LINUX JOURNAL

Since 1994: The Original Magazine of the Linux Community

DECEMBER 2015 | ISSUE 260 | www.linuxjournal.com

HOW-TO: INTERROGATE YOUR LINUX SYSTEM'S HARDWARE

INTERVIEW WITH

LinuxQuestions.org

Founder JEREMY GARCIA

AN INDEPTH LOOK AT

SECURE CODING PRACTICES

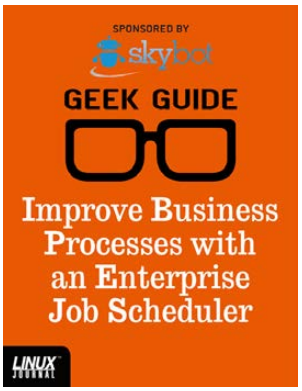
+

Can We Save Wireless from Regulators?

Build a Better BirdCam

Add Two-Factor Authentication to SSH

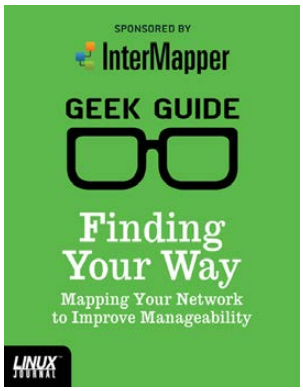
WATCH: ISSUE OVERVIEW



Improve Business Processes with an Enterprise Job Scheduler

Author:
Mike Diehl

Sponsor:
Skybot



Finding Your Way: Mapping Your Network to Improve Manageability

Author:
Bill Childers

Sponsor:
InterMapper

DIY Commerce Site

Author:
Reuven M. Lerner

Sponsor: GeoTrust

Combating Infrastructure Sprawl

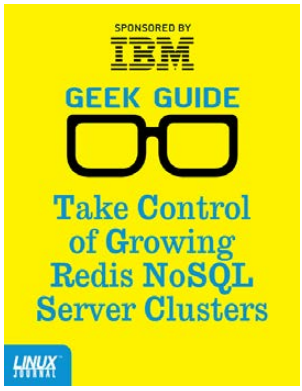
Author:
Bill Childers

Sponsor:
Puppet Labs

Get in the Fast Lane with NVMe

Author:
Mike Diehl

Sponsor:
Silicon Mechanics & Intel



Take Control of Growing Redis NoSQL Server Clusters

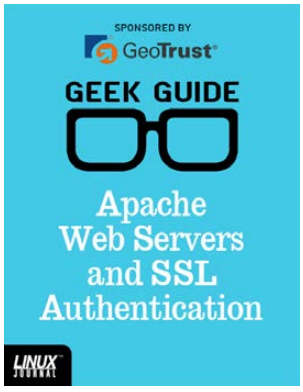
Author:
Reuven M. Lerner

Sponsor: IBM

Linux in the Time of Malware

Author:
Federico Kereki

Sponsor:
Bit9 + Carbon Black



Apache Web Servers and SSL Encryption

Author:
Reuven M. Lerner

Sponsor: GeoTrust

CONTENTS

DECEMBER 2015
ISSUE 260

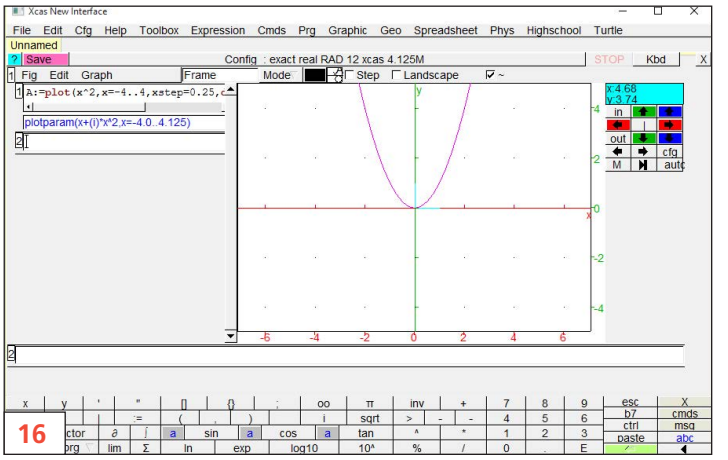
FEATURES

COLUMNS

IN EVERY ISSUE

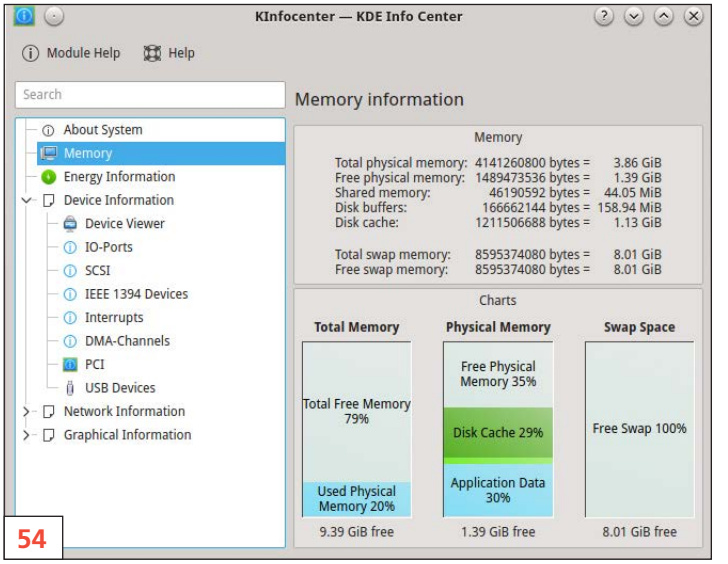
ON THE COVER

- **Interview with LinuxQuestions.org Founder Jeremy Garcia, p. 78**
- **How-To: Interrogate Your Linux System's Hardware, p. 54**
- **An Indepth Look at Secure Coding Practices, p. 42**
- **Add Two-Factor Authentication to SSH, p. 30**
- **Can We Save Wireless from Regulators?, p. 90**
- **Build a Better BirdCam, p. 34**



16


34



54

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

LINUX JOURNAL

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

Contributing Editors

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

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

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

SHAWN POWERS

You've Got Questions? We've Got Jeremy Garcia!

Anyone who's active in the Linux community knows that while we love open source and we swear by the kernel, the real power of Linux is the people making up the community. Whether it's folks using Linux in a server room, people contributing code or documentation to a project in their spare time, or even geeks putting Linux stickers on their laptops, Linux is about people. This month, Brian Conner has a great interview with Jeremy Garcia, the founder of LinuxQuestions.org. If there's a better example of a healthy and interactive Linux community, you'll be hard pressed to find it. If you want to know the history of LinuxQuestions, find out more about the man behind it, or even what the future holds, you should check out the interview. Jeremy is as cool as you'd expect him to be!

We also have our regular gang of columnists, starting with Dave Taylor, who teaches us how to do his taxes. More specifically, he shows us how to analyze CSV files from the command line. Proprietary file formats are frustrating to work with, but thanks to the simplicity and standard-ness of CSV, Dave proves it can be an awesome format for folks who like their calculating to be done in a script.

Kyle Rankin delves back into the world of green text on a black background (I'm assuming there, but I have no doubt I'm correct) when he demonstrates how to set up two-factor

VIDEO:
Shawn Powers runs through the latest issue.

authentication for SSH connections. In the past, he's described how to set up SSH keys with passphrases to increase security, but it's also possible to create true two-factor auth using Google Authentication. If you want to make your server more secure than passwords alone can manage, you won't want to miss his column.

I actually head back to my backyard this month and discuss some of the upgrades and changes to BirdCam. You might remember my articles outlining how I created a pseudo-streaming Webcam experience pointing at the bird feeders outside my office window. Since we recently moved, I took the opportunity to make some changes, and they were interesting enough that I thought I might share with the class. Whether you have a BirdCam, BabyCam or just use the code to improve your own weekend project, this column should teach some new tricks.

We also introduce a new columnist this month, and she's already a perfect fit in our *Linux Journal* family. You may remember her Guest EOF column from a year or so ago called "Girls and Software". In this issue, Susan Sons walks through the process to make sure you're using only trusted code when you install packages and dependencies. Distributions generally use cryptographically signed packages for their standard programs, but developers and/or package maintainers need to be diligent in order to avoid compromising security when offering custom applications. If that sounds confusing, be sure to read the first installment of Susan's, Under the Sink column. Welcome to the family, Susan!

Federico Kereki is back this month as well, this time showing how to pry as much information out of your Linux system as possible. When it comes to hardware, Linux supports just about everything under the sun. Thanks to a handful of tools, you can learn about the specific hardware on your system and use that information to troubleshoot those rare occasions when things don't work as planned.

This is a fun issue of *Linux Journal*, with a big focus on who we are as a community. We include all the bits and pieces you expect from an issue of *Linux Journal*, and if you're interested in being an active part of the Linux community, you couldn't pick a better issue to read. We hope you enjoy the December 2015 issue of *Linux Journal* as much as we've enjoyed putting it together!■

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

diff -u
WHAT'S NEW IN KERNEL DEVELOPMENT

Linux capabilities are one of the more fluid and less defined regions of kernel development. **Linus Torvalds** typically has no trouble violating **POSIX** standards if he sees a better way of doing something. In the case of filesystem capabilities, however, there's no standard to violate. The best we've got is a POSIX draft document that was discarded before becoming official. So really, anyone with a good idea can come along and make big changes in that part of the kernel.

Filesystem capabilities refer to a finer-grained set of permissions than the traditional choice between running something as a regular user or running as root.

