a couple options. I added the `ServerAliveInterval=30` option because in my case, my SSH connection would time out after a while. That took away the whole point of the script, so that option is to keep the connection alive.

2. Instead of running a shell, which is the default, I specified what program SSH is supposed to run on the remote machine. I actually have it do two things: first, clear the fnotify file in .irssi, and second, run `tail -f` on the same file. I clear the file so it doesn't get too long, but if you want to keep a record of all of your messages, you can leave that first part off—it won't affect how the script works.

3. `tail -f` is a command that stays running and prints any

new entries to the specified file to the command line. It's normally used for watching log files, but in this case, it's perfect. Every time someone mentions my user name, `tail -f` prints it to the screen.

4. Notice at the end of that line is the pipe symbol (|)—that sends the output to the program following it, instead of printing it to the screen. In this case, it "pipes" the output of `tail -f` into a while loop that reads the text into two different variables. The command `read title message` assigns the first word in the output (usually the name of the person messaging me) to the variable `$title` and the rest of the message to the variable `$message`.

5. The while loop then runs `notify-send` with the variables from above, and starts the loop over, waiting for another output from the remote SSH'd tail command. To end the program, you can press ^C. (But I actually automate that too; more on that a bit later.)

The script itself is simple, but it can be confusing to figure out what portions are occurring remotely and what portions are occurring locally. Once I learned that the SSH program can run a program other than the shell on a remote system, the scripting concept started to make sense. Hopefully, it does to you too.

Putting It All Together

You now have a system that will notify you when someone mentions your name in a running session of Irssi. I'll assume you run Irssi in a screen or tmux session, so you can connect and disconnect as need be. Because I connect from various machines, that's how I do it. To make things simple, however, I've written a second script that establishes the notification SSH connection and then SSHes me into the server. Then, when I disconnect from the interactive SSH session, it kills off the notification connection too. I call the script from above irc_notify, and the script below simply irc. When I want to use IRC, I type `irc` from the command line, and the notification stuff happens automatically. Here's that "irc" script:

```
#!/bin/bash
# Location: /usr/local/bin/irc
#
```

```
# Kill any existing fnotify sessions that might be
# running to avoid duplicate notifications
kill $(ps aux | grep [f]notify | awk '{print $2}') >
 ➥/dev/null &2>1
irc_notify &
ssh spowers@my.remote.server.com
# Kill all current fnotify sessions, so notifications
# don't pop up while I'm not connected to IRC
kill $(ps aux | grep [f]notify | awk '{print $2}')
```

The script isn't really anything complex, but the kill lines are a little tricky. I run `ps aux` and then grep for "fnotify" to make sure I have the correct SSH session. Unfortunately, when you grep for a particular word, you also get the actual grep process. Using the `[f]notify` regex, only the SSH process is returned. Then, the second field is the process ID (PID), so I use awk to get just that information and kill the process by number.

I have that command twice: once before launching the interactive SSH session to make sure the notification script isn't already running, and then once after I disconnect to clean up after myself. Ideally, the first kill statement never will find anything to kill off, since my script cleans up after itself, but sometimes things go wrong. I like to make sure I don't get duplicate notifications due to multiple fnotify scripts running.

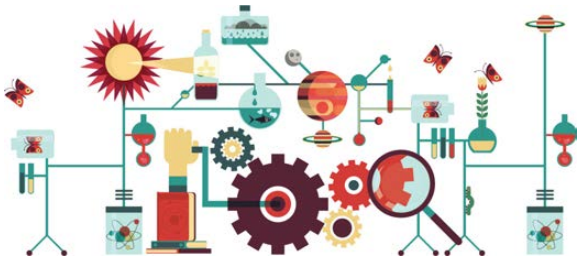
Cross Platform, and Beyond!

I regularly use the scripts on both Linux and OS X. I have no idea if there is something that would do graphical pop-ups on Windows, but I think the scripting part would work using Cygwin. I haven't had a need to test such a thing on Windows, but I'd be interested in hearing about it if you get it working.

I love using Irssi for chatting via IRC, and now that I have a GUI notification system for it, I'll probably start using Bitlbee for other IM protocols, making Irssi truly my one-stop shop for chatting needs. I hope this scripting lesson inspires you to do far more than I've accomplished—with Linux, the only limiting factor is your imagination! ■

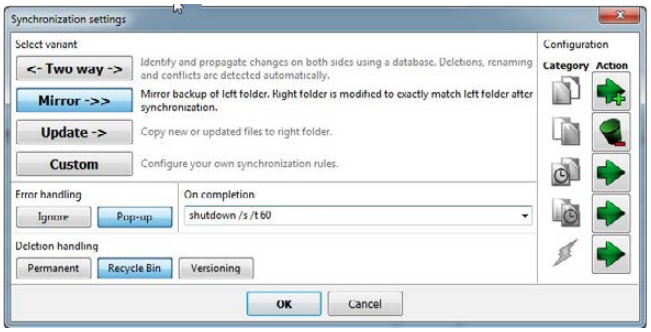
Shawn Powers is the Associate Editor for *Linux Journal*. He's also the Gadget Guy for LinuxJournal.com, and he has an interesting collection of vintage Garfield coffee mugs. Don't let his silly hairdo fool you, he's a pretty ordinary guy and can be reached via e-mail at shawn@linuxjournal.com. Or, swing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

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



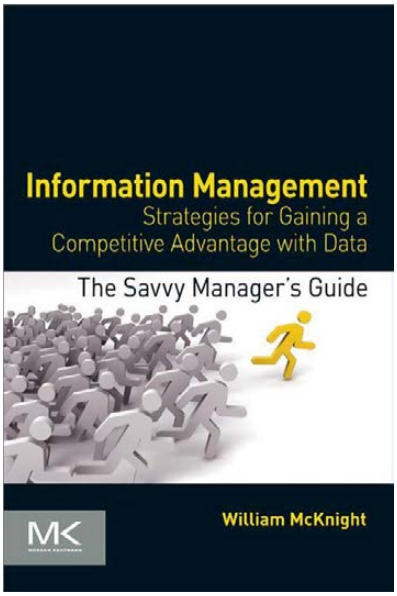
Versal

On-line learning is often one-dimensional, limited to videos and slide decks, says Versal, an interactive on-line course creation and publishing company. To prove that point, Versal has released version 1.5 of the firm's identically dubbed application (yes, Versal), a free and open publishing platform for anyone to create interactive on-line courses—no coding required. Versal's argument is that the combination of computers, together with the modern Internet, are designed for interactivity. The Versal team believes that by partnering with education enthusiasts and developers, together it can solve the biggest problems facing humanity. Versal 1.5 features collaborative authoring, meaning that for the first time, multiple authors can collaborate and communicate with each other in real time to develop transformative on-line courses. The result? Versal says this: "Imagine dozens of physics professors and researchers from around the world working together to author the ultimate guide to the fundamental structure of the universe. Or a network of global non-profit advocates creating a water filtration course to help individuals in developing countries. The possibilities are unlimited."
http://www.versal.com



FreeFileSync

With FreeFileSync, you pay much less—that is, nothing—and get features that usually are found only in the paid versions of competing folder comparison and synchronization software utilities. This benefit, adds FreeFileSync's developers, complements the fact that the software is free, open source and works on Linux, Mac OS X and Windows. FreeFileSync is optimized for both CPU and file I/O performance. On an older PC, the software can scan a hard drive with 200,000 files in less than three minutes. On a new PC with an SSD drive, the same scan takes about ten seconds. With FreeFileSync, files can be synchronized between two computers or between two hard drives on a single machine. It works with local drives, external drives and over a network. FreeFileSync also can copy locked files, perform binary file comparisons and configure the handling of symbolic links. The utility's innovative technology enables synchronization of files that have been moved or renamed. And finally, users can automate their synchronization tasks by running the utility as a batch job.
http://freefilesync.sourceforge.net



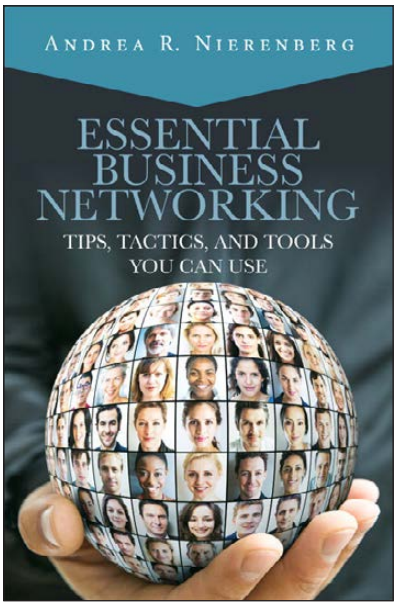
William McKnight's *Information Management* (Elsever/Morgan Kaufmann)

In his new book titled *Information Management: Strategies for Gaining a Competitive Advantage with Data*, tech author William McKnight shows how to take actions that make the most of company information. In the book, published by Elsever/Morgan Kaufmann, McKnight develops the value proposition for information in the enterprise and succinctly outlines the myriad forms of data storage that are now available. The big topics in this practical, hands-on book include data warehousing and the importance of information management and analytics; the technologies and data that advance an organization and extend data warehouses; Big Data, NoSQL and Hadoop; and chapters that unify these strands while addressing topics of agile development, modern business intelligence and organizational change management.
http://www.mkp.com

Janam Technologies' XG Series Mobile Computers

This top gun—Janam Technologies' XG Series gun-style mobile computer—is now more flexible and versatile than ever thanks to the addition of the Linux operating system. The XG Series is a line of rugged mobile computers that scan bar codes and communicate wirelessly. The series was designed for scan-intensive, extended shift use in demanding environments. With its battery in the handle, Janam says that the XG Series line of products is incomparably light and balanced in the hand and, in addition, offers industrial-quality construction and efficiency-enhancing features, such as well-spaced keys and mobile DDR memory. The Series includes the XG100 device with Honeywell's Adaptus imaging technology and the XG105 device with Motorola's SE965 high-performance laser scan engine. Most important, the company adds that the Linux option makes the XG series a highly appealing alternative to bundled commercial systems, particularly for retailers, manufacturers and government agencies.
http://www.janam.com

Andrea Nierenberg's *Essential Business Networking* (Pearson)



The more the world changes, the truer remains the old adage: it's not what you know, it's who you know. Certainly you've got to know something, but you get the point. We Linuxers are indeed quirky, independent individuals, often in need of a gentle push in the form of advice like Andrea Nierenberg presents in the new book *Essential Business Networking: Tips, Tactics, and Tools You Can Use*. Published by Pearson, this book is a simple read, and a selection of the author's top tips are organized into chapters that represent logical steps for building a powerful network. One can start at the beginning for an A to Z coverage of the topic. Or, if readers just want to polish a certain skill, they can skip around to explore a particular networking skill in-depth. The author has digested and synthesized all of the tips and techniques into bite-sized action plans, which allow for immediate implementation.
http://www.informit.com

Compuware Workbench

Compuware
THE TECHNOLOGY PERFORMANCE COMPANY

The rise of Compuware Workbench illustrates that the mainframe is still alive and kicking. Workbench is a modern, intuitive Eclipse-based mainframe development environment that "future-proofs" a company's mainframe assets. The need for Workbench is based on the fact that traditional mainframe environments underpin large volume transactions across a number of businesses, and these systems utilize a complex and antiquated mainframe development environment, with which many newer developers are not familiar. The Workbench gives new developers the tools to produce high-quality applications that drive business success, as well as prevent application outages and other business risks when experienced developers retire and take their knowledge with them. The upgraded Workbench now features significantly more robust data search and editing capabilities, additional debugging support and a code-coverage reporting feature. Workbench also has been more tightly integrated with Compuware APM for Mainframe and the company's other developer productivity solutions.
http://www.compuware.com

Onset's HOBO UX120-006M Analog Logger

In the business of data logging, the name of the game is boosting accuracy and memory while maintaining usability and not breaking the bank. Onsets's newest draft pick, its own LeBron James, if you will, is the HOBO UX120-006M Analog Logger, a device that fulfills the above criteria. The model is a new high-performance, LCD display data logger for building performance monitoring applications. The HOBO UX120-006M provides twice the accuracy of previous models, a deployment-friendly LCD and support for up to four external sensors for measuring temperature, current, CO2, voltage and more. This enables energy engineers, facility managers and others to solve a range of building performance applications easily and affordably, including energy audits, building commissioning studies and equipment scheduling optimization. New deployment-related features, such as visual confirmation of logger operation and battery status, eliminate the need to connect the logger to a computer to view the information. The larger memory capacity means that the logger can store up to 1.9 million measurements, enabling the loggers to be deployed for longer periods between offloads.
http://www.onsetcomp.com

Corsair's Hydro Series H105 Liquid CPU Cooler

Due in part to its innovative tool-free CPU mounting block, an independent source rated Corsair's new Hydro Series H105 Liquid CPU Cooler as "the simplest mounting mechanism we have experienced". Corsair adds that the H105 is the company's first Hydro Series cooler to be equipped with a 38mm-thick 240mm radiator, greatly increasing radiator surface area and improving heat dissipation. Fitted with a pair of high-performance SP120L PWM 120mm fans designed to balance static pressure and noise levels, the H105 pushes liquid cooling performance to the next level in the 240mm category. Connecting the cooling block and 38mm-deep 240mm radiator are a pair of sealed and kink-resistant rubber tubes, with a high-quality on-board ceramic-bearing pump providing reliable flow and heat transfer away from the CPU. The H105 is completely self-contained, requires no maintenance or filling and is supported by a five-year warranty.
http://www.corsair.com

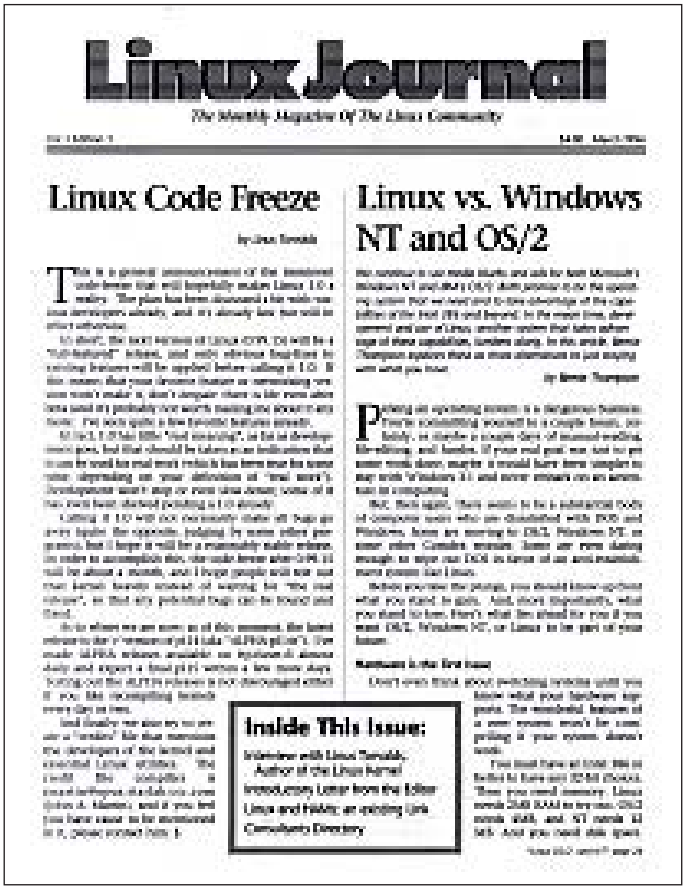
Please send information about releases of Linux-related products to newproducts@linuxjournal.com or New Products c/o *Linux Journal*, PO Box 980985, Houston, TX 77098. Submissions are edited for length and content.