Recently, **Eric W. Biederman** and **Andy Lutomirski** found themselves tackling filesystem capabilities from opposite directions. Eric wanted to allow a process that's been granted one set of capabilities to invoke system calls using an even more constrained set of capabilities. Presumably, the goal would be to increase security by preventing system calls from being abused for nefarious purposes. And, Andy wanted to allow one process to allow a completely separate process to perform system calls on its behalf. This might allow the formation of system call services to centralize all system call usage and make it easier to secure those uses.

The discussion went round and round. Eric's idea, as he later clarified, was actually a bit broader than it appeared at first glance—he wanted to convert the Linux implementation of POSIX capabilities into "Real Capabilities". The term Real Capabilities refers to a computer science concept that pre-dates POSIX capabilities. It refers to the idea of giving a process some sort of token that allows it to perform a specified action on a specified object.

Ultimately, nothing about capabilities, or any new patches in that area, can have real clarity until they go into the kernel. Before then, there's always the possibility that they'll violate something important or aim in the wrong direction. But, it's

cool to watch Eric and Andy, and lots of other folks, trying to figure it out.

One recurring problem with Linux is the need for **backward compatibility**. Actually, this affects virtually the whole Open Source world, but Linus Torvalds takes a particular strict stance on the issue with regard to the Linux kernel. If there's a compiled piece of user software out in the wild that relies on a kernel feature, even a dumb kernel feature, then future kernels have got to support that feature, so that the piece of user software will continue to run after a kernel upgrade.

It makes sense. But as Andy Lutomirski recently said, the result was a batch of features that existed only to support old programs. And by carrying these features perpetually into the future, he said, newer software ran the risk of accidentally using those features or even becoming reliant on them.

He proposed allowing new software to turn off those compatibility features explicitly, but that turned out to be more complex than he'd originally thought. Specifically, one of his cornerstone ideas—granting the ability of newer software to turn off the **vsyscall** page—was not easy to arrange. Andy's initial idea was to have the compiler identify at compile time software that used newer versions of **libc**, and then have that piece of software elect to disable **vsyscall** at runtime. But, he didn't see a good way to accomplish that, and **Brian Gerst** pointed out that vsyscall was globally shared and couldn't be shut off for individual processes.

This actually turned out not to be 100% true. Although Andy agreed that vsyscall was shared globally, the mechanisms to execute it were all emulated in the kernel, and those could be disabled on a per-process basis.

Rich Felker proposed another workaround for vsyscall's global availability. He said the kernel could simply unmap all means of executing vsyscall. Any older software that tried to access it would generate a page fault, which the kernel could then catch and emulate vsyscall just for that program.

But, Andy didn't go for that idea. He said that modern instrumentation tools might want to read the targets of calls, and a page fault would prevent that. Any vsyscall solution, Andy said, had to maintain compatibility for those tools.

On the other hand, as Rich said, those modern tools might never be used on ancient binaries in practice. And even if they were, it might be possible to code up specific kernel

workarounds for each use case in a less invasive way than trying to come up with a complete solution for vsyscall.

It's a robust debate, complicated by the fact that it's difficult to know for sure if anyone is actually running old binaries that depend on this or that kernel compatibility feature. But, Andy made it clear that cleaning out compatibility features was not really his primary goal, so much as it was to eliminate potential security holes. Apparently, **Google's Project Zero** had identified more exploits: http://googleprojectzero.blogspot.com/2015/08/three-bypasses-and-fix-for-one-of.html.

The Linux **framebuffer**, once a bastion of innovation, is now on the chopping block, in favor of the **Direct Rendering Manager** (DRM) subsystem. The **fbdev** maintainer, **Tomi Valkeinen**, has asked everyone to stop submitting new fbdev drivers and to aim their efforts at DRM instead.

It was not as uncontroversial as you might think. It turned out, as folks like **Thomas Petazzoni** said, that it still was easier to write very simple drivers for fbdev, than it was for DRM. Just in terms of lines of code, **Geert Uytterhoeven** noticed that the simplest fbdev drivers were just a few hundred lines of code, while the simplest DRM drivers might require a couple thousand.

No one argued that this would be a permanent problem. If anything, the discussion highlighted the need to write some supporting libraries for DRM and help speed up the ultimate elimination of fbdev.—**ZACK BROWN**

They Said It

The secret of being boring is to say everything.

—Voltaire

It's not enough that we do our best; sometimes we have to do what's required.

—Winston Churchill

The best way to escape from a problem is to solve it.

—Alan Saporta

To avoid criticism do nothing, say nothing, be nothing.

—Elbert Hubbard

The vitality of thought is in adventure. Ideas won't keep. Something must be done about them.

—Alfred North Whitehead

Node.js Version Chaos Management

I'm just starting out in the world of development, and many of the projects I'm interested in exploring are written in Node.js. If you're an old hand at such things, you already know that which version of Node you use on a particular application is vitally important. (This is actually one of the reasons Docker is so amazingly amazing when it comes to deploying Node apps, but I digress.)

For folks like me, the version issue can be confusing and frustrating. Thankfully, I ran across a simple tool with a simple name: n. Once you have Node.js installed on your system, using n, it's possible to download and make active a very specific version of the program, so your specific application works properly. In fact, when I was installing the NOMP stratum server for Bitcoin mining recently, I had to use n to try more than a dozen versions before I found the one that worked as expected.

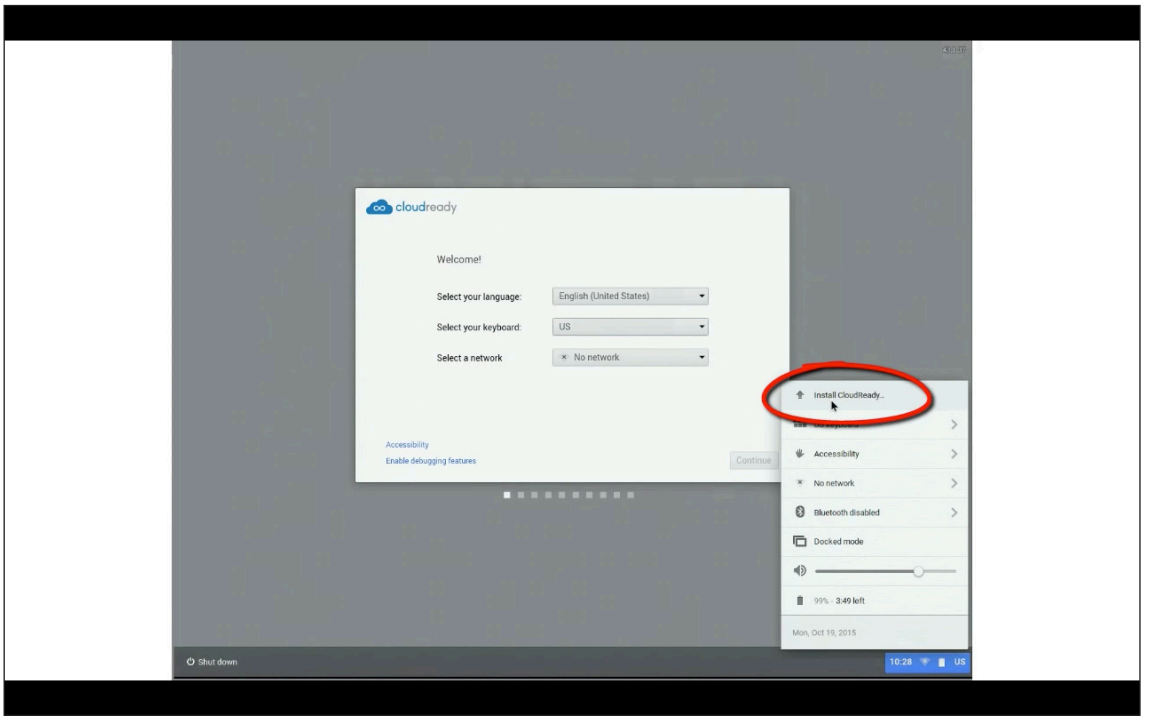
Node.js is a powerful, incredible language that is used by many smart developers. Those of us who are just getting started, however, are easily intimidated by version needs. If that describes you, or if you understand the nuances but just want a quick and easy way to manage it, check out n today. There are instructions on the Github page: https://github.com/tj/n.—**SHAWN POWERS**

Android Candy: How Clever We Once Were

I freely admit I learned about this app from my wife. In fact, I saw a few nostalgic posts on her Facebook timeline and investigated where they came from. It turns out she had installed an app called Timehop. I normally wouldn't write about something that at first glance seems like an ego-stroking shot of nostalgia, but I had so much fun looking at the posts it dug up, I couldn't help myself.

Timehop is a free app in the Google Play Store. Once you install it, you're guided through connecting to all the major social-media networks (Twitter, Facebook, Instagram and a few others I didn't recognize). Then, using some sort of popularity algorithm (or magic, I'm not sure which), it finds funny and/or memorable posts and photos from the past and shows them to you. You are able to share those old posts from the app and allow others to experience your little hit of nostalgia as well.

I still feel a little silly sharing this app with the *Linux Journal* community. But really, it's so much fun, if you're having a bad day, give it a try and see if a three-years-ago-you can cheer you up. I know two-year-ago Shawn made me smile!—**SHAWN POWERS**



Chromebookify Your Laptop Now!

A few years ago there was a project designed to boot generic laptops so they functioned as Chromebooks. It was a cool project, but unfortunately, the compatibility wasn't great, and it wasn't reliable to use on a daily basis. Although Chromebooks are old news these days, it still would be quite useful to transform aging laptop computers into Chromebooks. Because they have such low system requirements, older laptops running the ChromeOS can become quite useful again.

Thanks to the folks at Neverwhere, you can get the CloudReady installer that installs Chromium onto a wide variety of laptops from various manufacturers. (A long list of tested models is available at http://go.neverware.com/certifiedmodels.) I actually have a Dell D420 that is getting very long in the tooth as a Linux machine, but as a Chromebook, it's still quite effective. If you have an aging laptop, give CloudReady a try. It's free, and you even can boot off USB to check it out before installing.

CloudReady from Neverwhere: http://www.neverware.com/free.

—SHAWN POWERS

Symbolic Algebra Everywhere

Previously in this space, I have covered software packages like Maxima that can be used to solve symbolic mathematics problems. Several packages are available that can do those types of calculations. In this article, I discuss Xcas/Giac. Xcas is the GUI interface to the system. Giac is the command-line program that provides access to the core engine. Xcas has the functionality to handle symbolic algebra, two-dimensional and three-dimensional graphing, spreadsheets and statistics. It even has its own programming language that you can use to add extra functionality of your own. Although you can use the default interface that comes with Xcas, you also can link the CAS (Computer Algebra System) engine as a shared library to your own C++ code.

Packages are available for many different Linux distributions, but they usually aren't available via the default package management systems. For example, in Ubuntu, you need to add an APT source that points to the home page for Xcas. Then you can use the following to install it on your system:

```
sudo apt-get install giac python-giacpy
```

Once it is installed and you start it up, Xcas asks what mode you want to work in. You can select from spreadsheet, CAS, programming or geometry. Whenever you start a new session within Xcas, you get this same initial interface. If you want to change it later, select the Cfg→General Configuration menu option. This pops up a new window where you can select the Level option. If you choose the CAS option, you get the starting window shown in Figure 1.

To open a new tab with the same level, click the File→New Session menu item. You also can open a new tab using any of the available levels, or modes, using menu commands. They are a bit hard to find though. For example, you can get a new spreadsheet with the Spreadsheet→ New Spreadsheet menu item.

There is far too much functionality available within Xcas to explain how everything works in such a short article, but I'll try to cover some of the most interesting parts.

Let's start by looking at the command level. This operates in a form similar to the worksheet in Maple or

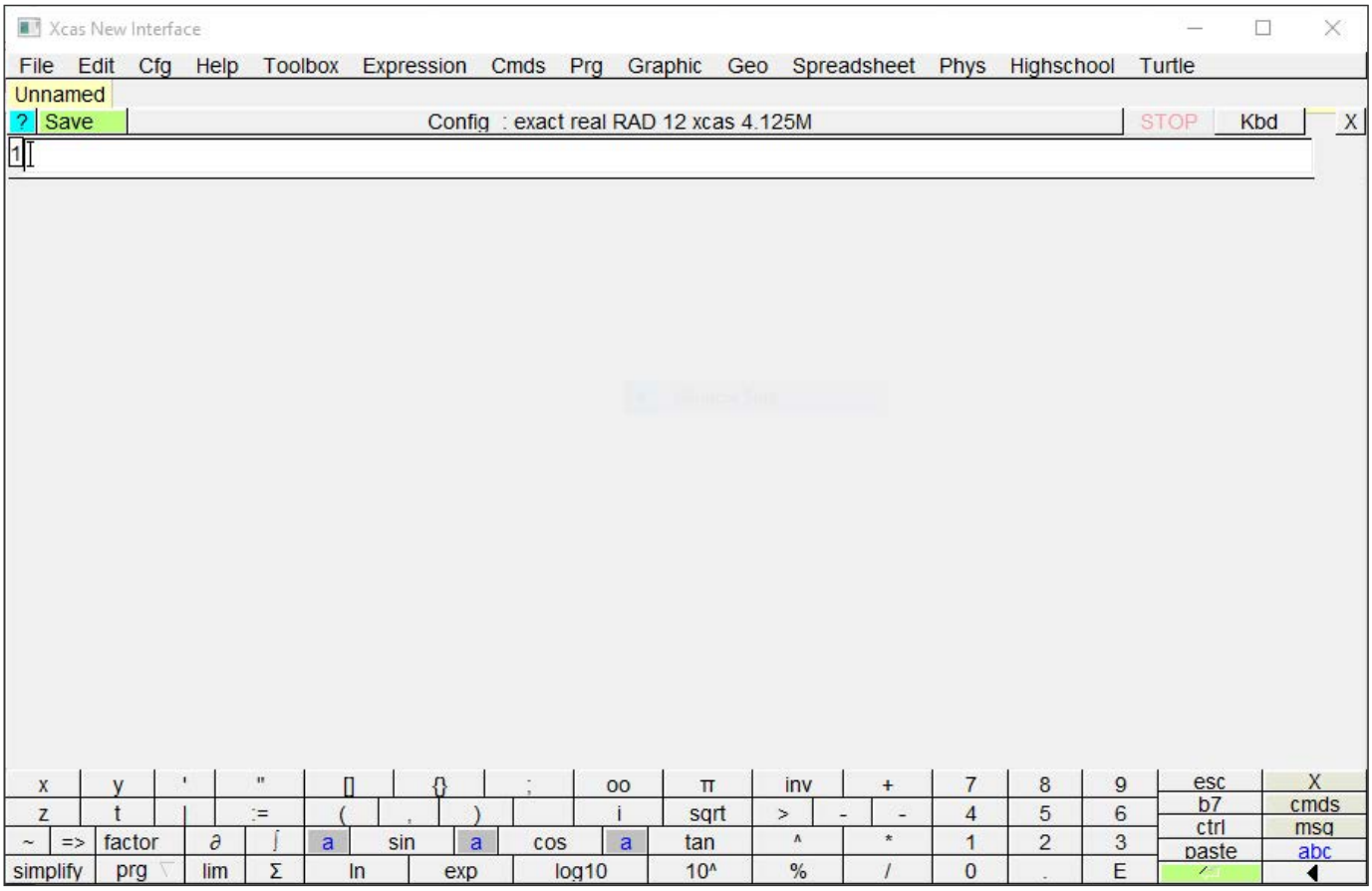


Figure 1. This the opening window in CAS mode.

Mathematica. You start with the first empty command line and enter the mathematical expression you want to evaluate. Pressing Enter runs the command, displays the output in a new pane, and creates a new command line and drops the cursor there, ready for your next command. This style should be comfortable to anyone with even a little bit of experience.

The keyboard panel at the bottom of the window gives you a selection of common elements that you will likely use within your commands. If you don't need to use it, you can remove that pane by clicking the Kbd button at the top of your session window.

The library of available commands is very large. Luckily, you can find the majority of them by clicking on the Cmds menu item. Here, you can find sections for several different areas, such as complex numbers, group theory, calculus or probability.

No system has everything that you may possibly need when you start doing any kind of scientific computing. This means that you need to be able to add

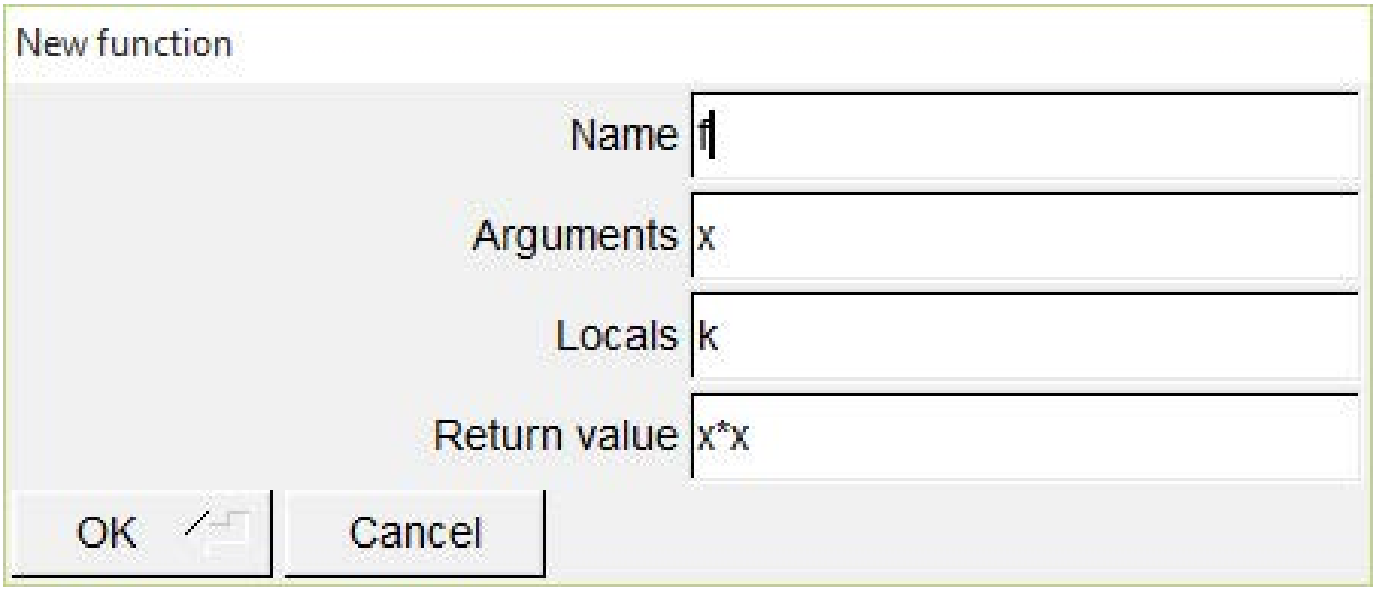


Figure 2. You can create your own functions in Xcas.

new functionality of your own devising. With Xcas, you can create a new function by clicking on the Prg→New program menu item. This pops up a new window where you can define the name, arguments, locals and a return value. Once you are happy with these settings and click the OK button, you will get a new program pane with a template ready for you. You then can add in any other code that is required by your new functionality.