20 YEARS OF LINUX

Linux Journal **issue #1 featured Linus Torvalds' announcement of a code freeze for Linux 1.0, alongside an article comparing Linux to its 32-bit competitors of the day: Windows NT and OS/2. Original author of that article Bernie Thompson stirs the pot again and tastes what the operating system world has been cooking up.**

BERNIE THOMPSON

20 years ago in the world of operating systems, IBM was a dying king and Microsoft an ascendant prince. Apple was in exile. And a young wizard named Linus Torvalds rebuilt venerable UNIX, which already had 20 years of its own history, and re-imagined it in the form of Linux. The world had a powerful new open-source platform on which to build.



Linux Journal

The Monthly Magazine Of The Linux Community

Linux Code Freeze

Linux vs. Windows NT and OS/2

Inside This Issue:

Interview with Linus Torvalds, Author of the Linux kernel

Introductory Letter from the Editor

Linux and WWW: an existing Link

Consultants Directory

Figure 1. Issue #1 of *Linux Journal*

By the end of the 1990s, IBM was nearly irrelevant to the personal computer market it had built. Microsoft had risen to nearly 90% market share (http://windowsitpro.com/systems-management/study-shows-windows-owns-desktop-market). Linux had a disconcerting combination of enormous investor hype and low market share. But Linux quietly was beginning to appear in set-top boxes and networking equipment. It was dominating the Internet server market. And a little company called Google was deploying thousands of Linux boxes to revolutionize search (http://linuxgazette.net/issue59/correa.html).

By the end of the 2000s, at least ten "year of the Linux desktop" milestones had come and gone without success. Windows still was dominant, but the PC era was about to end. Apple had returned from the dead to win the hearts and wallets of consumers and developers by building OS X and iOS on an open-source Mach+BSD kernel with different lineage from Linux. Linux was everywhere and nowhere: the Internet ran on Linux, yet few consumers' browsers did.

Here in 2014, the world seems to be shifting in Linux's direction, as long as a Linux-derived kernel is what counts. Windows 8 has alienated Microsoft's installed base. Mobile is ascendant. Android has reached 80% smartphone share (**http://www.idc.com/getdoc.jsp?containerId=prUS24442013**). And last year, Chromebooks rose to take 10% of the US computer market and 21% of laptop sales, outselling Apple MacBooks (**https://www.npd.com/wps/portal/npd/us/news/press-releases/u-s-commercial-channel-computing-device-sales-set-to-end-2013-with-double-digit-growth-according-to-npd**). Yet the Linux desktop, in the pure form of Ubuntu or one of the other distros, still shuffles along with less than 2% share by most measures.

From Megabytes to Gigabytes

In the first issue of *Linux Journal*, I wrote "Linux needs 2MB RAM to try out, OS/2 needs 4MB, and NT needs 12MB."

Those Linux and OS/2 numbers were for command-line configurations only. The NT number included its GUI since it couldn't operate without it. But the march of Moore's Law over the long stretch is stunning. In less than 20 years, we've seen nearly a 1,000x increase in typical memory sizes.

Today, Windows 8 can run in 512MB for just a single task or two, but 2GB is a normal minimum. Linux is still the most miserly. 256MB is normal for platforms like the Raspberry Pi, and running Linux on a wireless router with 8MB of RAM is not unheard of using DD-WRT micro version.

The original article focused on the issues of "Will Linux, OS/2, or NT work on my PC?" These issues have receded as even sub-$200 PCs today can run anything. The important question has become "What's the best fit for what I need my computer to do?"

The Importance of the OS

A good operating system mediates between users, applications and hardware with a focus on clarity, compatibility and security. It enables applications written today to work on the latest OS version, and a significant portion of the existing installed base of older versions. It enables those applications to keep working even as things underneath change over time. It allows hardware created today to support existing applications and ones not yet written. All it takes is for one essential application or piece of hardware to be unavailable on a new platform, and the user can't move. This creates powerful lock-in to a users' current system.

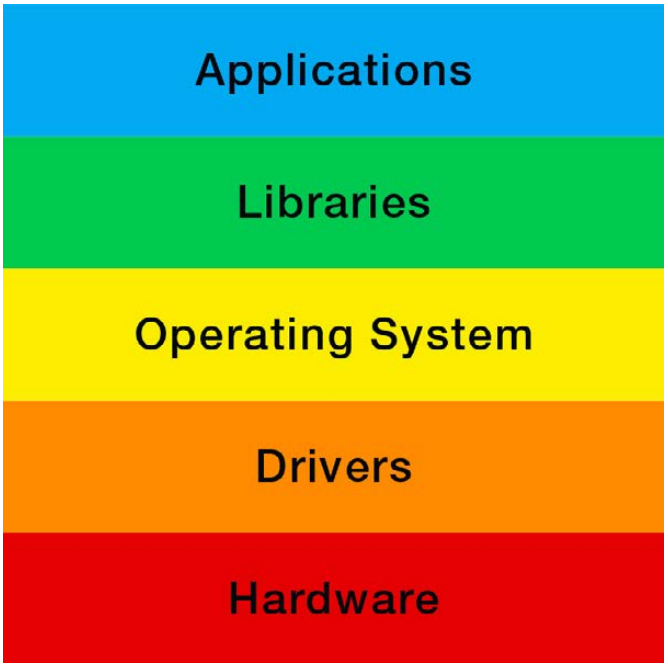


Figure 2. Simplified OS Layers

The OS also provides a UI and set of standard ways for users to get things done. Humans are flexible, but they suffer compatibility problems too. When a new UI moves things around, there's a human cost in time and frustration. When there's a shift in the way computers are used—from command line, to mouse and keyboard GUI, to touch GUI—not every use scenario or user shifts. We all know the command line is still critical for any IT pro. Operating systems must either pick a paradigm or find a way to expose similar functionality in each world.

This isn't easy, which is part of why the OS world has fractured and diversified in the shift to touch GUIs during the past few years. As platforms and UIs have been churned, at times it seems we've needed a Hippocratic Oath for programmers and UI designers: first, do no harm.

And in the realm of security, protecting users and their data has become even harder post-NSA. We've entered a Wild West era where we have ceded moral authority to deter nations, corporations and rogue groups from exploiting others on-line. It's everyone for themselves, and a trusted operating system is at the center of that.

How are the operating systems of today doing by these standards?

Windows has had the largest financial investment in it, by far, during the past 20 years. For the last decade, Microsoft has had thousands of employees working on Windows at any given moment. Executives are obsessed to distraction with avoiding the innovator's dilemma. Many investments are "big bets" that aim to anticipate customer and partner demand, rather than smaller steps to respond to it. Often priorities are focused on internal goals and "better together" initiatives that attempt to extend lock-in or push the user base to the latest version. When those initiatives are aligned with the interests of users and partners, they tend to succeed. But many have not.

Among the unheralded successes has been the Windows Update mechanism. For users, it appears that they plug any new or old device into their Windows PC, and it just works—no digging for disks for downloads. In fact, it's a smart cloud-based system that was implemented before the cloud was a buzzword. When a new device is plugged in, Windows automatically reads the plug-and-play IDs, checks its servers on the Web, and downloads and installs the best available driver automatically. More than ten years on, Linux and Mac OS X have nothing equivalent.

On the downside, the Windows 8 transition has been particularly jarring. Microsoft faced a tough choice: lose market share in the new tablet space or create a version of Windows and an ecosystem of applications that supports tablets well. It chose to sacrifice usability

as a desktop system to win new tablet users. The result is a confusing collision of two UI worlds. Some users have gone scrambling to find an alternative, driving up sales of Chromebooks and perhaps hastening the tablet shift.

Meanwhile, the Windows 8 App Store had a new API and no provision to run those apps on Windows 7 or earlier. This created another catch-22 for winning over application developers. It sped up the unraveling of the valuable application lock-in that Microsoft had established starting with Windows 3.1. A commercial application developer a decade ago would have been crazy not to design for Windows. Today, that same developer is crazy not to look first at designing for a hosted HTML, CSS and JavaScript model that would allow the application to work on any platform. That should be ringing alarm bells in Redmond, given the market share of other Microsoft platforms that weren't able to leverage the lock-in of Windows binaries (Windows CE, Zune, Windows Phone and so on).

Security-wise, it's important to understand that Windows 7 and 8 have perhaps the best line-of-code level security of any existing OS. Microsoft invests enormous efforts to identify and fix potential exploits before shipment. But the weekly "patch Tuesday" flood highlights how difficult it is to protect a big, juicy target like Windows. It's the potato blight of our era. We'll talk about Linux security later, but what protects Linux is less the quality of its code, and more the diversity of it.

After one and a half years, Windows 8 has crossed only 10% market share (**http://arstechnica.com/information-technology/2014/01/windows-8-x-breaks-10-percent-internet-explorer-11-makes-a-splash**). But Microsoft still has a chance to regain its footing, given the combined market shares of XP, 7 and 8.

Mac OS and iOS

Steve Jobs warned Microsoft that trying to merge the desktop and tablet worlds wouldn't work. And so far Apple has largely stuck to that line, with success. Millions of eager buyers are willing to pay a premium for Apple products. iOS has been passed by Android in pure

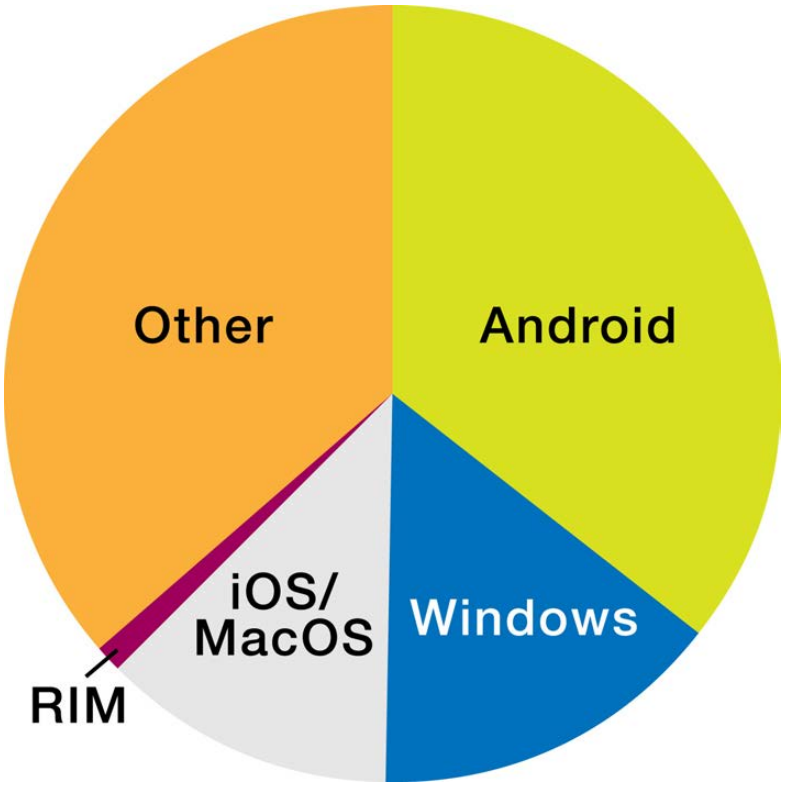


Figure 3.
2013 Device Shipments by Operating System
Source: Gartner. Includes PCs, Tablets, and Phones (http://www.gartner.com/newsroom/id/2408515)

unit sales, but still gets more use day to day.

Mac OS X was my main platform for much of the 2000s. It puts a nice UI on a deeply functional UNIX foundation. The excellent MacPorts system gives entry to the full catalog of open-source applications. Apple's XQuartz project enables even GUI X apps to be ported. Users that care about particular scenarios like working with photos, music and video have a platform with great support for those scenarios, because of Job's attention to end-to-end quality.

Apple has delivered many innovations, but among the most powerful was the successful iOS and later OS X App Stores. Small developers had withered under Microsoft. By enabling software developers to make money, these app stores quickly allowed Apple's application ecosystem to rival Microsoft's.

Apple has historically been more willing to sacrifice the compatibility of older applications and hardware. That might have become a problem as Apple's installed base grew. But the application

problem has been mitigated with the strategy of offering free OS upgrades combined with free application upgrades from the App Store. Your current software binaries won't work several versions from now, but you won't care because you'll have downloaded a free update.

Hardware compatibility, however, often has been sacrificed. One gets the sense that Apple sees a robust hardware ecosystem as competing with it, rather than a source of strength for the platform as a whole.

In security, Apple has been both smart and lucky. Smart to build on UNIX. Smart to introduce strict app store criteria and the Gatekeeper feature to steer users away from untrusted apps. But it also has been lucky to stay under the radar. If Windows and OS X's market shares were reversed, Apple would be forced to have a much higher level of focus on individual exploits, being a monoculture like Windows.

Android and Chrome OS

Google also has kept the tablet and desktop worlds apart somewhat, in the form of Android for tablets and Chrome OS for laptops, both built on the Linux kernel.

Linus' strategy of benevolent dictatorship for the Linux kernel has delivered stable progress through the years and kept the worst decisions out. However, above the kernel, progress on Linux has been unstable and constantly disruptive...mostly to users, if not to competitors.

With Android, Google's role has been to provide that stability above the kernel, along with the opportunity for a for-profit ecosystem of software and hardware to build around the platform. The result has been an amazing explosion of Linux-based devices.

With Chrome OS, Google has consummated the process of making the browser the OS. HTML, CSS and JavaScript are the new terminal to the cloud. For those of us who live on the Web and hosted applications, there is enormous peace of mind in having no local software and few security issues to think about—especially when

we're gifting a laptop to grandpa or our daughter, or if we think we're on the NSA's naughty list.

Because Android and Chrome OS are open source, whether you're Amazon building the Kindle or CyanogenMod, you have the freedom to take it in new directions.

Google has succeeded in hastening the end of the Windows monopoly by grabbing a significant chunk of market share with an open-source operating system based on Linux. All this makes it likely that the next generation of applications will be developed with Web compatibility and platform portability as a primary concern. Even if Android and Chrome OS do not continue their meteoric rise, for the moment they have mitigated much of the lock-in that would have prevented users from moving to the next innovative platform.

The Linux Distributions

The ever-churning cauldron of Linux distros and spins is the hotbed of innovation from which Android, Chrome OS, Kindle, TiVO and countless other innovations past and future are born. On the downside, competing groups innovating in different directions have obvious consequences for some key OS characteristics: clarity and compatibility.