There are menu options within the programming pane to help you with the syntax of programming structures, such as loops, conditionals and IO. In Xcas, functions need to be compiled before they can be used. This compiling step happens when you click the OK button in the programming pane. If there are any errors, you will get a message in the output pane. If there are no errors, you will get a "Success compiling" message.

You can include graphics inline within a session. If you want a general graphics pane, click the Geo→New figure 2d or Geo→New figure 3d menu item. This gives you a graphics pane along with an associated command pane where you can enter the plotting commands you want drawn. If you have a specific item drawn, you can select one of the other items in the Geo menu section. For example, if you want to graph a function, you can go to Geo→Graph→Function. This

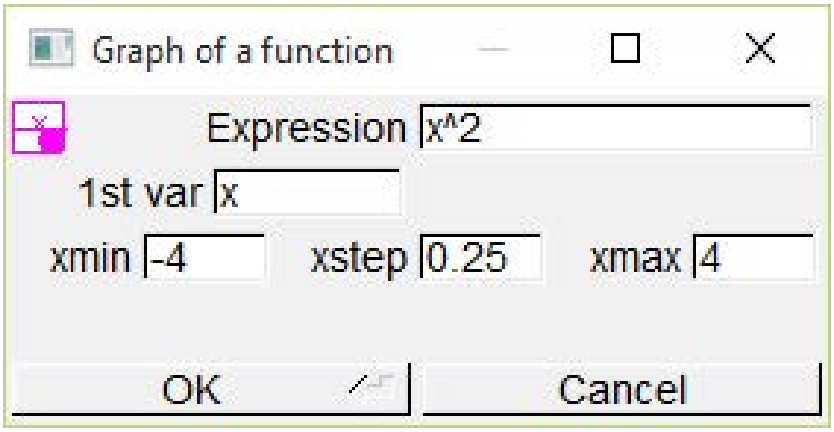


Figure 3. Graphing functions is pretty easy.

pops up a new window where you can enter the function you want to graph, along with the limits of the independent variable. When you click OK, you get the graph drawn inline within your current session.

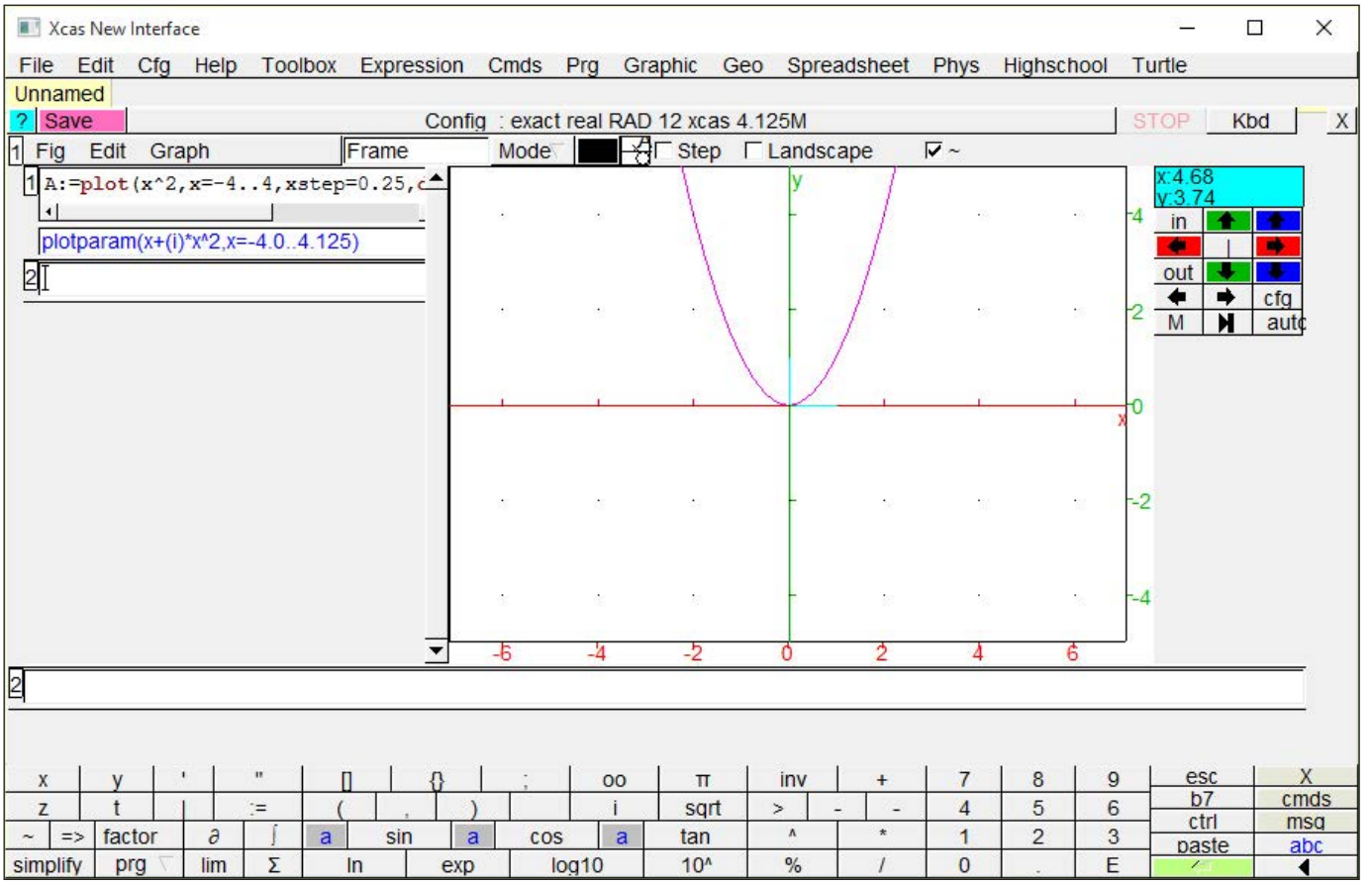


Figure 4. Graphs show up inline within your session.

Xcas is designed to be reasonably good at interacting with other CAS software. With this idea in mind, it is no surprise that you can import and export worksheets using several different formats. Xcas will handle Maple and Mu PAD file formats fairly well. It also can handle the file format used by TI calculators (like the TI-89 or the Voyage 200). With this type of support, you should be able to share your work with many other people.

With Xcas, you can work on almost any system that you have access to. You can use your Linux system to do major amounts of work, and then you can continue that work on your Android or Apple device, or even use your Texas Instruments calculator. Although the interface is a bit confusing, and the learning curve is rather steep, there is no denying just how powerful Xcas is.—**JOEY BERNARD**

A BZ Media Event

WANTED

DEVELOPERS, IT PROS, INFORMATION WORKERS & PROJECT MANAGERS SEEKING THE BEST SHAREPOINT TRAINING!

Learn what's new in SharePoint and Office 365!
Look into the future with SharePoint 2016!

Whether you want to learn about what's coming in SharePoint 2016, are still making the most out of SharePoint 2013 or even 2010, or getting started with Office 365, you will find the SharePoint and Office 365 training you need at SPTechCon.

SharePoint 2016 training!

- SharePoint 2016 Design and Implementation
- SharePoint Farm Architecture and Performance: Testing on 2013 and Planning for 2016
- What's New for Power Users in SharePoint 2016?
- What's New for the BI workloads in SharePoint 2016?

…and much, much more!

Office 365 training!

- Demystifying Office 365 Administration
- The 10 Best Office 365 Features You've Never Used (But Should)
- Making Office 365 Work for the Business: Building Powerful No-Code Solutions
- How Office 365 is Changing the Face of Collaboration and Communication

…and much, much more!

SharePoint 2013 training!

- Going Mobile with SharePoint 2013
- SharePoint Designer 2013 Workflows – An Introduction
- Upgrading to SharePoint 2013
- SharePoint 2013: Online vs. On-Premises

…and much, much more!

Still on SharePoint 2010?

There's plenty of excellent information for you as well.

SPTechCon
The SharePoint Technology Conference

FEBRUARY 21-24, 2016

AUSTIN, TEXAS

www.sptechcon.com

Non-Linux FOSS: Airsonos

I love Sonos. There probably are some audiophiles reading this who rolled their eyes at my lack of auditory prowess, but honestly, the speakers sound wonderful to my 1980s-damaged eardrums. Granted, the Wi-Fi-enabled speakers are very expensive, thus limiting my supply. I'm amazed at the ability for the speakers to sync a single audio source throughout my house perfectly without the need for wires. At all.

The problem (apart from the price tag) is the limited options for music sources. You can stream radio stations, Pandora radio and even MP3 music files from a central network-accessible server. For my family of teenage girls, however, the inability to stream via Airplay (yes, my family has many Apple products) is a showstopper. So in their upstairs bathroom there's a $300 speaker on the shelf, and they just listen to their phone speakers while in the shower. It breaks my heart. Sonos offers line-in options for its larger speakers, but it's really a kludge and doesn't work well.

Enter Airsonos. An open-source project, Airsonos is a Node.js-based application that runs on a server and probes the network for on-line Sonos speakers. It then creates Airplay devices for each speaker, and an iPhone or iPad easily can stream to a Sonos speaker. I personally run Airsonos

as a Docker app, and it's a "set it and forget it" sort of application. In fact, Airsonos has all the makings of an Editors' Choice award-winning project:

- It's open source.
- It runs on a Linux system.
- It's easily Dockerized.
- And, it solves a real problem in an awesome way!

So, this month's Editors' Choice award goes to Airsonos, with shared award status to the Dockerized app version maintained by "justintime"—thank you for making my nerdy world a better place!

- Airsonos: https://github.com/stephen/airsonos.
- Dockerhub: https://github.com/justintime/docker-airsonos.

—SHAWN POWERS

DAVE TAYLOR

Analyzing Comma-Separated Values (CSV) Files

Introducing FIX-CSV, a script to analyze and fix errors in comma-separated values (CSV) files, so Dave finally can do his taxes. No, really. Read on!

Ugh. I've been working on my taxes. I know, it's a bit weird to be doing my taxes in the autumn, but if you defer and file an extension with the IRS every year, well, then you're used to tax time being September/October, not April 15th.

I have a very old-school, geeky way of preparing for my own taxes, and it involves using an Excel spreadsheet to enter all my line item expenses then normalizing and cleaning up the data. When that's all done, which typically involves a lot of sorting and re-sorting of the data, I then export it all as a comma-separated values data file and pull out a Linux shell script to analyze and summarize expenses by category.

I suppose I could do that in the spreadsheet program itself, but it either would involve me having to learn the spreadsheet's programming language (for example, Visual Basic in Microsoft Excel 2016) or manually click-dragging series of cells to summarize their values. Both are tedious, and however peculiar my solution, the idea of actually learning Visual Basic just boggles my mind, so that's just not an option.

But, there's a lurking problem in the CSV format I use, and to understand it, I need to dig in to exactly what's being formatted.

A typical expense entry has four fields: date, category, amount and any detailed notes or comments. For example:

```
4/10/15    subscriptions    19.99    Linux Journal
```

All of it's neatly organized in columns and data cells, as befits a spreadsheet, of course.

Random aside: did you know that it was a spreadsheet that proved the viability of the personal computer back in the day? VisiCalc was the groundbreaking app with its sophisticated interface (for the day, at least) and ability for accountants and business folk to create sophisticated mathematical tables and regular people to...balance their checkbooks. Yes, one of the killer apps for the very first generation of PC was checkbook balancing. We've come a long way!

With a spreadsheet populated with these four fields, the easiest way to create a dataset for further work is to export it as comma-separated values, the "CSV" format. Here's how that particular line will be exported:

```
4/10/14,subscriptions,19.99,Linux Journal
```

Not too bad, and it's easily managed by changing the field separator to a comma. For example, to extract just the numeric value: `cut -d, -f3`.

In fact, once the output is sorted by category, it's a simple awk script to read the CSV file line by line, testing each category against the previous and summing up values as it goes:

```
BEGIN { sum=0; category=""; FS="," }
{ if ( $2 != category ) {
    if ( sum > 0 ) { print category," == ",sum; }
    sum=$3
  }
  else {
    sum+=$3
  }
  category=$2
}
END { print category, " == ", sum }
```

Awk scripts are blocks of code that match specified regular expressions, although all three of the above blocks are somewhat special cases. The first, `BEGIN` is executed before the first line of input is read in, so it just initializes variables. The last, `END`, is run after the last line is processed.

And the middle section? It's a regular expression that matches every line (by being omitted entirely). Since the field separated is set to a comma, it means that within the main awk block, $1 is the date, $2 is the category, $3 is the amount and $4 is the comment.

For the sample input line, it'd be:

```
$1 = 4/10/14
$2 = subscriptions
$3 = 19.99
$4 = Linux Journal
```

If you're thinking about the field separator, it's immediately obvious what's going to cause trouble. Instead of actually escaping the comma, Excel has just quoted the field that has the comma in the output.

That's easy enough, and easy to understand, I expect. The code's also quite readable, so you can see what's going on.

The problem? The problem arose when I encountered lines where one of the fields had a comma. For example, if I had the comment field on this line be "Linux Journal, annually", the CSV output would be:

```
4/10/14,subscriptions,19.99,"Linux Journal, annual"
```

If you're thinking about the field separator, it's immediately obvious what's going to cause trouble. Instead of actually escaping the comma, Excel has just quoted the field that has the comma in the output.

In this particular instance, it's not that big of a problem. All that happens is that instead of having "Linux Journal, annual" as field 4, you'd end up with "Linux Journal" in field 4 and "annual" in field 5.

Where this does turn out to be a problem is with the expense itself. In particular, Excel displays four-digit values with a comma if they're a currency: 1,300.00

With. A. Comma.

And, that comma survives the export to CSV format, which is a bit mind-boggling. Suffice it to say, it turns out to be tricky, as you can see here:

```
4/10/14,subscriptions,"1,300.99",Linux Journal
```

The easy way to solve the problem is to choose a different cell format style that excludes the predilection of the spreadsheet to export with commas. But hey, you read my column so you're probably used to taking the long, circuitous route. So, let's do it again!

A bit of analysis reveals that if you simply split out the lines that contain quotes from those that don't, you quickly can identify those that need fixing or tweaking. Let's start with

the raw file that contains two lines: one with the embedded comma problem, one without:

```
4/7/14,subscriptions,199.99,Ask Dave Taylor Monthly
4/10/14,subscriptions,"1,300.99",Linux Journal
```

There are lots of ways to identify the line with the problem, including picking lines with more than the expected four fields, but let's do something easier:

```
$ grep \" expenses.csv
4/10/14,subscriptions,"1,300.99",Linux Journal
```

The `cut` command now can be used to extract just the quoted field—`cut -d\" -f2`—and then any comma removed with `sed`.

In other words, use a script block like this, if the line in question is stored in the variable `inline`:

```
f1=$(echo $inline | cut -d\" -f1)
f2=$(echo $inline | cut -d\" -f2)
f3=$(echo $inline | cut -d\" -f3)
```

Let's examine what these three `cut` statements do: `f1` is everything prior to the first quote mark; `f2` is

everything that's been quoted, and `f3` is everything after the quoted passage. In the case of the Linux Journal subscription, it'd look like this:

```
f1=4/10/14,subscriptions,
f2=1,300.99
f3=,Linux Journal
```

That's just about all of the hard work done because now you safely can strip the commas from `f2` without affecting the rest of the line, safely stored in `f1` and `f3`.

Then it all can be reassembled in a single line:

```
echo $f1`echo $f2|sed 's/,//g'`$f3
```

Remember here that the backticks denote a sequence that's going to be passed to a subshell and its output substituted. With the Linux Journal line, the output is exactly as desired:

```
4/10/14,subscriptions,1300.99,Linux Journal
```

It turns out that's the solution, and you now have all the basic pieces of the script itself. Actually, there's no need to separate out files with quoted lines versus those that don't have quotes because that can be done within the script itself.

And so, here's the succinct script that can fix the CSV file quickly and easily:

```
#!/bin/sh
# fix CSV files with embedded commas
while read inline
do
  if [ ! -z "$(echo $inline | grep \")" ]
  then
    f1=$(echo $inline | cut -d\" -f1)
    f2=$(echo $inline | cut -d\" -f2)
    f3=$(echo $inline | cut -d\" -f3)
    echo $f1`echo $f2|sed 's/,//g'`$f3
  else
    echo $inline
  fi
done
exit 0
```

Does it work? Let's give it a whirl:

```
$ sh fix-csv-commas.sh < expenses.csv
4/7/14,subscriptions,199.99,Ask Dave Taylor Monthly
4/10/14,subscriptions,1300.99,Linux Journal
```

And there you go. As for me, well, it's back to finishing up my taxes now that I've managed to burn a few hours creating this useful "CSV-Fixer" script.■

Dave Taylor has been hacking shell scripts since the dawn of the computer era. Well, not really, but still, 30 years is a long time! He's the author of the popular *Wicked Cool Shell Scripts* (10th anniversary update coming very soon from O'Reilly and NoStarch Press) and can be found on Twitter as @DaveTaylor and more generally at his tech site http://www.AskDaveTaylor.com.

KYLE RANKIN

Two Factors Are Better Than One

Make it even harder for an attacker to compromise your SSH accounts.

Although I've always been interested in security, there are just some security measures I've never liked. SSH brute-force attacks end up being a major way that attackers compromise Linux systems, but when it comes to securing SSH, I've never been a fan of changing your SSH port to something obscure, nor have I liked scripts like fail2ban that attempt to detect brute-force attacks and block attackers with firewall rules. To me, those measures sidestep the real issue: brute-force attacks require password authentication. If you disable password authentication (set `PasswordAuthentication` to no in your sshd_config) and use only SSH keys, you can relax about all those brute-force attacks knocking on your door.

In a past article ("Secret Agent Man", December 2013), I wrote about why you should set a passphrase on your SSH keys and how to use SSH Agent to make password-protected keys a bit less annoying. In one respect, you can think of password-protected SSH keys as a form of two-factor authentication. The key is something you have, and the password is something you know. The problem, however, is that if you host a system with multiple users, you can't enforce password-protected SSH keys from the server side. So in this article, I discuss how to add two-factor authentication to an SSH server that accepts only keys.

These days, more services on-line offer two-factor authentication (2FA) as an extra layer of security on top of a user name and password. After you perform your normal authentication, you provide your 2FA token (usually a

string of digits) that authenticates you. Although in the past, 2FA required you to carry around a special hardware dongle, these days, a number of software approaches can use your cell phone instead. Some approaches use TOTP (Time-based One-Time Password), so your phone just needs accurate time but no network to function. Other approaches use push notifications, SMS or even a phone call to share the 2FA token, and some implementations can use all of the above.

Some 2FA SSH implementations work via the `ForceCommand` directive placed in the SSH configuration for a particular user and let you enable 2FA on a per-user basis. Others offer a PAM module you can add system-wide (and use for sudo authentication as well as SSH). Although a number of excellent 2FA SSH implementations exist for Linux, I've chosen Google Authenticator for a few reasons:

- It's free, and the source is available.
- It's been available and tested for a number of years.
- Packages are available for a number of distributions.
- Clients are available for a number of phone operating systems.
- It uses a custom PAM module, so it's easy to add 2FA system-wide.
- It provides a backup in the form of backup codes in case users lose or wipe their phones.