In practice, for hardware that is well documented, the results at the kernel level are generally excellent. If you can get your driver into the kernel, it tends to get carried forward intact. On the other hand, trying to keep a binary driver in sync with the Linux kernel is difficult, expensive and usually futile. Linux's strategy of sacrificing binary compatibility for the freedom to innovate and keep the kernel clean has proven powerful over time.

There are some exceptions, such as graphics, which touches many layers of the stack. The transition to compositing desktops has hurt performance on Linux scenarios where a beefy GPU isn't available. And the more functionally complex KMS/DRM driver model has had many impacts on users. OS features like support for multiple monitors that require participation at many layers have been difficult to achieve.

At the application level, the story is more mixed than the kernel. Competing libraries, versions and package managers make it difficult to port even open-source applications to every distribution. There are many potential libraries to become dependent on, and each dependency may evolve in ways that will demand that your application change or bifurcate to keep up.

Some distros focus on minimizing churn and compromising by providing facilities, such as a commercial app store. Ubuntu and Red Hat Enterprise Linux are best known as playing this role on the desktop. When that plot has been lost, back-to-basics distros like Linux Mint have won converts.

In the cloud, Amazon has a growing list of applications in its marketplace built on Linux (often Red Hat- or Debian-based). You can see Linux's customizability, maintainability and lack of licensing cost at work in how Amazon prices Linux-based hosting vs. Windows hosting. Linux is a more cost-effective choice.

AMAZON m1.small EC2 PRICING LINUX VS. WINDOWS

- **Linux: $0.060/hour**
- **Windows: $0.091/hour**

Can the Linux distributions do a better job for end users? Yes, definitely. Branches like Ubuntu's effort to replace X with Mir should not be taken so lightly. It would be a boon to Linux adoption if there were more efforts to consolidate or combine projects, to present a unified front to applications and end users. Is the universe forever expanding until we're all on isolated islands, or will it consolidate? It's a delicate balance. We could start with a joint development conference between the Canonical and Red Hat teams to find common ground.

On the plus side, Linux's software and hardware diversity make it difficult for mass exploits. And

more fundamentally, open source is essential for trust through transparency. Any individual, nation or corporation can do its own code review to weed out potential exploits. While not a panacea for sophisticated, targeted attacks, these characteristics allow Linux to deliver the best security of any platform in practice.

The Future

Despite the challenges, the future for Linux is unstoppably bright.

Beyond the desktop, Linux leads in servers and has completely conquered the embedded market. It's impossible to conceive of any one company being able to port a commercial OS to as many CPU and device architectures as Linux has. As the embedded world transitions from 8/16-bit processors to 32-bit, billions of new devices will be running Linux inside.

Even today, average Microsoft or Apple executives are probably (knowingly or unknowingly) running several copies of Linux in their homes: perhaps their TV, router, thermostat or the coming generation of appliances and vehicles.

Linux is currently winning the mobile race with Android and grabbing significant "desktop" share with Chrome OS. Efforts such as CyanogenMod, Firefox OS and Ubuntu for phones will keep creating innovative options. And as applications and users increasingly live within the browser, the barriers to move between platforms will continue to fall.

The revolution that Linus started 20 years ago is accelerating. "World domination", which was previously a sly joke, now appears inevitable. Happy birthday, *Linux Journal*.■

Bernie Thompson is a former IBM and Microsoft OS developer, and a Linux kernel contributor. He is the Founder of Plugable Technologies: a Linux-aware USB, Bluetooth and network devices company. You can reach him at bernie@plugable.com.

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

SILICON MECHANICS

SERVERS • STORAGE • HPC

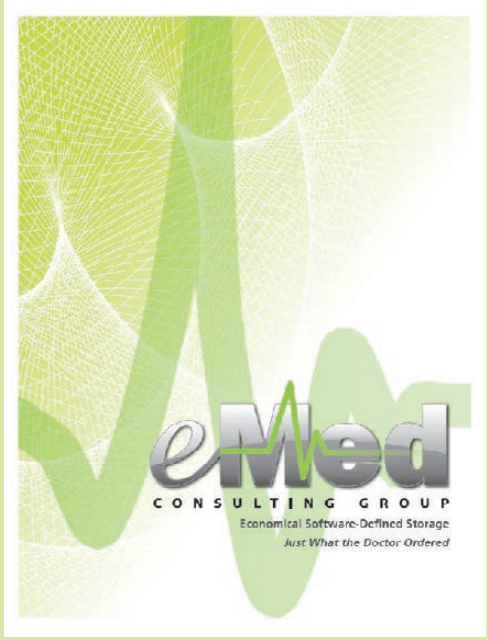
Are you considering software-defined storage?

zStax
ZFS Unified Storage

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, the **zStax StorCore** ZFS unified storage appliances have the right mix of performance, capacity, and reliability to fit your needs.

March Case Study Feature



Economical Software-Defined Storage:
Just What the Doctor Ordered

zStax StorCore 64

zStax StorCore 104

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

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.

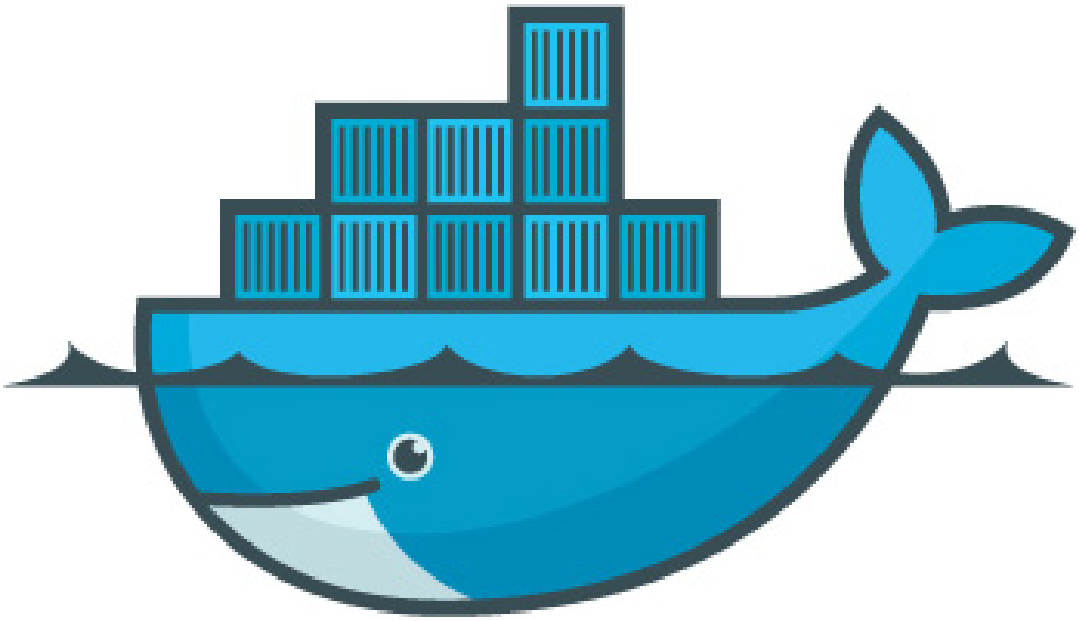
Benefits of an SOE:

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**



Docker: Lightweight Linux Containers for Consistent Development and Deployment

Take on "dependency hell" with Docker containers, the lightweight and nimble cousin of VMs. Learn how Docker makes applications portable and isolated by packaging them in containers based on LXC technology.

DIRK MERKEL

Imagine being able to package an application along with all of its dependencies easily and then run it smoothly in disparate development, test and production environments. That is the goal of the open-source Docker project. Although it is still not officially production-ready, the latest release (0.7.x at the time of this writing) brought Docker another step closer to realizing this ambitious goal.

Docker tries to solve the problem of "dependency hell". Modern applications often are assembled from existing components and rely on other services and applications. For example, your Python application might use PostgreSQL as a data store, Redis for caching and Apache as a Web server. Each of these components comes with its own set of dependencies that may conflict with those of other components. By packaging each component and its dependencies, Docker solves the following problems:

- Conflicting dependencies: need to run one Web site on PHP 4.3 and another on PHP 5.5? No problem if you run each version of PHP in a separate Docker container.

- Missing dependencies: installing applications in a new environment is a snap with Docker, because all dependencies are packaged along with the application in a container.

- Platform differences: moving from one distro to another is no longer a problem. If both systems run Docker, the same container will execute without issues.

Docker: a Little Background

Docker started life as an open-source project at dotCloud, a cloud-centric platform-as-a-service company, in early 2013. Initially, Docker was a natural extension of the technology the company had developed to run its cloud business on thousands of servers. It is written in Go, a statically typed programming language developed by Google with syntax loosely based on C. Fast-forward six to nine months, and the company has hired a new CEO, joined the Linux Foundation, changed its name to Docker Inc., and announced that it is shifting its focus to the development of Docker and the Docker ecosystem. As further indication of Docker's popularity, at the time of this writing, it has been starred on GitHub 8,985 times and has been forked 1,304 times. Figure 1 illustrates Docker's rising popularity in Google searches. I predict that the shape of the past 12 months will be dwarfed by the next

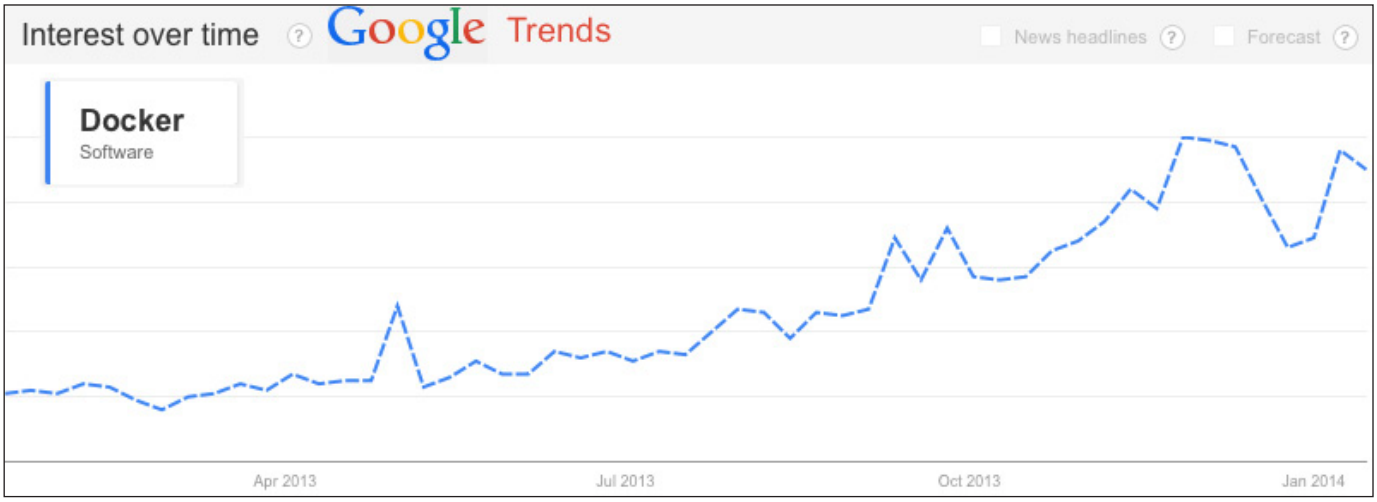


Figure 1. Google Trends Graph for "Docker Software" for Past 12 Months

12 months as Docker Inc. delivers the first version blessed for production deployments of containers and the community at large becomes aware of Docker's usefulness.

Under the Hood

Docker harnesses some powerful kernel-level technology and puts it at our fingertips. The concept of a container in virtualization has been around for several years, but by providing a simple tool set and a unified API for managing some kernel-level technologies, such as LXCs (LinuX Containers), cgroups and a copy-on-write filesystem, Docker has created a tool that is greater than the sum of its parts. The result is a potential game-changer for DevOps, system administrators and developers.

Docker provides tools to make creating and working with containers as easy as possible. Containers sandbox processes from each other. For now, you can think of a container as a lightweight equivalent of a virtual machine.

Linux Containers and LXC, a user-space control package for Linux Containers, constitute the core of Docker. LXC uses kernel-level namespaces to isolate the container from the host. The user namespace separates the container's and the host's user database, thus ensuring that the container's root user does not have root privileges on the host. The process namespace is responsible for displaying and managing only processes running in the container, not the host. And, the network namespace provides the container

with its own network device and virtual IP address.

Another component of Docker provided by LXC are Control Groups (cgroups). While namespaces are responsible for isolation between host and container, control groups implement resource accounting and limiting. While allowing Docker to limit the resources being consumed by a container, such as memory, disk space and I/O, cgroups also output lots of metrics about these resources. These metrics allow Docker to monitor the resource consumption of the various processes within the containers and make sure that each gets only its fair share of the available resources.

In addition to the above components, Docker has been using AuFS (Advanced Multi-Layered Unification Filesystem) as a filesystem for containers. AuFS is a layered filesystem that can transparently overlay one or more existing filesystems. When a process needs to modify a file, AuFS creates a copy of that file. AuFS is capable of merging multiple layers into a single representation of a filesystem. This process is called copy-on-write.

The really cool thing is that AuFS allows Docker to use certain images as the basis for containers. For example, you might have a CentOS Linux image that can be used as the basis for many different containers. Thanks to AuFS, only one copy of the CentOS image is required, which results in savings of storage and memory, as well as faster deployments of containers.

An added benefit of using AuFS is Docker's ability to version container images. Each new version is simply a diff of changes from the previous version, effectively keeping image files to a minimum. But, it also means that you always have a complete audit trail of what has changed from one version of a container to another.

Traditionally, Docker has depended on AuFS to provide a copy-on-write storage mechanism. However, the recent addition of a storage driver API is likely to lessen that dependence. Initially, there are three storage drivers available: AuFS, VFS and Device-Mapper, which is the result of a collaboration with Red Hat.

As of version 0.7, Docker works with all Linux distributions. However, it does not work with most non-Linux operating systems, such as Windows and OS X. The recommended way of using Docker on those OSes is to provision a virtual machine on VirtualBox using Vagrant.

Containers vs. Other Types of Virtualization

So what exactly is a container and how is it different from hypervisor-based virtualization? To put it simply, containers virtualize at the operating system level, whereas hypervisor-based solutions virtualize at the hardware level. While the effect is similar, the differences are important and significant, which is why I'll spend a little time exploring the differences and the resulting differences and trade-offs.

Virtualization: Both containers and VMs are virtualization tools. On the VM side, a hypervisor makes siloed slices of hardware available. There are generally two types of hypervisors: "Type 1" runs directly on the bare metal of the hardware, while "Type 2" runs as an additional layer of software within a guest OS. While the open-source Xen and VMware's ESX are examples of Type 1 hypervisors, examples of Type 2 include Oracle's open-source VirtualBox and VMware Server. Although Type 1 is a better candidate for comparison to Docker containers, I don't make a distinction between the two types for the rest of this article.

Containers, in contrast, make available protected portions of the operating system—they effectively virtualize the operating system. Two containers running on the same operating system don't know that they are sharing resources because each has its own abstracted networking layer, processes and so on.