Install Google Authenticator

As I mentioned, Google Authenticator is packaged for a number of distributions, so, for instance, on Debian-based systems, you can install it with:

```
$ sudo apt-get install libpam-google-authenticator
```

If for some reason it isn't packaged for your distribution, you also can just go to https://github.com/google/google-authenticator/tree/master/libpam, download the software and make and install it according to the documentation there. You also will need to install the Google Authenticator app on your phone.

Configure User Accounts

I recommend setting up Google Authenticator for all of your user accounts (or at least all of the sysadmin accounts) before enforcing 2FA in SSH to make it easier to enroll all of the users and avoid the risk of locking people out. To configure Google Authenticator, each user needs

to log in and run google-authenticator. You will be presented with a series of questions where it's safe to answer "y"; however, I generally answer no to extending the time window to four minutes, and I also answer no to rate limiting, since as I disable password authentication, I'm less concerned with brute-force attacks. The output looks something like this:

```
$ google-authenticator

Do you want authentication tokens to be time-based (y/n) y
https://www.google.com/chart?chs=200x200&chld=M|0&cht=qr&chl
➥=otpauth://totp/username@debian%3Fsecret%3D4SK2LTLCTLCEV757
QR Code Removed
Your new secret key is: 4SK2LTLCTLCEV757
Your verification code is 221544
Your emergency scratch codes are:
  53267360
  44975412
  59302752
  36003899
  64736155

Do you want me to update your "/home/username/.google_authenticator"
 ➥file (y/n) y

Do you want to disallow multiple uses of the same authentication
token? This restricts you to one login about every 30s, but it
increases your chances to notice or even prevent man-in-the-middle
attacks (y/n) y

By default, tokens are good for 30 seconds and in order to
compensate for possible time-skew between the client and the
server, we allow an extra token before and after the current time.
If you experience problems with poor time synchronization, you can
increase the window from its default size of 1:30min to about 4min.
Do you want to do so (y/n) n

If the computer that you are logging into isn't hardened against
brute-force login attempts, you can enable rate-limiting for the
authentication module. By default, this limits attackers to no
more than 3 login attempts every 30s. Do you want to enable
rate-limiting (y/n) n
```

If you have libqrencode installed, the output also will contain a QR code in the console you can scan with the Google Authenticator app on your phone. Otherwise, you simply can enter the secret key into your Google Authenticator application on your phone. Also, be sure to write down those backup codes and store them in a safe place. These are one-time-use codes you can use to get back in to the system in case you ever lose or wipe your phone. Once you are logged back in, you can run google-authenticator again.

Configure PAM and SSH

Once your phone and user accounts are configured with Google Authenticator, you are ready to enforce 2FA in PAM and SSH. To do this, edit your /etc/pam.d/sshd file and add the following to the

top of the file:

```
auth required pam_google_authenticator.so
```

On my Debian system, I noticed that once I finished the configuration process, I would not only be prompted for my 2FA token, I'd also be prompted for my local system password. Because I wasn't interested in three-factor authentication (two-and-a-half factor authentication?), I noticed I needed to comment out the following further down in the file:

```
@include common-auth
```

Of course, if you aren't on a Debian-based system, this extra step may not be necessary.

The final step is to configure SSH. Hopefully you already have disabled password authentication for SSH in the past, and if not, I recommend you consider it. Most of the SSH 2FA guides out there (this one included) will tell you to enable `ChallengeResponseAuthentication` in your /etc/ssh/sshd_config:

```
ChallengeResponseAuthentication yes
```

I noticed, however, that when you are using key-based authentication instead of passwords, you need to add an additional setting to the config file:

```
AuthenticationMethods publickey,keyboard-interactive
```

Once these settings are in place, you can enable them by restarting your SSH service, which depending on your system may be one of the following:

```
$ sudo service ssh restart
$ sudo service sshd restart
```

After SSH has restarted, you should get an additional prompt the next time you SSH to the server:

```
$ ssh kyle@server1.example.com
Authenticated with partial success.
Verification code:
```

Type in the verification code that shows up in your Google Authenticator phone app, and you can log in. The nice thing about adding 2FA to SSH is that it provides an additional means of protection in case your computer is ever compromised or stolen. Attackers also would have to compromise or steal your phone before they could access your systems.■

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.

SHAWN POWERS

BirdCam, Round Three

BirdCam 3.0? Check out Shawn's latest improvements to BirdTopia.

I've been writing for *Linux Journal* since 2007 when I shared the process I used to make my own MAME arcade cabinet. Out of all the projects, how-tos, reviews and silliness I've written, nothing has gotten more feedback and discussion than my BirdCam. It's been more than a year since I last wrote about my setup, and since that time, I've moved to a new city and upgraded my cameras and software significantly. So to answer many of the questions I get about the current state of BirdCam, I figured I'd write about the technical details and hopefully inspire similar projects or upgrades for others. Plus, I get to talk about BirdCam, and that's just pure fun!

A Short Refresher

If you weren't with *LJ* two years ago when I first wrote about BirdTopia, I'll quickly catch you up. I have a bunch of bird feeders in my backyard, right outside my office window (Figure 1). I decided it would be awesome to put a public Webcam on them, but I didn't want to pay for LiveStream or anything like that. In fact, I wanted it to be strictly image-based instead of video. The initial article is at http://www.linuxjournal.com/content/its-bird-its-another-bird. Like most folks, I fiddle with things constantly, and BirdCam changed so much, I followed up later with an article on my improvements, including how I create a daily video archive: http://www.linuxjournal.com/content/birdcam-round-two.

If you don't want to take the time to read those articles, no worries. The TL;DR version is that I take periodic snapshots of my bird feeders, and have an auto-refreshing Web page that shows a 1–2FPS "video", which is hosted at http://birds.brainofshawn.com. (Don't worry about swamping my home connection; the feed is scaled to the

Figure 1. BirdCam now has far less clutter, but way more appeal.

cloud—the details of which are also in the initial article!)

New Hardware

This is by far the most frustrating development with my BirdCam setup. I want to move the bulk of the cameras outside so my office window isn't cluttered with wires, and also so the reflection of my new south-facing window doesn't cause J.J. Abrams-like lens flare. Unfortunately, finding an outdoor camera with the same image quality as my repurposed Galaxy S2 phone is difficult. I'm sure there are $500 Axis-brand cameras that have great image quality and an ability to pull still images, but that's way beyond my price range.

I have had decent luck with the $79 Foscam FI9803P camera. It does 720p, and with a convoluted URL, it's possible to pull still images from it. It struggles with lens flare much like my cameras pointing through the window, but it's tolerable. With an outdoor camera, I don't have the problem of window reflection recording *me* in my office as evening approaches. Nobody wants

You may recall one of the problems I had with BirdCam was to get JavaScript that would regularly refresh the page and work across multiple browsers and platforms.

to stare at me in the reflection, but with window-based cameras, that happens most evenings. I've contacted several companies selling outdoor-rated cameras, and most of them require the use of a proprietary app to view images.

I also attempted to use a Dropcam (Nest cam now I guess), and although the super-wide angle is nice, and it *is* possible to pull still images, the cloud-only factor is a showstopper for me. I don't want to use constant bandwidth to send video to the cloud when all I want is to pull still images. Perhaps there's a way to hack the firmware, but out of the box, a Dropcam uses too much bandwidth for my purposes. If anyone knows of an affordable 720p or 1080p outdoor security camera with optical zoom and the ability to pull stills via http(s), please let me know. I'm actually considering building a small heated box with a window in order to put an old Android phone outside. The photo quality on cell phones is so amazing, it will be hard to beat an Android device running IP Cam.

HTML Updates

You may recall one of the problems I had with BirdCam was to get JavaScript that would regularly refresh the page and work across multiple browsers and platforms. The code I posted last time did mostly work, but I managed to come up with JavaScript that is seemingly foolproof. I also updated the CSS to handle resizing of the main page better and split BirdCam into three separate pages. The result is clickable thumbnails that "refocus" the page. Listing 1 shows the format of each page, the only difference being that with the other two pages, the links and images are rearranged. I've added comments to each section explaining the functionality.

Image Changes

If you visit http://birds.brainofshawn.com, you might notice there no longer are text overlays with sunset info and so on. I might add the temperature back, but I found the rest to be overkill. Instead, there is a timestamp

Listing 1. house.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 ➥"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">

<!-- The above lines officially denote this as the
type of html file we need -->

<head>
<title>The birds. Or not.</title>

<!-- All these meta tags are to stop browsers and/or
servers from caching images. They are largely redundant,
but all are required because various browsers expect
different meta tags -->

<meta http-equiv="cache-control" content="max-age=0" />
<meta http-equiv="cache-control" content="no-cache" />
<meta http-equiv="expires" content="0" />
<meta http-equiv="expires" content="Tue, 01 Jan 1980
 ➥1:00:00 GMT" />
<meta http-equiv="pragma" content="no-cache" />
<meta http-equiv="Content-Type" content="text/html;
 ➥charset=iso-8859-1" />
<meta name="viewport" content="width=device-width;
 ➥initial-scale=1; minimal-ui">
</head>

<!-- The body and div tags specify the web page should
go edge to edge, and not stretch beyond 1280px -->

<body style="margin:0px;padding:0px;outline:none;border:0px">
<div style="width:100%;max-width:1280px;position:relative">

<!-- Each of these are thumbnails placed exactly in the
corners of the main image, they link to other pages which
are exactly like this page, but with the images reordered. -->
<a href="window2.html">
<img style="width:25%;max-width:1280px;z-index:200;position:
➥absolute;top:0px;left:75%" src="window2.jpg?time=" />
</a>
<a href="window.html">
<img style="width:25%;max-width:1280px;z-index:200;position:
➥absolute;top:0px;left:0%" src="window.jpg?time=" />
</a>

<!-- This image is the main image for the page. There's a
note to click the thumbnails to switch -->

<img style="width:100%;max-width:1280px;z-index:100;position:
➥relative;top:0px;left:0%" src="house.jpg?time=" />
<center><em>Click on smaller images to switch camera
 ➥views</em></center>
</div>

<!-- This Javascript is the magic. It basically reloads the
images every 2 seconds, adding ?time=TIMESTAMP to each image
so the server fetches it fresh. I can mostly follow its logic,
but I'm not sure why it works so much more consistently than my
previous attempts. Nevertheless, it seems perfect. -->

<script>
  setInterval(function() {
      var images = document.images;
      for (var i=0; i<images.length; i++) {
          images[i].src = images[i].src.replace(/\btime=[^&]*/,
           ➥'time=' + new Date().getTime());
      }
  }, 2000); // 10000 milliseconds = 10 seconds
  </script>
</body>
</html>
```

I think adding thumbnails to the bottom corners will keep the display fairly neat and provide easy access to more bird action.

on each photo in the corner, and that's it. That means my daily archive videos have a really nice "clock" you can watch while they play. If you head over to the BirdCam YouTube channel, you can see what I'm talking about (http://snar.co/windowcam). I've noticed with frustration that most of the video footage in the new house is "DogCam", because my dogs wander back and forth in front of the cameras constantly. We're planning to change where the dogs are allowed in the yard, so hopefully that will end soon.

With the simplification of images, I hope to add two more cameras eventually, if I find cameras I like. I think adding thumbnails to the bottom corners will keep the display fairly neat and provide easy access to more bird action. At least one of the new views will be a hummingbird camera once spring arrives, so check the feed periodically to see what happens in the future!

YouTube Updates

If you've tried to create an automated YouTube script based on my last BirdCam article, you probably noticed it no longer works. In fact, I didn't pay attention to the warning e-mail messages YouTube sent me, and I ended up getting locked out of my Google account for using an old API. Thankfully, I was able to recover my account without too much trouble, but still, it was scary. The ytu program (a script that uploads video to YouTube) itself needs to be updated, and the Google authentication process is different. Also, the command-line options for ytu have changed, and by default, videos are marked private. So in order to get an automated process, you'll need to follow these steps:

1) Head over to TASVideos and download YTU version 2: http://tasvideos.org/YoutubeUploader.html.

2) Extract the tar.bz2 file and read the "Obtaining Credentials.pdf" file. It will walk you through the (*required!*) process of getting the credentials that will allow ytu to work. It requires you to create a new Google developers

project. Thankfully, the instructions are easy to follow.

3) Modify (or create!) your scripts to upload the video to YouTube. Here is what my modified code looks like:

```
#!/bin/bash
/usr/local/bin/ytu/ytu \
     -permission ap \
     /processed_video/birdvideo.mp4 \
     "Video Title Here" \
     "Tags,comma,separated" \
     1 \
     < /usr/local/etc/description.txt
```

Notice the `-permission ap`, which makes the video public. Supposedly videos are public by default, but that wasn't the case for me, and I had to add that option manually. Also, the 1 on the line by itself is the category ID. 1 is film and animation, but ytu will give you a listing in the documentation if you want something different.

The mencoder stuff is all exactly the same as before. Luckily, turning thousands and thousands of images into a video is still really easy. Be sure to read my last BirdCam article for more details.

Compensating for the Earth's Tilt

It has annoyed me for a couple years now that I basically had to guess when to turn the cameras on and off for the night. In fact, I usually just leave things on overnight, and hope the lack of light stops the cameras from recording any motion. A few 12-hour videos of my screensaver reflected in the window caused me to research the problem a little more. Here's what I came up with.

There's a cool program called sunwait developed by Dan Risacher in 2004 that calculates sunrise/sunset times based on latitude and longitude. It's no longer being developed, but it compiles easily and works well even with modern distributions. You can get the program from https://www.risacher.org/sunwait.

The program does two things for me:

1) It gives the sunset/sunrise data, which I then can update daily and store in my /etc folder, which I use as a test to determine whether I should download images from my cameras.

2) It literally will "wait" on the command line when invoked until sunrise or sunset. This is useful because I have it "wait" for sunset, then kill off the motion program. Then in the morning, I have it "wait" for sunrise and start motion back up.

The simplicity is brilliant, and it has required me to change my scripts a bit. Here's the script I use to get sunrise and sunset times. Specifically, I

pick "Civil" sunrise and sunset, which aligns with "when you can still see" rather than the actual time the sun sets. For camera purposes, it's far more useful. I call the program "sun" in the /usr/local/bin folder:

```
#!/bin/bash
if [ $1 == "-rise" ]
then
/usr/local/bin/sunwait -p 45.3733N 84.9553W |
 ➥grep Civil | awk -F" " '{print $4}'
elif [ $1 == "-set" ]
then
/usr/local/bin/sunwait -p 45.3733N 84.9553W |
 ➥grep Civil | awk -F" " '{print $7}'
else
echo 'Type either "sun -rise" or "sun -set"'
fi
```

Then in the root user's crontab, I have these two entries:

```
1 4 * * * /usr/local/bin/sun -rise > /etc/sunrise
2 4 * * * /usr/local/bin/sun -set > /etc/sunset
```

So every morning around 4am, the sunrise and sunset data is updated on my computer. Then I have two crontab entries as my normal birdcam user, which starts and stops motion at the proper time:

```
0 16 * * * /usr/local/bin/sunwait civ down 45.3733N
 ➥84.9553W; pkill motion
0 5 * * * /usr/local/bin/sunwait civ up 45.3733N 84.9553W;
 ➥pkill motion; sleep 10; motion
```

Those lines start sunwait well in advance of any sunset or sunrise, and start or kill motion accordingly at the right time. (The start script kills off motion, in case the computer was restarted and motion already is running.) Finally, the birdcam user has one more cronjob that puts "off-line" photos for the Web server to use:

```
2 16 * * * /usr/local/bin/sunwait civ down 45.3733N 84.9553W;
 ➥sleep 30; cp /offline.jpg > /dev/shm/house.jpg
```

As far as the script that fetches images from my IP camera, I have a conditional loop that runs like the following:

```
#!/bin/bash

#Variables -- change to fit your needs
TEMP_PHOTO=/dev/shm/.housetemp.jpg
SUNRISE=`cat /etc/sunrise`
SUNSET=`cat /etc/sunset`
TIME=`date +%k%M`
# Get Photos, or offline photos if cameras offline

if [ $TIME -gt $SUNRISE -a $TIME -lt $SUNSET ]
then
   if eval "ping -c 1 192.168.1.178 > /dev/null"
   then
     /usr/bin/wget -r --timeout=5 --quiet -O \
     $TEMP_PHOTO \
```