Operating Systems and Resources: Since hypervisor-based virtualization provides access to hardware only, you still need to install an operating system. As a result, there are multiple full-fledged operating systems running, one in each VM, which quickly gobbles up resources on the server, such as RAM, CPU and bandwidth.

Containers piggyback on an already running operating system as their host environment. They merely execute in spaces that are isolated from each other and from certain parts of the host OS. This has two significant benefits. First, resource utilization is much more efficient. If a container is not executing anything, it is not using up resources, and containers can call upon their host OS to satisfy some or all of their dependencies. Second, containers are cheap and therefore fast to create and destroy. There is no need to boot and shut down a whole OS. Instead, a container merely has to terminate the processes running in its isolated space. Consequently, starting and stopping a container is

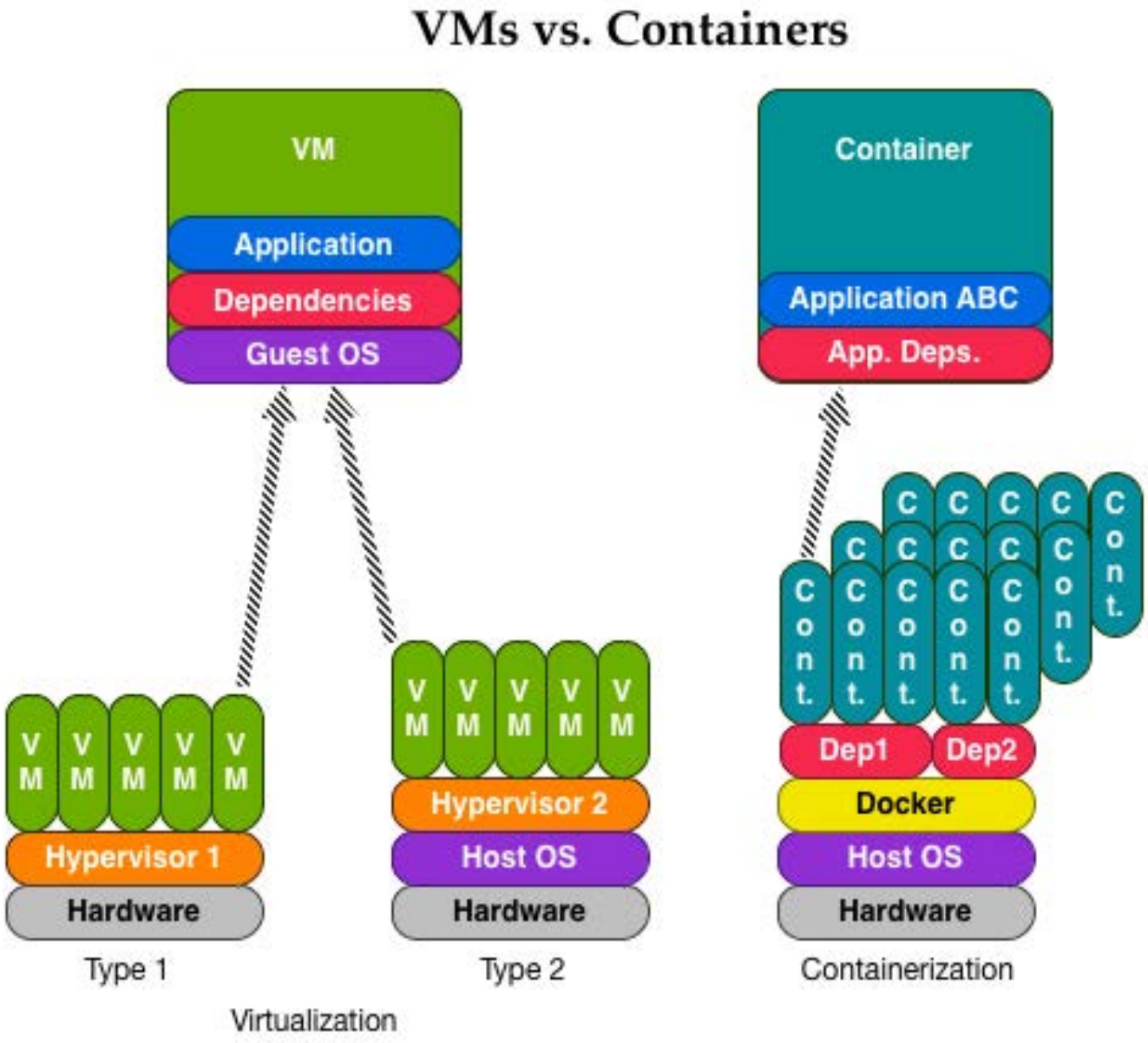


Figure 2. VMs vs. Containers

more akin to starting and quitting an application, and is just as fast.

Both types of virtualization and containers are illustrated in Figure 2.

Isolation for Performance and Security: Processes executing in a Docker container are isolated from processes running on the host OS or in other Docker containers. Nevertheless, all processes are executing in the same kernel. Docker leverages LXC to provide separate namespaces for containers, a technology that has been present in Linux kernels for 5+ years and is considered fairly mature. It also uses Control Groups, which have been in the Linux kernel even longer, to implement resource

auditing and limiting.

The Docker dæmon itself also poses a potential attack vector because it currently runs with root privileges. Improvements to both LXC and Docker should allow containers to run without root privileges and to execute the Docker dæmon under a different system user.

Although the type of isolation provided is overall quite strong, it is arguably not as strong as what can be enforced by virtual machines at the hypervisor level. If the kernel goes down, so do all the containers. The other area where VMs have the advantage is their maturity and widespread adoption in production environments. VMs have been hardened and proven themselves in many different high-availability environments. In comparison, Docker and its supporting technologies have not seen nearly as much action. Docker in particular is undergoing massive changes every day, and we all know that change is the enemy of security.

Docker and VMs—Frenemies: Now that I've spent all this time comparing Docker and VMs, it's time to acknowledge that these two technologies can actually complement each other. Docker runs just fine on already-virtualized environments. You obviously don't want to incur the cost of encapsulating each application or component in a separate VM, but given a Linux VM, you can easily deploy Docker containers on it. That is why it should not come as a surprise that the officially supported way of using Docker on non-Linux systems, such as OS X and Windows, is to install a Precise64 base Ubuntu virtual machine with the help of Vagrant. Simple detailed instructions are provided on the **http://www.docker.io** site.

The bottom line is that virtualization and containers exhibit some similarities. Initially, it helps to think of containers as very lightweight virtualization. However, as you spend more time with containers, you come to understand the subtle but important differences. Docker does a nice job of harnessing the benefits of containerization for a focused purpose, namely the lightweight packaging and deployment of applications.

Docker Repositories

One of Docker's killer features is the ability to find, download and start container images that were created by other developers quickly. The place where images are stored is called a registry, and Docker Inc. offers a

public registry also called the Central Index. You can think of the registry along with the Docker client as the equivalent of Node's NPM, Perl's CPAN or Ruby's RubyGems.

In addition to various base images, which you can use to create your own Docker containers, the public Docker Registry features images of ready-to-run software, including databases, content management systems, development environments, Web servers and so on. While the Docker command-line client searches the public Registry by default, it is also possible to maintain private registries. This is a great option for distributing images with proprietary code or components internally to your company. Pushing images to the registry is just as easy as downloading. It requires you to create an account, but that is free as well. Lastly, Docker Inc.'s registry has a Web-based interface for searching for, reading about, commenting on and recommending (aka "starring") images. It is ridiculously easy to use, and I encourage you to click the link in the Resources section of this article and start exploring.

Hands-On with Docker

Docker consists of a single binary that can be run in one of three different ways. First, it can run as a dæmon to manage the containers. The dæmon exposes a REST-based API that can be accessed locally or remotely. A growing number of client libraries are available to interact with the dæmon's API, including Ruby, Python, JavaScript (Angular and Node), Erlang, Go and PHP.

The client libraries are great for accessing the dæmon programmatically, but the more common use case is to issue instructions from the command line, which is the second way the Docker binary can be used, namely as a command-line client to the REST-based dæmon.

Third, the Docker binary functions as a client to remote repositories of images. Tagged images that make up the filesystem for a container are called repositories. Users can pull images provided by others and share their own images by pushing them to the registry. Registries are used to collect, list and organize repositories.

Let's see all three ways of running the docker executable in action. In this example, you'll search the Docker repository for a MySQL image. Once you find an image you like, you'll download it, and tell the Docker dæmon to run the command (MySQL). You'll do all of this from the

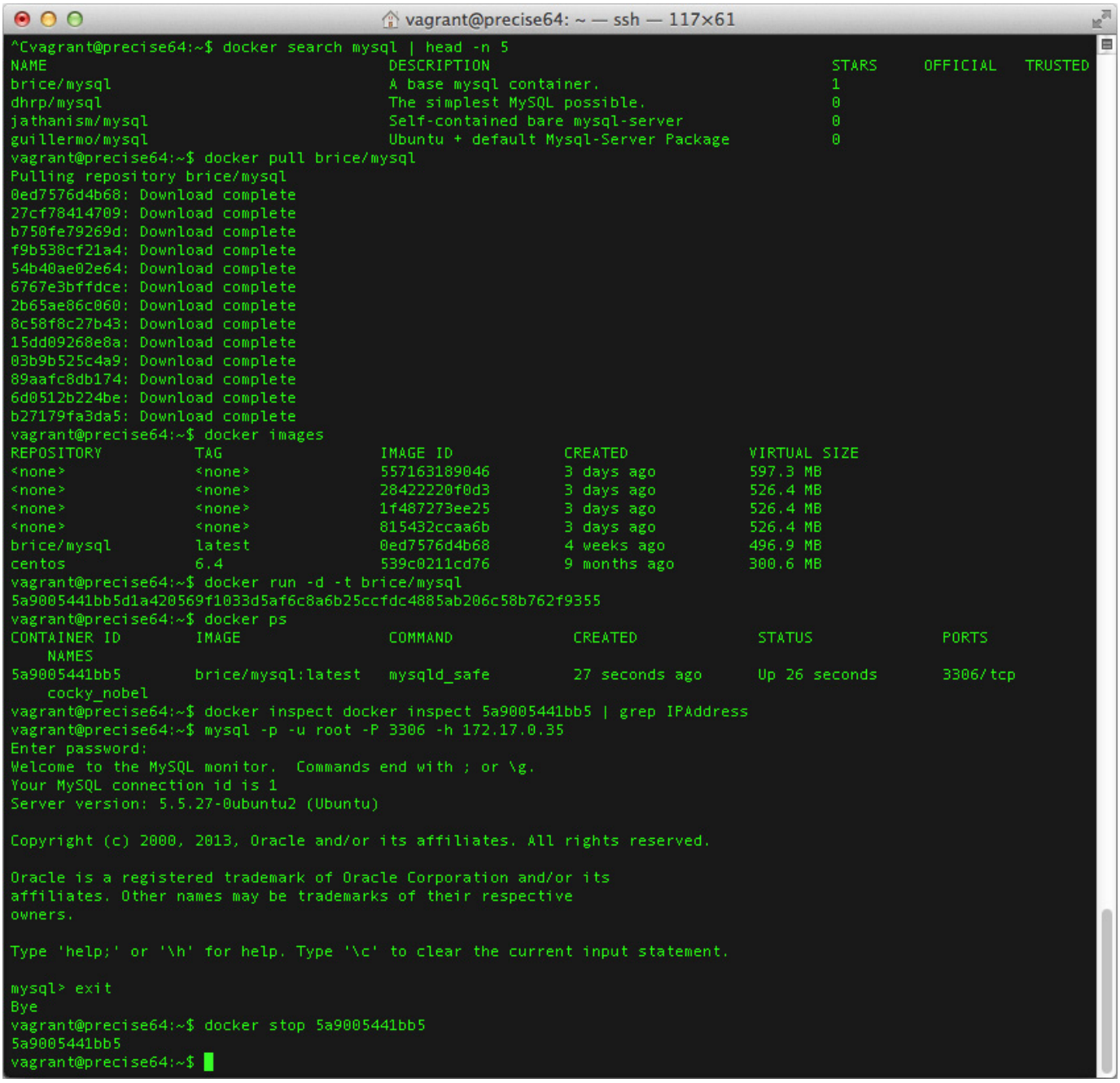


Figure 3. Pulling a Docker Image and Launching a Container

command line.

Start by issuing the `docker search mysql` command, which then displays a list of images in the public Docker registry that match the keyword "mysql". For no particular reason other than I know it works, let's download the "brice/mysql" image, which you do with the `docker pull brice/mysql` command. You can see that Docker downloaded not only the specified image, but also

the images it was built on. With the `docker images` command, you list the images currently available locally, which includes the "brice/mysql" image. Launching the container with the `-d` option to detach from the currently running container, you now have MySQL running in a container. You can verify that with the `docker ps` command, which lists containers, rather than images. In the output, you also see the port on which MySQL is listening, which is the default of 3306.

But, how do you connect to MySQL, knowing that it is running inside a container? Remember that Docker containers get their own network interface. You need to find the IP address and port at which the mysqld server process is listening. The `docker inspect <imageId>` provides a lot of info, but since all you need is the IP address, you can just grep for that when inspecting the container by providing its hash `docker inspect 5a9005441bb5 | grep IPAddress`. Now you can connect with the standard MySQL CLI client by specifying the host and port options. When you're done with the MySQL server, you can shut it down with `docker stop 5a9005441bb5`.

It took seven commands to find, download and launch a Docker container to get a MySQL server running and shut it down after you're done. In the process, you didn't have to worry about conflicts with installed software, perhaps a different version of MySQL, or dependencies. You used seven different Docker commands: search, pull, images, run, ps, inspect and stop, but the Docker client actually offers 33 different commands. You can see the full list by running `docker help` from the command line or by consulting the on-line manual.

Before exercising Docker in the above example, I mentioned that the client communicates with the dæmon and the Docker Registry via REST-based Web services. That implies that you can use a local Docker client to interact with a remote dæmon, effectively administering your containers on a remote machine. The APIs for the Docker dæmon, Registry and Index are nicely documented, illustrated with examples and available on the Docker site (see Resources).

Docker Workflow

There are various ways in which Docker can be integrated into the development and deployment process. Let's take a look at a sample workflow illustrated in Figure 4. A developer in our hypothetical company might be

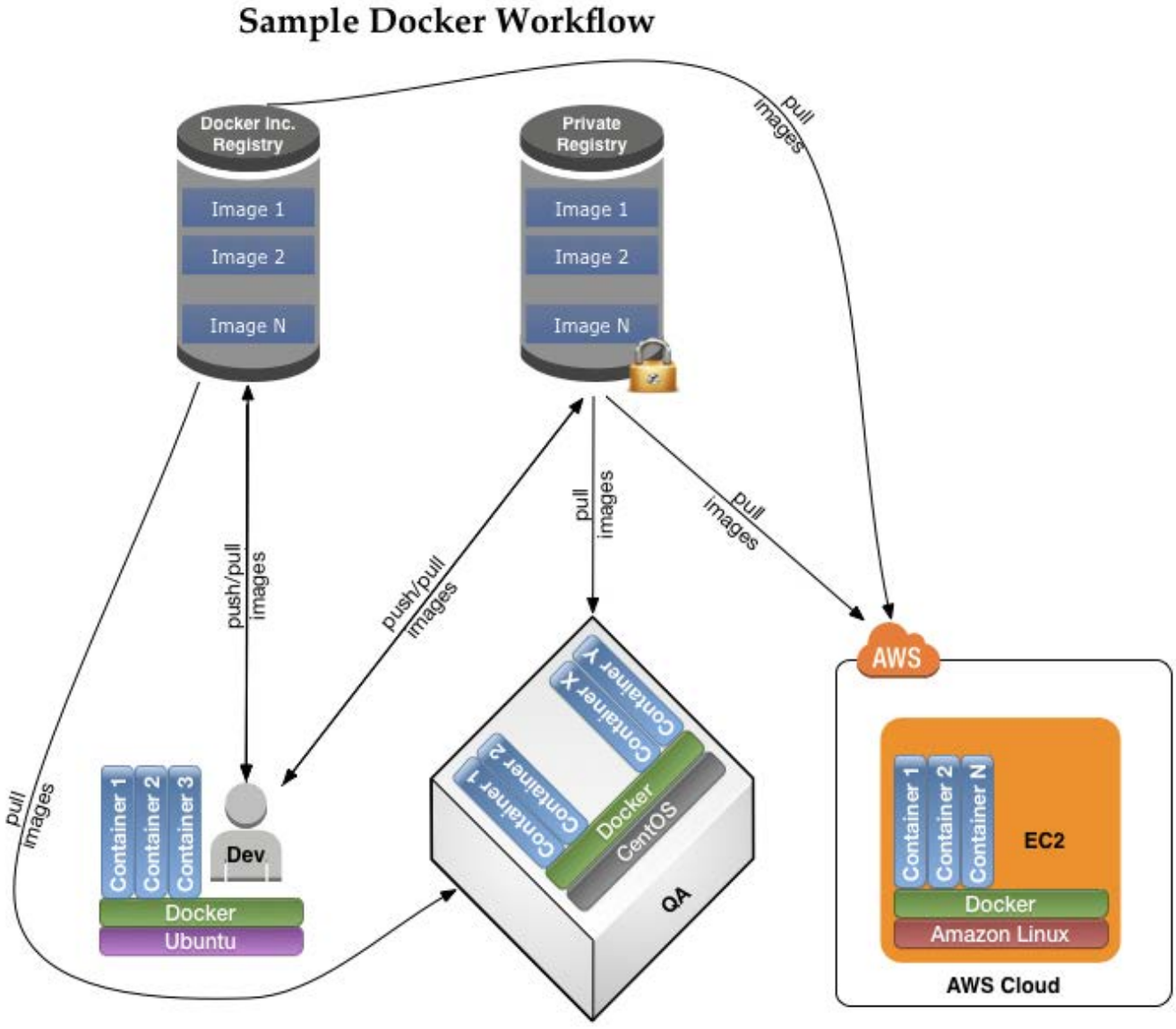


Figure 4. Sample Software Development Workflow Using Docker

running Ubuntu with Docker installed. He might push/pull Docker images to/from the public registry to use as the base for installing his own code and the company's proprietary software and produce images that he pushes to the company's private registry.

The company's QA environment in this example is running CentOS and Docker. It pulls images from the public and private registries and starts various containers whenever the environment is updated.

Finally, the company hosts its production environment in the cloud, namely on Amazon Web Services, for scalability and

elasticity. Amazon Linux is also running Docker, which is managing various containers.

Note that all three environments are running different versions of Linux, all of which are compatible with Docker. Moreover, the environments are running various combinations of containers. However, since each container compartmentalizes its own dependencies, there are no conflicts, and all the containers happily coexist.

It is crucial to understand that Docker promotes an application-centric container model. That is to say, containers should run individual applications or services, rather than a whole slew of them. Remember that containers are fast and resource-cheap to create and run. Following the single-responsibility principle and running one main process per container results in loose coupling of the components of your system. With that in mind, let's create your own image from which to launch a container.

Creating a New Docker Image

In the previous example, you interacted with Docker from the command line. However, when creating images, it is far more common to create a "Dockerfile" to automate the build process. Dockerfiles are simple text files that describe the build process. You can put a Dockerfile under version control and have a perfectly repeatable way of creating an image.

For the next example, please refer to the "PHP Box" Dockerfile (Listing 1).

Let's take a closer look at what's going on in this Dockerfile. The syntax of a Dockerfile is a command keyword followed by that command's argument(s). By convention, command keywords are capitalized. Comments start with a pound character.

The FROM keyword indicates which image to use as a base. This must be the first instruction in the file. In this case, you will build on top of the latest CentOS base image. The `MAINTAINER` instruction obviously lists the person who maintains the Dockerfile. The `RUN` instruction executes a command and commits the resulting image, thus creating a new layer. The `RUN` commands in the Dockerfile fetch configuration files for additional repositories and then use Yum to install curl, git, wget, unzip, httpd, php-mysql and yum-utils. I could have combined the `yum install` commands into

Listing 1. PHP Box

```
# PHP Box
#
# VERSION 1.0

# use centos base image
FROM centos:6.4

# specify the maintainer
MAINTAINER Dirk Merkel, dmerkel@vivantech.com

# update available repos
RUN wget http://dl.fedoraproject.org/pub/epel/6/x86_64/
➥epel-release-6-8.noarch.rpm; rpm -Uvh epel-release-6-8.noarch.rpm

# install some dependencies
RUN yum install -y curl git wget unzip

# install Apache httpd and dependencies
RUN yum install -y httpd

# install PHP and dependencies
RUN yum install -y php php-mysql

# general yum cleanup
RUN yum install -y yum-utils
RUN package-cleanup --dupes; package-cleanup --cleandupes;
 ➥yum clean -y all

# expose mysqld port
EXPOSE 80

# the command to run
CMD ["/usr/sbin/apachectl", "-D", "FOREGROUND"]
```

a single RUN instruction to avoid successive commits.

The EXPOSE instruction then exposes port 80, which is the port on which Apache will be listening when you start the container.

Finally, the CMD instruction will provide the default command to run when the container is being launched. Associating a single process with the launch of the container allows you to treat a container as a command.

Typing `docker build -t php_box .` on the command line will now tell Docker to start the build process using the Dockerfile in the current working directory. The resulting image will be tagged "php_box", which will make it easier to refer to and identify the image later.

The build process downloads the base image and then installs Apache httpd along with all dependencies. Upon completion, it returns a hash identifying the newly created image. Similar to the MySQL container you launched earlier, you can run the Apache and PHP image using the "php_box" tag with the following command line: `docker run -d -t php_box`.

Let's finish with a quick example that illustrates how easy it is to layer on top of an existing image to create a new one:

```
# MyApp
#
# VERSION       1.0

# use php_box base image
FROM php_box

# specify the maintainer
MAINTAINER Dirk Merkel, dmerkel@vivantech.com

# put my local web site in myApp folder to /var/www
ADD myApp /var/www
```

This second Dockerfile is shorter than the first and really contains only two interesting instructions. First, you specify the "php_box" image as a starting point using the `FROM` instruction. Second, you copy a local directory to the image with the `ADD` instruction. In this case, it is a PHP project that is being copied to Apache's DOCUMENT_ROOT folder on the images. The result is that the site will be served by default when you launch the image.

Conclusion

Docker's prospect of lightweight packaging and deploying of applications and dependencies is an exciting one, and it is quickly being adopted by the Linux community and is making its way into production environments. For example, Red Hat announced in December support for Docker in the upcoming Red Hat Enterprise Linux 7. However, Docker is still a young project and is growing at breakneck speed. It is going to be exciting to watch as the project approaches its 1.0 release, which is supposed to be the first version officially sanctioned for production environments. Docker relies on established technologies, some of which have been around for more than a decade, but that doesn't make it any less revolutionary. Hopefully this article provided you with enough information and inspiration to download Docker and experiment with it yourself.■

Dirk Merkel has experience architecting solutions and managing the software development process in large and small organizations. His focus is on open source and often Web-centric technologies, including Java, PHP, Perl, Ruby, MySQL, Apache and so on. He is the CTO for Vivantech Inc.

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

Resources

Main Docker Site: **https://www.docker.io**
Docker Registry: **https://index.docker.io**
Docker Registry API: **http://docs.docker.io/en/latest/api/registry_api**
Docker Index API: **http://docs.docker.io/en/latest/api/index_api**
Docker Remote API: **http://docs.docker.io/en/latest/api/docker_remote_api**

DOCKER UPDATE

As this article was being published, the Docker team announced the release of version 0.8. This latest deliverable adds support for Mac OS X consisting of two components. While the client runs natively on OS X, the Docker dæmon runs inside a lightweight VirtualBox-based VM that is easily managed with boot2docker, the included command-line client. This approach is necessary because the underlying technologies, such as LXC and name spaces, simply are not supported by OS X. I think we can expect a similar solution for other platforms, including Windows.

Version 0.8 also introduces several new builder features and experimental support for BTRFS (B-Tree File System). BTRFS is another copy-on-write filesystem, and the BTRFS storage driver is positioned as an alternative to the AuFS driver.

Most notably, Docker 0.8 brings with it many bug fixes and performance enhancements. This overall commitment to quality signals an effort by the Docker team to produce a version 1.0 that is ready to be used in production environments. With the team committing to a monthly release cycle, we can look forward to the 1.0 release in the April to May timeframe.

The Growing Role of UEFI Secure Boot in Linux Distributions

The increasing use of UEFI Secure Boot in Linux-based distributions delivers heightened security to a wide range platforms and devices.

MARK DORAN

With the increasing prevalence of open-source implementations and the expansion of personal computing device usage to include mobile and non-PC devices as well as traditional desktops and laptops, combating attacks and security obstacles against malware is a growing priority for a broad community of vendors, developers and end users. This trend provides a useful example of how the flexibility and standardization provided by the Unified Extensible Firmware Interface (UEFI) technology addresses shared challenges in ways that help bring better products and experiences to market.

The UEFI specification defines an industry-leading interface between the operating system (OS) and the platform firmware, improving the performance, flexibility and security of computing devices. Designed for scalability, extensibility and interoperability, UEFI technology streamlines technological evolution of platform firmware. In 2013, developers of several open-source Linux-based operating systems, including Ubuntu 12.10, Fedora 18 and OpenSUSE 12.3, began using UEFI specifications in their distributions.

Additional features of UEFI include improved security in the pre-boot mode, faster booting, support of drives larger than 2.2 Terabytes and integration with modern 64-bit firmware device drivers. UEFI

standards are platform-independent and compatible with a variety of platform architectures—meaning, users of several different types of operating systems, including both Linux and commercial systems, can enjoy the benefits of UEFI. Equally, because the UEFI specification includes bindings for multiple CPU architectures, these benefits apply on a variety of hardware platforms with these operating systems.

While UEFI Secure Boot may be one of the most talked about features, the complete set of features in the UEFI specification provide a standardized interoperable and extensible booting environment for the operating system and pre-boot applications. The attributes of this environment make it ideal for increased use in a rapidly widening array of Linux-based distributions. UEFI specifications are robust and designed to complement or even further advance Linux distributions. Industry experts expect to see continued expansion of their use during 2014 and beyond.

UEFI Secure Boot in Linux-Based Distributions

Malware developers have increased their attempts to attack the pre-boot environment because operating system and antivirus software vendors have hardened their code. Malware hidden in the firmware is virtually untraceable by the operating system, unless a search specifically targets malware within the firmware. UEFI Secure Boot assists with system firmware, driver and software validation. UEFI Secure Boot also allows users of Linux-based distributions to boot alternate operating systems without disabling UEFI Secure Boot. It provides users with the opportunity to run the software of their choice in the most secure and efficient manner, while promoting interoperability and technical innovation.

Secure Boot is an optional feature of the UEFI specification. The choice of whether to implement the feature and the details of its implementation (from an end-user standpoint) are business decisions made by equipment manufacturers. For example, consider the simplest and most usual case in which a platform uses UEFI-conformant firmware and a UEFI-aware operating system. When this system powers on (assuming it has UEFI Secure Boot enabled), the UEFI firmware uses security keys stored in the platform to validate the bootloader read from the disk. If the bootloader signature does not match the signature key needed for verification, the system will not boot.

In general, the signature check will succeed because the platform owner will have purchased the system with pre-installed software set up by the manufacturer to pre-establish trust between the firmware and operating system. The signature check also will succeed if the owner has installed an operating system loader that is trusted along with the appropriate keys that represent that trust if those keys are not already present in the platform. The case in which the signature check fails is most likely to arise when untrusted malware has insinuated its way into the machine, inserting itself into the boot path and tampering with the previously installed software. In this way, UEFI Secure Boot offers the prospect of a hardware-verified, malware-free operating system bootstrap process that helps improve system deployment security.

Without UEFI Secure Boot, malware developers can more easily take advantage of several pre-boot attack points, including the system-embedded firmware itself, as well as the interval between the firmware initiation and the loading of the operating system. The UEFI specification promotes extensibility and customization of security-enhanced interfaces, but allows the implementers to specify how they are used. As an optional feature, it is up to the platform manufacturer and system owner to decide how to manage UEFI Secure Boot. Thus, implementations may vary in how they express policy, and of course, UEFI Secure Boot is no panacea for every type of malware or security vulnerability. Nevertheless, in a variety of implementations that have already reached the market, UEFI Secure Boot has proven to be a practical and useful tool for improving platform integrity and successfully defending the point of attack for a dangerous class of pre-operating system malware.

The broadened adoption of UEFI Secure Boot technology, particularly by the Linux community, is not only a movement toward innovation, but also a progressive step toward the safeguarding of emerging computer platforms. The evolution of firmware technology in a variety of sectors continues to gain momentum, increasing the use of UEFI technology in Linux and commercial systems. This is a testament to the cross-functionality of UEFI between devices, software and systems, as well as its ability to deliver next-generation technologies for nearly any platform.

Disabling UEFI Secure Boot in Open-Source Implementations

A variety of models has emerged for the use of UEFI Secure Boot in the Open Source community. The minimal approach is to use the ability to disable the feature—a facility that is present in practically all platforms that implement UEFI Secure Boot. In so doing, the platform owner makes the machine compatible with any operating system that the platform supports regardless of whether that operating system supports UEFI Secure Boot. The downside of taking this approach is giving up the protection that having the feature enabled affords the platform, in terms of improved resistance to pre-operating system malware.

There are a couple key points to understand about the ability to enable or disable Secure Boot in any platform. The UEFI specification leaves both the choice of whether to implement Secure Boot—as well as the choice to provide an "on/off switch"—up to system designers. Practical considerations usually make appropriate choices obvious, depending on the intended use of the product. For example, a system designed to function as a kiosk that has to survive unattended by the owner in a retail store environment would likely choose to lock down the software payload as much as practical to avoid unintended changes that would compromise the kiosk's basic function. If the kiosk runtime booted using UEFI Secure Boot, it may make sense to provide no means to disable the feature as part of the strategy for maximizing kiosk availability and uptime.

General-purpose compute platforms present a different dynamic. In these cases, there is an expectation in the marketplace that the owner's choice of one or more operating systems can be installed on the machine, regardless of what shipped from the factory. For manufacturers of this class of systems, the choice of whether to allow enabling/disabling of UEFI Secure Boot takes into consideration that their customers want to choose from any available operating system, given than some may include no support for UEFI Secure Boot. This is true for open source as well as commercial operating system support. A vendor building a machine that supports all the operating system offerings from Microsoft's catalog, for example, must support older versions that have no UEFI Secure Boot support, as well as the newer ones from the Windows 8 generation that do have such support. Indeed,

the need for the enable/disable feature appears in Microsoft's own platform design guide as a mandatory requirement, ensuring that conforming systems can run back catalog products as well as the newest products.

Following the same line of reasoning, most general-purpose platforms are shipping with not only the enable/disable feature, but also with facilities for the platform owner to manage the key store. This means owners can remove pre-installed keys, and in particular, add new ones of their own choosing. This facility then provides the basis for those who choose to roll their own operating system loader images, such as self-signing, or to select an operating system loader signed by the CA of their choice, regardless of whether or not the appropriate keys shipped from the factory.

In some cases, the creators of Linux distributions have chosen to participate directly in the UEFI Secure Boot ecosystem. In this case, a distribution includes an operating system loader signed by a Certificate Authority (CA). Today, the primary CA is the UEFI CA hosted by Microsoft, which is separate but parallel to the CA used for Microsoft's own software product management. At the time of this writing, no other CA has offered to participate; however, the UEFI Forum would welcome such an offer, as having a second source of supply for signing events would be ideal.

In other cases, Linux distributions provide users with a general-purpose shim-bootloader that will chain boot to a standard, more complete Linux bootloader in a secure manner. This process extends the chain of trust from UEFI Secure Boot to the Linux system environment, in which it becomes the province of the operating system-present code to determine what, if anything, to do with that trust.

Linux-Based Platforms that Leverage UEFI Secure Boot

The past year has marked the implementation of UEFI specifications in three popular Linux-based operating systems: Ubuntu 12.10, Fedora 18 and OpenSUSE. Below are additional details about their use of UEFI standards.

Canonical's Ubuntu 12.10

Support for a base-level compatibility between Canonical's Ubuntu and UEFI firmware began in October 2012, with the releases of 12.10 64-bit and 12.04.2 64-bit. At the time of release, industry experts projected that most machines would ship with a firmware

compliant with version 2.3.1 of the UEFI standard. Currently, all Ubuntu 64-bit versions now support the UEFI Secure Boot feature. When deployed in Secure Boot configurations, the Ubuntu boot process uses a small "boot shim", which allows compatibility with the third-party CA.

Fedora 18 The UEFI Secure Boot implementation in Fedora 18 prevents the execution of unsigned code in kernel mode and can boot on systems with Secure Boot enabled. Fedora also boots on UEFI systems that do not support or have disabled Secure Boot. The bootloaders can run in an environment in which the boot-path validation process takes place without UEFI. In this mode, there are no restrictions on executing code in kernel mode. In Fedora 18, UEFI Secure Boot extends the chain of trust from the UEFI environment into the kernel. The verification process takes place before loading kernel modules.

OpenSUSE 12.3 The recent establishment of UEFI as the standard firmware on all x86 platforms was a milestone for the Open Source community, specifically for OpenSUSE.

OpenSUSE 12.2 included support for UEFI, and the recent OpenSUSE 12.3 provides experimental support for the Secure Boot extension.

The Linux Community's Increasing Use of UEFI Technology for Security Solutions on Next-Generation Platforms

The increased reliance on firmware innovation across non-traditional market segments, combined with the expansion of personal computing from traditional desktops and laptops to an ever-wider range of form factors, is changing the landscape of computing devices. Although mobile devices have traditionally had a custom, locked-down environment, their increasing versatility and the growing popularity of open-source operating systems brings growing vulnerability to complex security attacks. While UEFI Secure Boot cannot unilaterally eradicate the insurgence of security attacks on any device, it helps provide a cross-functional solution for all platforms using UEFI firmware—including Linux-based distributions designed for tablets, smartphones and other non-PC devices. Currently, no one has claimed or demonstrated an attack that can circumvent UEFI Secure Boot, where properly implemented and enabled. The expansion of UEFI technologies into the Linux space addresses the growing demand for security, particularly across the mobile and non-PC application continuum.

What's Next for UEFI Technology in Linux-Based Applications?

As UEFI specifications continue to enable the evolution of firmware technology in a variety of sectors, their use will continue to gain momentum. In addition, the popularity and proliferation of

Key Features of UEFI

- Support of a more secure system, across multiple interfaces.
- Faster boot times.
- Speedier time to market.
- Extensibility, modularity and easy prototyping during development.
- UEFI specifications allow developers to reuse code during the building process, promoting more efficiency.

Linux-based distributions will create even greater demand for UEFI technology. The recent use of UEFI specifications in Linux-based operating systems, such as Ubuntu 12.10, Fedora 18 and OpenSUSE 12.3, underscores this trend.

These distribution companies, along with the Linux Foundation and a number of other thought-leading groups from the Open Source community, are now members of the UEFI Forum. This is an important step forward for the ecosystem as a whole, improving innovation and collaboration between the firmware and operating system communities. For example, as mentioned above, many systems include facilities to self-manage the key stores in the platform, but today, limited potential for automating this exists. Proposals from the Open Source community address this limitation, with the promise of significant simplification for installing open-source operating systems in after-market scenarios. By providing a venue where discussion of such proposals reaches the ears of all the right stakeholders, the UEFI Forum helps speed up the arrival of such solutions in the market. This is exactly the kind of innovation and collaboration that the UEFI Forum is eager to foster.

The increasing deployment of UEFI technology in both Linux and commercial systems is a testament to its ability to deliver next-generation technologies for nearly any platform. A growing number of Linux distributions use UEFI specifications, allowing users to launch virtually any operating system of their choosing, while still enjoying the added security benefits of UEFI Secure Boot. With the expansion of UEFI specifications across numerous platforms, its intended purpose—to streamline and aid in firmware innovation by promoting interoperability between software, devices and systems—is realized.■

Mark Doran is an Intel Fellow and the chief platform software architect within the System Software Division for the Software and Services Group at Intel Corporation. As lead architect for the Unified Extensible Firmware Interface (UEFI) program, Doran develops industry standards-based firmware for Intel architecture systems. Doran also serves as president of the UEFI Forum, a world-class non-profit industry standards body that works through industry collaboration to enable the evolution of platform technologies and to advocate a standardized interface that simplifies and secures platform initialization and firmware bootstrap operations.

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

Two-Factor Authentication System for Apache and SSH

If you have personal publicly accessible Web sites and/or publicly accessible SSH services, you should take steps to limit your risks by adding a simple, yet effective two-factor solution.

JAMES LITTON

If you run a publicly accessible Web server for your own use (and let's face it, if you're reading *Linux Journal*, there's a very good chance you do), how do you go about limiting the risk of someone accessing your site and doing bad things? How about SSH, an even bigger concern? In today's world, it's imperative to think about your exposure and take steps to limit as much risk as possible.

In this tutorial, I walk through the steps necessary to implement a home-grown two-factor authentication system for accessing your Web sites and for SSH access.

The Infrastructure and the "Challenge"

Running your own hardware can be a pain in the neck. After dealing with hardware failures, such as failed fans, failed power supplies, bad hard disks and the like, you finally may decide to dump your co-lo or bedroom closet and your hardware and jump into the world of elastic computing. One such option is Amazon's EC2 platform, which offers a variety of Linux flavors and has one of the most robust and mature cloud platforms available. I'm not an Amazon representative, but I'm the first to say *try it*. It's amazing stuff, and a micro instance is free for a year.

In the test scenario for this article, I use an Amazon EC2 server running Ubuntu 12.04 LTS to host a couple Web applications. If you use a different flavor of Linux, the

It turns out you easily can set up your own, homegrown, two-factor solution and use it to control access to your Web apps and SSH, while also making it possible to allow occasional access to your sites by other users.

instructions easily can be adapted to meet your specific needs. Let's assume the applications are, for the most part, for personal use only. If the sites were accessed only from work or home, you simply could secure the sites by creating firewall rules to allow Web traffic from only those IP addresses. This, incidentally, is exactly how one should secure SSH.

Let's assume though that this won't work for your Web apps because you do a fair amount of traveling and need to be able to access those applications while you're on the road, so a couple firewall rules won't help you. Let's also assume that your applications have their own security systems, but you still want an extra layer of security.

You could have set up a VPN server, but every once in a while, you might like to give a family member access to one of your sites, so a VPN approach wouldn't work.

Another consideration is Google Authenticator for true two-factor authentication. You certainly could go down this path, but you're looking for something you can do yourself—something that is self-contained and yours.

Just like so many things in the Linux world, where there's a will, there's a way! It turns out you easily can set up your own, homegrown, two-factor solution and use it to control access to your Web apps and SSH, while also making it possible to allow occasional access to your sites by other users.

Apache Authentication and Authorization

Since the Web server for this example is Apache, let's leverage the server's authentication and authorization capabilities to ask for a set of credentials before any of your sites are served up to a user.

In the interest of keeping things simple, and since you will follow best practice and allow only https traffic to and from your Web server,

let's use the `mod_auth_basic` module for authentication.

Start by becoming root and installing Apache on your fresh Ubuntu install:

```
sudo su
apt-get install apache2
```

Let's assume your Web applications run in subfolders off of the main www document folder. This allows you to take care of all your sites at once by creating a single .htaccess file in the http server root folder:

```
vim /var/www/.htaccess
```

Now, let's add a few lines that tell Apache to require authentication and where to look for the password file:

```
AuthType Basic
AuthName "restricted area"
AuthUserFile /home/ubuntu/.htpasswd
require valid-user
```

With that in place, you now need to change the ownership of the file so the Apache process can read its contents:

```
chown www-data:www-data /var/www/.htaccess
```

Next, you need to create the .htpasswd file that you reference in your .htaccess file and configure its ownership so the Web server can read it:

```
htpasswd -cb /home/ubuntu/.htpasswd jameslitton test123
chown www-data:www-data /home/ubuntu/.htpasswd
```

Now you need to tell Apache to require authentication and to use the `mod_auth_basic` module for that purpose:

```
vim /etc/apache2/sites-available/default-ssl
```

Then you need to change `AllowOverride None` to `AllowOverride AuthConfig`:

```
Service apache2 restart
```

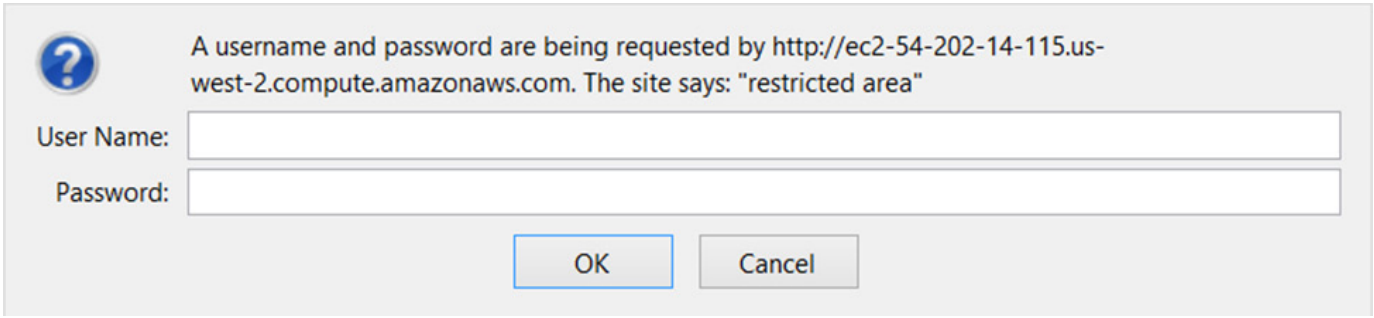


Figure 1. Authentication Request from `mod_auth_basic`

Visiting your site now prompts for a user name and password (Figure 1).

One-Time Day Password/PIN

The approach I'm going to take here is to have your secondary authentication password change daily instead of more frequently. This allows the `mod_auth_basic` approach described above to work. I won't go into the details here, but suffice it to say that every time the password changes, an immediate re-authentication is required, which is not the behavior you want.

Let's go with a six-digit numeric pin code and have that delivered to a mobile phone at midnight every night. I'm a big fan of Pushover, which is a service that pushes instant notifications to mobile phones and tablets from your own scripts and application.

To set this up, create a bash script:

```
vim /home/ubuntu/2fac.sh
```

Now add the following lines:

```
1  #!/bin/bash
2  ppwd=`od -vAn -N4 -tu4 < /dev/urandom | tr -d '\n' | tail -c 6`
3  curl -s -F "token=id" -F "user=id" -F "message=$ppwd"
   ➥https://api.pushover.net/1/messages.json
4  htpasswd -b /home/ubuntu/.htpasswd jameslitton $ppwd
5  echo $ppwd | base64 >/home/ubuntu/.2fac
```

Line 2 produces a random six-digit PIN code and assigns it to a variable called ppwd. Line 3 sends the PIN to the Pushover service for delivery to your mobile phone. Line 4 updates the .htpasswd file with the new password, and last but not least, Line 5 stores a copy of the PIN in a format that you can recover, as you will see later on.

Now save the script, and make it executable:

```
chmod +x /home/ubuntu/2fac.sh
```

To complete this solution, all you need to do is schedule the script to run, via cron, at midnight each night:

```
crontab -e
00 00 * * * /home/ubuntu/2fac.sh
```

Making It Web-Accessible

You certainly could leave it there and call it done, but suppose you didn't receive your code and want to force a change. Or, perhaps you gave someone temporary access to your site, and now you want to force a password change to ensure that that person no longer can access the site. You always could SSH to your server and manually run the script, but that's too hard. Let's create a Web-accessible PHP script that will take care of this for you.

To start, change the ownership of your 2fac.sh script so your Web server can run it:

```
chown www-data:www-data /home/Ubuntu/2fac.sh
```

Now you need to create a new folder to hold your script and create the PHP script itself that allows a new "key" to be run manually:

```
mkdir /vaw/www/twofactor
vim /var/www/twofactor/index.php
```

```
1  <?php
2  exec('/home/ubuntu/2fac.sh');
3  header('Location: http://www.google.com');
4  ?>
```

Because it's conceivable that you're needing to force a new key because you didn't receive the previous one, you need to make sure the folder that holds this script does not require authentication. To do that, you need to modify the Apache configuration:

```
vim /etc/apache2/sites-available/default-ssl
```

Now add the following below the Directory directive for /var/www:

```
<Directory /var/www/twofactor/>
        satisfy any
</Directory>
```

Now let's configure ownership and restart Apache:

```
chown -R www-data:www-data /var/www/twofactor
Service apache2 restart
```

So thinking this through, it's conceivable that the Pushover service could be completely down. That would leave you in a bad situation where you can't access your site. Let's build in a contingency for exactly this scenario.

To do this, let's build a second script that grabs a copy of your PIN (remember the .2fac file that you saved earlier) and e-mails it to you. In this case, let's use your mobile carrier's e-mail to SMS bridge to SMS the message to you.

Start by installing mailutils if you haven't done so already, and be sure to select the Internet option:

```
apt-get install mailutils
```

Now create the second script:

```
vim /home/Ubuntu/2fac2.sh
```

Then add the code:

```
#!/bin/bash
ppwd=`cat /home/ubuntu/.2fac | base64 --decode`
echo " " | mail -s $ppwd xxx5551212@vtext.com
```

Don't forget to change the file's ownership:

```
chown www-data:www-data /home/ubuntu/2fac2.sh
chown www-data:www-data /home/ubuntu/.2fac
```

With that out of the way, now you need to modify the PHP script:

```
vim /var/www/twofactor/index.php
```

Replace line 2 with the following:

```
2  if (isset($_GET["sms"])) {
3    exec('/home/ubuntu/2fac2.sh');
4    } else {
5    exec('/home/ubuntu/2fac.sh');
6    }
```

Then create two bookmarks, so that any time you want to generate a new PIN and have it sent to you via Pushover, you simply can click the link and it's done. The second bookmark will send a copy of the existing PIN to the e-mail address of your choice in the unlikely event that the Pushover service is unavailable.

- 2Factor = https://www.thelittonfamily.com/twofactor/index.php
- 2Factor—SMS = https://www.thelittonfamily.com/twofactor/index.php?sms=1

Extending to SSH

Extending this solution to cover SSH is really pretty simple. The key is to use the little-known `ForceCommand` directive in your sshd_config file. This forces the SSH dæmon to run a script before spanning the terminal session.

Let's start with the script:

```
vim /home/ubuntu/tfac-ssh.sh
```

Now add the following lines:

```
1  #!/bin/bash
2  code=`cat .2fac | base64 --decode`
3  echo -ne "Enter PIN: "
4  while IFS= read -r -s -n1 pass; do
5    if [[ -z $pass ]]; then
6       echo
7       break
8    else
9       echo -n '*'
10      input+=$pass
11   fi
12 done
13 if [ $code = $input ];
14 then
15   sleep 1
16   clear
17   /bin/bash
18 else
19   sleep 1
20   curl -s -F "token=id" -F "user=id" -F "message=$input"
      ➥https://api.pushover.net/1/messages.json
21 fi
```

Figure 2. Two-Factor Request from SSH

Line 2 loads the PIN into a variable. Lines 3–12 prompt for the PIN and echo a star back for each key press. Line 13 compares the user's input to the PIN. If they match, lines 14–17 clear the screen and start a bash session. If the user's input does not match the PIN, lines 18–21 send a notification to Pushover so you know a failure occurred and then ends the session.

Let's configure the SSH dæmon to run the script:

```
vim /etc/ssh/sshd_config
```

Now add the following to the top of the file:

```
ForceCommand /home/ubuntu/tfac-ssh.sh
```

This approach works great. The only limitation is no backspaces. If you press the wrong key, your session will be terminated, and you'll have to try again.

There you have it, a poor-man's two-factor authentication implementation with very little effort and from my experience, it's rock solid!■

James Litton is an entrepreneur, business owner and a regular contributor to various Linux-related periodicals.

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

EOF

DOC SEARLS

Our Assignment

We need to protect the freedoms in which Linux was born and grew up.

I've been with *Linux Journal* since it was a gleam in Phil Hughes' eye, back in 1993. Phil's original plan was for something he called "a free software magazine". I was one of the friends Phil recruited to think and talk, mostly by by e-mail, about how to make the magazine happen. The project was pretty far downstream when Phil sent the whole thing sideways with five words: "There's this kid from Finland...." That was the first I'd heard of Linus, or of Linux. But Phil was one of the world's experts on UNIX (having fathered many UNIX publications in previous years), and he was convinced that Linux was exactly the free operating system the world was waiting for. He was right.

And so *Linux Journal* was born, in March 1994, just as Linux itself arrived at version 1.0. Its first Editor in Chief was Bob Young, who knew almost nothing about Linux when Phil recruited him. ("He was selling circuit boards or something from a booth in the back of a tradeshow", Phil said.) Not long after that, Bob left to start a Linux company of his own, called Red Hat.

The first piece I wrote for *Linux Journal* was an interview with Craig Burton for an insert called *Websmith*. Craig sought me out, because he wanted to alert the Linux folks to LDAP, which he said was throwing a monkey wrench into Microsoft's plans to do for networked directories what it had done for desktop operating systems. Craig was right, and the wrench worked.

I started writing full time for *LJ* in 1998, covering the open-sourcing of Netscape's browser (now known as Firefox) and the creation of its new parent, Mozilla.org. This coincided with the birth of the open-source movement and the dot-com explosion,

And so *Linux Journal* was born, in March 1994, just as Linux itself arrived at version 1.0.

for which Linux itself was ground zero. The biggest IPOs of 1999 (a record IPO year) were Red Hat, Andover (which had earlier acquired Slashdot) and VA Linux (which later acquired Andover). *Linux Journal* also had offers at the time to sell out, but Phil turned them down. If he had said yes, some of us (especially Phil) would have scored big, but *Linux Journal* would have been long gone by now.

Back then, most of the staff worked in our Seattle headquarters. I worked remotely from California or wherever else I happened to be, and would fly in every couple months for meetings and work, and enjoyed it totally. We hit all the Linux tradeshows, plus O'Reilly's OSCon and Emerging Tech conferences. *LJ* was hot stuff, and so were our advertisers, most of which were venture-funded dot-com players. Then, when the crash hit in 2000, most of those advertisers vanished, leaving nobody to bill, sue or get on the phone. And then, after the attacks on 9/11/2001, companies everywhere dropped all kinds of discretionary expenses, including travel and advertising, and that hit us hard too. (I got an extra whammy by losing every one of the speaking gigs that were lined up at the time.)

Then, over the next few years, the Web became a "content delivery platform" for literally millions of blogs, sites that were "publishers" in name only, and "social networks", such as Facebook, Twitter, Instagram, Tumblr and the rest. So, while *Linux Journal* eagerly covered open-source CMSes (content management systems) like Drupal and WordPress, it also had to compete in a world filled with abundant low-grade "content" or worse: editorial matter scraped and republished, often without attribution, from legitimate sources (including *Linux Journal*), just to game Google and other advertising systems. That *Linux Journal* is still alive, and thriving, is testimony to amazing leadership and fortitude by Carlie Fairchild (our Publisher), Jill Franklin (our Executive Editor) and everybody else on our masthead (**http://www.linuxjournal.com/staff**).

But that's just us. There are bigger battles going on that I want to take this anniversary opportunity to talk about. One is over the future of journalism. The other is over

the future of the Internet as an environment for developing and deploying Linux and other open platforms like it. Both are battles over the same organizing principle.

First, journalism.

For the past year and a half, I've been a visiting scholar at the Arthur L. Carter Journalism Institute at NYU (**http://journalism.nyu.edu**). My main work there has been helping professor Jay Rosen (who stars on Twitter as @JayRosen_NYU) with Studio 20 (**http://studio20nyu.tumblr.com**), which is defined as "a consultancy that gets paid in problems" (**http://www.flickr.com/photos/docsearls/11416036656/in/set-72157638756349696**). In academic terms, the class is clinical: students work on real-world problems with real-world publishers. During my time there, Studio 20 students have consulted *Fast Company*, *Pro Publica*, *The Wall Street Journal*, *Quartz*, *Pando Daily*, *Syria Deeply*, *ABC News*, *TimeOut New York*, *Atavist*, *Al Jazeera America*, *DFM Thunderdome* and others (**http://studio20nyu.tumblr.com/post/50351221259/studio-20s-networked-reporting-project**). Here are a few of the learnings I've gathered over the course of that time, both from the class and from experience with *Linux Journal* and other media:

1. The future of journalism is what Jay Rosen calls *networked* (**http://pressthink.org/2013/05/designs-for-a-networked-beat**), Andrew Leonard calls *open-source* (**http://www.salon.com/1999/10/08/geek_journalism**), and Dan Gillmor calls *We the Media* (**http://www.authorama.com/we-the-media-1.html**). Everybody with a stake in the output contributes input. While this comprises a kind of model, it is far from being a complete system. The media between individual contributors—telephony, texting, e-mail, blogging and postings on "social" and other media—are all fluxy and provisional. You use whatever works today, which might not be what worked yesterday or will work tomorrow. The old system was as solid and vertical as an office building—and mostly happened inside of buildings filled with paid staff working on finished pieces for publishing or airing at specific dates and times. The new system is all scaffolding, all the way down, with very few people getting paid for the work they do.

2. Most professional journalists who had work when the Web was born are out

The portfolio of helpful skills for journalism today goes far beyond writing, photography, video and the ability to dig into a subject.

of the business, or under-employed within it. Many work for little or no money. Crazy as it may seem, they keep working because they believe the world needs what they do. (Rollo May once wrote that writers differ from other artists in this one significant respect: they suffer the illusion that the world really needs to hear what they have to say.) In this respect, journalists are a lot like Linux kernel hackers.

3. The portfolio of helpful skills for journalism today goes far beyond writing, photography, video and the ability to dig into a subject. I was amazed to hear new graduate students in journalism, when asked what skills they bring to the table, say stuff like "Ruby, Python and PHP". When we went around the room introducing ourselves on the first day of Studio 20, I heard only one student say "writing" and one say "investigative reporting". The rest all talked about their skills with software tools and services.

4. "Direct response" advertising is all the rage in digital media (**http://wfoa.wharton.upenn.edu/perspective/docsearls**). This stuff might be called advertising, but it is instead directly descended from direct mail, better known as junk mail. It works on the assumption that surveillance-fed "big data" mills can give individuals an ideally personalized "advertising experience". For whatever good it does (such as keeping publishers alive), it is also why the most popular browser extensions and add-ons are ones that block ads and tracking. It also models spying for the NSA. In *The Intention Economy* (**http://www.amazon.com/The-Intention-Economy-Customers-Charge/dp/1422158527**), I call it a bubble, and I stand by that claim. If you want to know more about why it is doomed, read Don Marti (**http://zgp.org/~dmarti**), our former Editor in Chief, who is doing the world's deepest and most prophetic thinking and writing on the topic (**http://zgp.org/~dmarti/business/#.UtwhOKWtv0E**).

5. The subscription model is stronger than ever. Interesting fact: when *Linux Journal* went all-digital (dropping print), we lost a few subscribers but gained many more. This was, and remains, a Good Thing. But it's also one aspect of another thing....

6. We no longer own our stuff. We just have limited rights to use it. This is true of music, movies, books and countless apps on mobile phones and tablets. In a *de facto* sense, it's even true of much of the hardware we depend on, every day. Look at what it actually says in the terms and conditions (**http://www.tacma.com**) you accept when you buy your smartphone or tablet, and the software that runs on it—even the stuff that costs nothing. Also look at who is snarfing up your usage data and what's being done with it. (If you can tell at all. Most of it is behind walls you can't penetrate.) Then consider this fact: there are few generic white-box phones or tablets. The existence of those in the computer market made both Linux and the Internet possible.

7. The commercial world has turned into a forest of silos. Every "loyalty program", every subscription system (ours included), and every Web site and service that requires its own login and password is a silo. Every one of these silos exists for the convenience of those who maintain it, and every one compounds the inconveniences suffered by the individuals who need to maintain countless separate "relationships" contained inside all these silos. Implicit in this "system" is the assumption that a captive customer or user is more valuable than a free one.

Next, the Internet.

I've probably written more about the Internet in *Linux Journal* than about any other topic. Two of my longest and most cited pieces were both titled "Saving the Net". The first was in June 2003 (**http://www.linuxjournal.com/article/6989**), and the second was in November 2005 (**http://www.linuxjournal.com/article/8673**). Here's an excerpt from the first:

> The Net's problem, from telco and cable industries' perspective, is it was born without a business model. Its standards and protocols imagine no coercive regime to

require payment—no metering, no service levels, no charges for levels of bandwidth. Worse, it was designed as an end-to-end system, where all the power to create, distribute and consume are located at the ends of the system and not in the middle. In the words of David Isenberg (**http://www.isen.com**), the Internet's innards purposefully were kept "stupid". All the intelligence properly belonged at the ends. As a pure end-to-end system, the Net also was made to be symmetrical. It wasn't supposed to be like TV, with fat content flowing in only one direction.

The Net's end-to-end nature is so severely anathema to cable and telco companies that they have done everything they can to make the Net as controlled and asymmetrical as possible. They want the Net to be more like television, and to a significant degree, they've succeeded. Most DSL and cable broadband customers take it for granted that downstream speeds are faster than upstream speeds, that they can't operate servers out of their houses and that the only e-mail addresses they can use are ones that end with the name of their telephone or cable company.

And why not? These companies "own" the Net, don't they? Well, no, they don't. They only "provide" it—critical difference.

Eleven years later, those companies are now solving that problem by shifting billing for Internet services from a single monthly subscription rate to billing for data traffic (the telco model) and for content (the cable TV model). They're implementing both gradually moving television to the Net—and turning the Net into mostly-TV in the process. Their thinking goes like this:

1. The Net and TV are both just screens. In "Report: 90% Of Waking Hours Spent Staring At Glowing Rectangles" (**http://www.theonion.com/articles/report-90-of-waking-hours-spent-staring-at-glowing,2747**), *The Onion* writes, "staring blankly at luminescent rectangles is an increasingly central part of modern life. At work, special information rectangles help men and women silently complete any number of business-related tasks, while entertainment rectangles—larger

So, what does this mean for Linux? Won't we still be able to write code, submit patches, participate in lists and so on?

and louder and often placed inside the home—allow Americans to enter a relaxing trance-like state after a long day of rectangle-gazing." So the distinction between watching TV and using the Net hardly matters.

2. If you have a cable or satellite subscription, you can already watch many networks—or all of them (we use Dish Anywhere)—on your laptop or hand-held. Likewise, you can watch lots of stations and networks, also with subscriptions, delivered via IP, the Internet Protocol.

3. Captive lawmakers have kindly allowed their overlords in the content and transport industries to verticalize entertainment supply chains, making it easy to shift TV from cable and satellite to the Net, while keeping the existing billing systems intact and opening opportunities for many more. Susan Crawford unpacked this nicely in *Captive Audience: The Telecom Industry and Monopoly Power in the New Gilded Age* (**http://yalepress.yale.edu/book.asp?isbn=9780300153132**).

4. Bandwidth is naturally scarce, and it costs a lot to build out infrastructure, especially for mobile devices that can suck down 4K video (**http://en.wikipedia.org/wiki/4K_resolution**) over cellular connections. Why not price the offerings accordingly?

And that's how Hollywood will finish body-snatching the Internet.

So, what does this mean for Linux? Won't we still be able to write code, submit patches, participate in lists and so on?

Sure, but what else? What options will be foreclosed in a system that's run by Hollywood and its allies at Apple, Google, Facebook and Microsoft—and the only software available for most people will be on the shelves of company stores?

In a word, *freedom*. In *Linux Journal*'s bones is a belief that free software, free hardware and free people are more valuable—to

themselves and to the world—than captive ones. We believe in openness too, but freedom is the deeper, more essential and more personal virtue. Freedom is embodied in the GPL v2 license (**http://opensource.org/licenses/GPL-2.0**) that Linus chose for Linux at the start, and which I am sure is one reason Linux succeeded to degrees other OSes can only envy. Freedom is also in the hearts of Linux kernel hackers, and many Linux developers and users.

But that population is a shrinking minority among professionals working with Linux, simply because Linux is a huge success, and has enlarged the general talent pool. That's why you see billboards yelling "**Do You Know Linux? WE ARE HIRING!**" From the perspective of Linux 1.0, this is a dream come true. Yet knowing Linux isn't the same as sharing the values that made Linux kick butt. For some perspective on what's happening here, let's revisit "A Tale of Three Cultures" (**http://www.linuxjournal.com/article/5912**), which I wrote for the March 2002 issue of *Linux Journal*. In it, I described what I saw as three different overlapping constituencies, each with their own cultures (Figure 1).

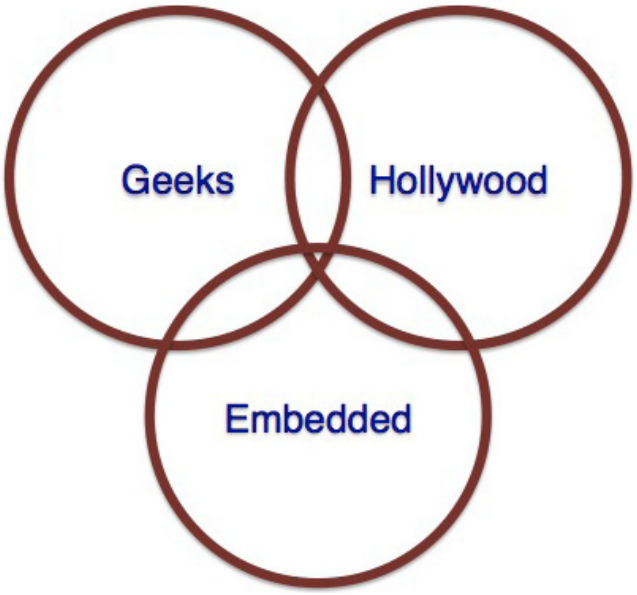


Figure 1. Three Cultures

Lawrence Lessig drew the Geeks/Hollywood distinction in the June 16, 2002 *eWeek*, where he wrote, "There's a civil war brewing in my state of California. It is again a war between the Silicon Valley-based IT industry in the North and Hollywood content and entertainment producers in the South. Silicon Valley has become the target of punitive legislation being pushed by Hollywood in Congress" (**http://www.eweek.com/c/a/Midmarket/Hollywood-v-Silicon-Valley-Make-New-Code-Not-War**). In August of that year, he said this to the geeks assembled at an O'Reilly conference (**http://www.oreillynet.com/pub/a/policy/2002/08/15/lessig.html**):

- *Creativity and innovation always builds on the past.*

- *The past always tries to control the creativity that builds upon it.*
- *Free societies enable the future by limiting this power of the past.*
- *Ours is less and less a free society.*

The title of Larry's talk was "Free Culture", same as the book (**http://www.free-culture.cc**) he was writing at the time. He was fighting against the expansion of copyright law at the time. We (and he) lost that fight, because Hollywood controls Washington. But never mind that. Mind instead the word *culture*. That's what we're talking about. We're kinda Libertarian in our approach to technology, and we'd rather not screw around with "policy", as it's known in academic circles.

The third culture is Embedded. Here's what I said about it in *A Tale of Three Cultures*:

> I was at the Embedded Systems Conference in San Francisco (**https://web.archive.org/web/20020803003421/http://www.esconline.com/sf**), which seemed a world away in time. It was a big conference, filling most of both the North and South Halls at the Moscone Center. Linux was huge here. Domination of the embedded world looks no less inevitable, in spite of the huge Microsoft and Wind River booths, which stood out like boulders in the stream of history.
>
> [Embedded is] purely technical. It's pre-Net, pre-UNIX and maybe even pre-cultural. It shows up where raw technology meets the real world, and its concerns are utterly practical. "Here's the problem", it says. "Let's solve it." This is a heads-down culture and civilization depends on it. Embedded systems are what run our cash registers and brake systems, our airplane guidance systems, our factory robotics, our flow meters, our stoplights and our heating systems. The Net and Linux are both handy ways to solve countless embedded systems problems—extremely handy, it turns out. One morning at SXSW I read that embedded Linux will soon run in something like 60,000 cash registers at Home Depot. It's a big story, but mostly a technical one. Does Home Depot give a damn about Linux as a cause? Or about the lawmaking that threatens to turn the Net into nothing more than a backbone for industrial-grade commerce, plus

> a bunch of culverts for moving "content" stamped and sanitized by ubiquitous digital content management? I kind of doubt it.

Good as it is, and much as we celebrate its success, Android is an embedded Linux operating system. It is also run by one giant company. The Android source FAQ (**http://source.android.com/faqs.html**) makes that quite clear. Android is Google's show. To contrast that with Linux, dig what Andrew Morton (**http://en.wikipedia.org/wiki/Andrew_Morton_%28computer_programmer%29**) told me a few years back (**http://www.linuxjournal.com/content/linux-now-slave-corporate-masters**):

> Look for example at the IBM engineers that do work on the kernel. They understand (how it works) now. They are no longer IBM engineers that work on the kernel. They're *kernel* developers that work for IBM. My theory here is that if IBM management came up to one of the kernel developers and said "Look, we need to do that", the IBM engineer would not say, "Oh, the kernel team won't accept that." He'd say "WE won't accept that." Because now they get it. Now they understand the overarching concern we have for the coherency and longevity of the code base.

One point here is that kernel developers are autonomous individuals who work for the kernel, not for any one company—even if that company employs them to work on the kernel. The same goes for the people we call "users". The Net, by design, supports autonomy, independence and freedom for everybody. Protocols such as HTTP, FRP, IRC, NNTP, POP, SMTP and IMAP all give individuals their own way of connecting with and communicating with anybody or anything, outside any one company's or government's controlling systems. Those all embodied principles I call NEA: Nobody owns it, Everybody can use it, and Anybody can improve it. Linux is that way too. It's only natural for companies operating in the Net's wide open commons to try enclosing it. Usually this fails. (Read Greg Kroah-Hartman's take on what Apple's doing with Thunderbolt, **http://www.kroah.com/log/blog/2013/06/20/hardware**, for a perfect example of what these big companies never seem to learn.) But I'm not sure about Hollywood. It won the battle that Larry Lessig outlined

14 years ago. And now Linux geekery is highly diluted by its embedded uses and the corporate purposes of embedded development work. The tragedy of the Internet's commons is one where free and open geek culture is losing to the cultures of Hollywood and embedded development, and the expediencies of both.

But the cause of freedom got a huge lift from Edward Snowden's revelations about NSA spying (**http://en.wikipedia.org/wiki/Edward_Snowden**). That was a wake-up call, and the world has not fallen asleep since then. In fact, the world is now more awake than ever to the high level of surveillance going on in the digital world, and its threats to freedom. So, while the sun shines, we can make hay.

A year before the Snowden revelations, Eben Moglen (**http://en.wikipedia.org/wiki/Eben_Moglen**) gave a keynote at Freedom to Connect (**http://freedom-to-connect.net/2012**) followed by a conversation onstage with Isaac Wilder (**http://thefnf.org/people-2**) and myself. After Eben walked from podium to chair, I said what he gave was not only one of the best speeches I had ever heard, but one of the most important. I stand by that claim today. You can find audio at Archive.org (**https://archive.org/details/EbenMoglensFreedomToConnectKeynoteInnovationUnderAusterity**), video on YouTube (**https://www.youtube.com/watch?v=G2VHf5vpBy8**), and a transcript at the Software Freedom Law Center (**http://www.softwarefreedom.org/events/2012/freedom-to-connect_moglen-keynote-2012.html**). The talk was 47 minutes long and addressed to the same audience I am writing for here: people with a deep and abiding interest in free and open software and hardware. Here is my abridged hack of the transcript:

> The greatest technological innovation of the late 20th century is the thing we now call the World Wide Web. An invention less than 8000 days old. That invention is already transforming human society more rapidly than anything since the adoption of writing....
>
> The browser made the Web very easy to read. Though we built Apache, though we built the browsers, though we built enormous numbers of things on

top of Apache and the browsers, we did not make the Web easy to write. So a little thug in a hooded sweatshirt made the Web easy to write, and created a man in the middle attack on human civilization, which is unrolling now to an enormous music of social harm. But that's the intermediary innovation that we should be concerned about. We made everything possible including, regrettably, PHP, and then intermediaries for innovation turned it into the horror that is Facebook.

If we'd had a little bit more disintermediated innovation, if we had made running your own Web server very easy, if we had explained to people from the very beginning how important the logs are—and why you shouldn't let other people keep them for you—we would be in a rather different state right now.

We created the idea that we could share operating systems and all the rest of the commoditizable stack on top of them. We did this using the curiosity of young people. That was the fuel, not venture capital. What we need to say is that that curiosity of young people could be harnessed because all of the computing devices in ordinary day-to-day use were hackable.... This is happening now elsewhere in the world as it happened in the United States in the 1980s. Hundreds of thousands of young people around the world hacking on laptops. Hacking on servers. Hacking on general-purpose hardware available to allow them to scratch their individual itches, technical, social, career and just plain ludic itches. "I wanna do this, it would be neat." Which is the primary source of the innovation that drove all of the world's great economic expansion in the last ten years. All of it. Trillions of dollars of electronic commerce....It should embolden us to point out once again that the way innovation really happens is that you provide young people with opportunities to create on an infrastructure that allows them to hack the real world, and share the results.

The nature of the innovation established by Creative Commons, by the Free Software Movement, by Free Culture, which is reflected

in the Web, in Wikipedia, in all the Free Software operating systems now running everything, even the insides of all those locked-down vampiric Apple things I see around the room. All of that innovation comes from the simple process of letting the kids play and getting out of the way. Which, you are aware, we are working as hard as we can to prevent now completely. Increasingly, all around the world, the actual computing artifacts of daily life for human individual beings are being made so you can't hack them. The computer science laboratory in every 12-year-old's pocket is being locked-down....If you prevent people from hacking on what they own themselves, you will destroy the engine of innovation from which everybody is profiting.

We said from the beginning that Free Software is the world's most advanced technical educational system. It allows anybody anywhere on Earth to get to the state of art in anything computers can be made to do, by reading what is fully available and by experimenting with it, and sharing the consequences freely. True computer science. Experimentation, hypothesis formation, more experimentation, more knowledge for the human race.

Which brings us back to this question of anonymity, or rather, personal autonomy. One of the really problematic elements in teaching young people, at least the young people I teach, about privacy, is that we use the word privacy to mean several quite distinct things. Privacy means secrecy, sometimes. That is to say, the content of a message is obscured to all but its maker and intended recipient. Privacy means anonymity; sometimes that means messages are not obscured, but the points generating and receiving those messages are obscured. And there is a third aspect of privacy, which in my classroom, I call autonomy. It is the opportunity to live a life in which the decisions that you make are unaffected by others' access to secret or anonymous communication.

There is a reason that cities have always been engines of economic growth. It isn't because bankers live there. Bankers live there

because cities are engines of economic growth. The reason cities have been engines of economic growth since Sumer is that young people move to them to make new ways of being. Taking advantage of the fact that the city is where you escape the surveillance of the village, and the social control of the farm.

The network, as it stands now, is an extraordinary platform for enhanced social control. Very rapidly, and with no apparent remorse, the two largest governments on earth, that of the United States of America and the People's Republic of China have adopted essentially identical points of view. A robust social graph connecting government to everybody and the exhaustive data mining of society is both governments' fundamental policy with respect to their different forms of what they both refer to, or think of, as stability maintenance. It is true of course that they have different theories of how to maintain stability for whom and why, but the technology of stability maintenance is becoming essentially identical.

We who understand what is happening need to be very vocal about that. But it isn't just our civil liberties that are at stake.... We need to make clear that the other part of what that costs us is the very vitality and vibrancy of invention culture and discourse.... And that freedom to tinker, to invent, to be different, to be non-conformist—for which people have always moved to the cities that gave them anonymity, and a chance to experiment with who they are, and why they can do.

This more than anything else, is what sustains social vitality and economic growth in the 21st century. Of course we need anonymity for other reasons. Of course we are pursuing something that might be appropriately described as protection for the integrity of the human soul....

We need Free Software, we need Free Hardware we can hack on, we need Free Spectrum we can use to communicate with one another, without let or hindrance. We need to be able to educate and provide access to educational material to everyone on Earth without regard to the ability to

pay. We need to provide a pathway to an independent economic and intellectual life, for every young person. The technology we need, we have.

I have spent some time…trying to make use of cheap, power-efficient compact server computers, the size of AC chargers for mobile phones, which with the right software we can use to populate the Net with robots that respect privacy, instead of the robots that disrespect privacy, which we now carry in almost every pocket.

We need to retrofit the first law of robotics into this society within the next few minutes or we're cooked. We can do that. That's civil innovation. We can help to continue the long lifetime of general-purpose computers everybody can hack on—by using them, by needing them, by spreading them around. We can use our own force as consumers and technologists to deprecate closed networks and locked-down objects.

Then came Edward Snowden. And then, a few months later, in November and December of last year, Eben gave four lectures at Columbia Law School titled "Snowden and the Future" (**http://snowdenandthefuture.info**). In them, he revisited some of what he said in the speech above, and laid out specific assignments I now pass along to *Linux Journal* readers (**http://snowdenandthefuture.info/PartIV.html**):

We need to decentralize the data, you understand. If we keep it all in one great big pile, if there's one guy who keeps all the e-mail and another guy who does all the social sharing about getting laid, then there isn't really any way to be any safer than the weakest link in the fence around that pile.

But if every single person is keeping her and his own, then the weak links on the outside of that fence get the attacker exactly one person's stuff. Which, in a world governed by the rule of law, might be exactly optimal: one person is the person you *can* spy on because you've got probable cause.

E-mail scales beautifully without anybody at the center keeping all of it. We need to make a mail server for people that costs five

bucks and sits on the kitchen counter where the telephone answering machine used to be, and that's the end of it. If it breaks, you throw it away.

Decentralized social sharing is harder, but not so hard that we can't do it. Three years ago I called for it. Wonderful work has been done that didn't produce stuff everybody is using, but it's still there: it can't go away, it's free software, it will achieve its full meaning yet.

For the technologically gifted and engaged around the world this is the big moment, because if we do our work correctly, freedom will survive and our grandkids will say: so what did you do back then? I made SSL better.

So let's do that. And much more.■

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!