```
    "http://192.168.1.178:88/cgi-bin/CGIProxy.fcgi?cmd=
➥snapPicture2&usr=admin&pwd=passwd"
    mv $TEMP_PHOTO /dev/shm/house.jpg
  fi
sleep 2
rm -rf /dev/shm/.house*
fi
```

What does all that do? Well, if you follow the logic in the cronjobs and scripts, you should find that at sunrise every day, the motion program is started, and my script for downloading images from the IP camera starts fetching images. At sunset, motion is killed off, and the off-line image is put in place. Since this updates every day, my camera follows the pattern of the sun throughout the year, even honoring Daylight Savings Time!

Your Own Project

It's very unlikely you're as nuts about birds as I am. Thankfully, much of the stuff I use for BirdCam easily can be adapted to other projects. I'm absolutely loving the ability to schedule things according to the sunrise and sunset at my locale. If nothing else, that is a powerful tool for managing scripts. Hopefully you're able to get as much joy from creating your own projects as I get from BirdCam! (And really, don't hesitate to e-mail me about good IP cameras, I'm having a tough time coming up with decent options.)■

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.

SUSAN SONS

Chain of Custody

For all users who have wondered "did someone backdoor this?", there should be developers ensuring that the code they put out into the world can be verified and tampering detected. Package maintainers and users also must exercise diligence in order to avoid running untrusted code. This article walks you through the chain of custody between a hypothetical OSS project's developers and users, explaining what could go right or wrong at each step.

There's a great deal to be said for secure coding practices. However, if the program the user receives is not the one the developer created—complete and unchanged—those secure coding practices may not matter. In this article, I follow the paths that a hypothetical piece of software, foobard, may take from its development team to its users, describing how that path can be exploited and how it can be protected.

Alice and Bob are great at coding. They maintain a robust test suite and accept only patches that pass all tests. They regularly fuzz test the application as a whole and use static analysis tools to alert them of potential flaws in their code. Their architecture is extremely well thought out, and their choices in dependencies are sane. Throughout these examples, I'm assuming that foobard, as written by Alice and Bob, does not present any unknown security risks. Unfortunately, there are many places that this can fall apart before foobard reaches users.

Alice and Bob are using CVS to maintain foobard. After all, it's not a huge project, and CVS is what they have always used. The server that hosts the foobard CVS repo, however, was compromised and began serving up spyware tarballs

on one of its Web pages. Alice and Bob don't know exactly what access the attacker achieved or how long ago the compromise happened, so they can't trust their server backups to have an unmodified copy of the repo. CVS offers no built-in integrity checking mechanism for the code itself, and modifying CVS history is trivial. Alice and Bob can try to cobble together—from whatever is stored on their laptops or in other theoretically safe locations—enough data to spot-check the foobard repo and ensure that none of their code has been changed by the attacker. However, spot-checking provides little guarantee about the code's overall integrity and won't help reconstruct a full, known-good history.

If that sounds far-fetched, consider that even the server owner may be the attacker. You may remember that popular open-source host SourceForge was caught changing a hosted project's installer to install malware in addition to the requested software (see Resources for more information).

This could have been prevented if Alice and Bob were using a modern source code management tool such as Git or Mercurial, both of which use hashes to identify commits, and both of which allow code signing. In Git, you GPG-sign a tag, and in Mercurial, you GPG-sign the manifest key in a changelog entry. In either case, that signature can be used not just to verify the integrity of one commit, but also of that commit and all of its ancestors. This doesn't mean there is no way to corrupt the authoritative repository on the server, but when best practices are used, it becomes astronomically difficult for attackers to hide that corruption, requiring a timed compromise of multiple machines.

This protection, of course, relies in part on the secrecy of the private GPG key(s) used for signing tags (or manifests). If Alice or Bob loses a copy of such a private key, it must be revoked and replaced as soon as possible, before an attacker has had time to brute-force the key's passphrase.

Now that that's sorted, with Alice and Bob migrating to Git and tagging releases with GPG-signed tags, they've increased the security of one link in the chain. I'll go so far as to assume that, having learned this lesson, Alice and Bob also learned to sign any release tarballs they offer. By changing these two practices, Alice and Bob also have mitigated some risks from unreliable DNS (when one can verify the code itself, one need not care if it came from the expected URL) and potential SSL issues (for

the same reason: they're checking the code not trusting its origin). Another member of the Open Source community, Carol, now can get a known-good copy of the foobard source. Of course, before she can use foobard, Carol needs to build it.

The build scripts for foobard include checking for, and if need be retrieving, several dependencies. Although these dependencies were well chosen, foobard's build system will retrieve and build these packages blindly without checking their integrity at all. This is arbitrary code execution with the permissions of whatever user ran foobard's build script. Users' ISPs already are injecting ads into Web sites using their position between users and the Internet, so there is no reason to believe that they (or a state actor, or a DNS registrar, or a router manufacturer, or a server compromise) never will cause you to grab something other than the dependencies you expected.

To solve this, Alice and Bob have two choices:

1. Ensure that the build script exits with an explanatory error when a dependency is not found locally, so that Carol can get dependencies in her usual, probably sane, way.
2. Ensure that the build script does appropriate integrity checking of any dependencies it downloads, *and* that any dependencies' build scripts do the same, all the way down the dependency tree.

Let's assume that Alice and Bob chose option one, as it's by far the least laborious. Now, in theory, Carol can get a known-good copy of foobard and build it without running or installing software of unknown origin on her machine. This is good, because once the machine doing the compiling is compromised, the binary cannot be trusted (nor can anything else on that system). They are depending on either Carol or some tool she runs to check the signatures on the code she downloads.

Carol, it turns out, is a package maintainer for a binary Linux distribution. It doesn't matter which one for the purposes of this article. Now that she has gotten a known-good copy of foobard, and known-good copies of all relevant dependencies, and has built foobard, Carol is packaging it up for a repository that will provide the prebuilt binary to thousands of users. She should, in turn, ensure that the packages she generates are signed before being

Major distributions, such as Red Hat, Fedora and Debian, for example, do sign official packages cryptographically, and their package managers reject packages with bad signatures.

passed on to package mirrors.

The state of things at the time of this writing (mid-September 2015) is that binary Linux distributions vary in how they check the integrity of the software that they package. Major distributions, such as Red Hat, Fedora and Debian, for example, do sign official packages cryptographically, and their package managers reject packages with bad signatures. Gentoo uses a Git-backed package management strategy that signs commit hashes rather than individual packages, achieving the same general effect plus protection of the package metadata and prevention of metadata replay attacks. However, the source code those ebuilds retrieve is not checked, as far as I can tell.

None of these Linux distributions have published policies that I can find that would bar the signing and distribution of packages for code that was not signed by its developer, or that pulls in unsigned code or binaries at build time. In short, most package managers are verifying the authenticity of packages, but package management teams don't seem to be differentiating between packages made from known-good code and packages made from code of which they cannot verify the integrity.

To the best of my knowledge, current package managers still consider "valid code signing key" to be a binary property. That is, a code signing key either is considered valid by your package manager for signing any package, or is not considered valid at all. As such, people who maintain a portage overlay (or deb/rpm repository) with your favorite game in it could sign (or their compromised key could sign) binutils or sudo. So, package maintainers who think their packages' importance is not high enough to merit a diligent approach to information security may cause your system to replace crucial system utilities typically run as root or capable of mediating root access.

Linux and other open-source

software is used around the world: in medical care, the power grid, the Internet and countless other bits of infrastructure that we rely on every day. Luckily, it's possible to make the kinds of software supply chain attacks described here incredibly difficult to pull off. Doing so will take concerted effort by developers, distribution maintainers (both packagers and maintainers of the packaging systems), as well as users.

OSS Developers Should:

- Use a source control system with integrated integrity checking, such as Git or Mercurial, for managing all projects.
- Cryptographically sign each release in the source control system (via tag or equivalent) and each release tarball.
- Carefully safeguard their private keys: both code-signing keys and the SSH keys used to commit code.
- Rapidly revoke and replace keys that may be compromised. Remember: new GPG/SSH keys are free; the damage to your project's reputation if compromised code goes out with your valid signature is irreversible.
- Ensure that the build system generates errors for missing dependencies, rather than blindly downloading and building them without integrity checking.
- Get their GPG keys signed by other developers, and in turn, sign those developers' keys, so that users have a better idea of which GPG keys to trust.
- Choose dependencies with similarly good distribution practices, and file bugs with dependencies that are not following these recommendations.

Linux Distributions Should:

- Use caution in obtaining source code for generating packages, checking that the code is signed by a trusted key and not building against any untrusted code such as something downloaded by the build system without integrity checking.
- Make contact with upstream

developers when public Git/Mercurial history changes to ensure that the change was expected and not a sign of tampering.

- File bugs with upstream developers who do not use modern source control systems and/or don't cryptographically sign releases.

- Never accept packages that are not cryptographically signed by the package maintainer.

- Set a date to stop packaging code that was not signed by its development team, communicate that date upstream and stick to it.

- Ensure that the package manager checks signatures on all packages it retrieves, and that it checks for revocation of package-signing keys.

- Check the cryptographic signatures of any additional files that a package may download.

- Ensure that the package manager warns the user if a package's integrity cannot be verified, either due to a failed signature check or to the package relying on some resource (such as a proprietary blob from a third-party site) that is not signed at all.

- Design package management tools that allow a particular package-signing key to be valid only for certain packages.

Users Should:

- Be suspicious of any program not signed by its developer (or package maintainer), whether that software is open source or being distributed as a compiled binary. Ideally, one never would run unsigned code at all. However, in applications that are not life-critical, one may need to compromise at minimizing the amount of unsigned code in use, and not running unsigned code as root.

- Exercise due diligence in obtaining source code to compile: check that the code is signed by a reasonably trusted key and does not download anything at build time without authenticating it.

- File bugs with developers who do not use modern source control systems and/or do not cryptographically sign releases.

- Not enable package repositories if those repositories' maintainers are not signing packages, or if the maintainers' keys can't be verified.

Some of these things are being done most of the time, and the overall picture is improving. Running software inevitably involves trust, as no one has both the time and the skill to audit every piece of code running on their systems. We can do a better job of making sure that we only trust code that came from the people we think it came from.■

Susan Sons serves as a Senior Systems Analyst at Indiana University's Center for Applied Cybersecurity Research (http://cacr.iu.edu), where she divides her time between helping NSF-funded science and infrastructure projects improve their security, helping secure a DHS-funded static analysis project, and various attempts to save the world from poor information security practices in general. Susan also volunteers as Director of the Internet Civil Engineering Institute (http://icei.org), a nonprofit dedicated to supporting and securing the common software infrastructure on which we all depend. In her free time, she raises an amazing mini-hacker, writes, codes, researches, practices martial arts, lifts heavy things and volunteers as a search-and-rescue and disaster relief worker.

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

Resources

"SourceForge grabs GIMP for Window's account, wraps installer in bundle-pushing adware" by Sean Gallagher: http://arstechnica.com/information-technology/2015/05/sourceforge-grabs-gimp-for-windows-account-wraps-installer-in-bundle-pushing-adware

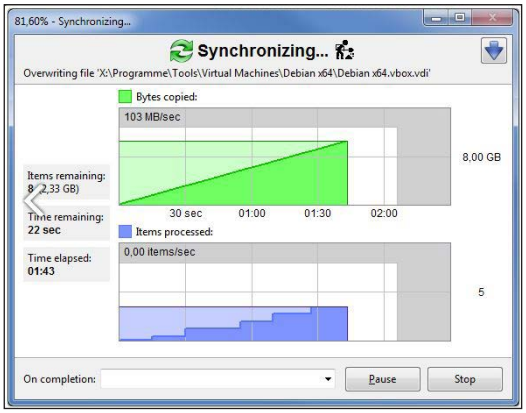
"How a banner ad for H&R Block appeared on apple.com—without Apple's OK" by Nate Anderson: http://arstechnica.com/tech-policy/2013/04/how-a-banner-ad-for-hs-ok

"XcodeGhost hack: Delete infected iOS apps immediately" by Buster Hein: http://www.cultofmac.com/389693/xcodeghost-hack-delete-these-infected-ios-apps-immediately

"Reflections on Trusting Trust" by Ken Thompson: https://www.ece.cmu.edu/~ganger/712.fall02/papers/p761-thompson.pdf

Bitwig GmbH's Bitwig Studio

Bitwig Studio is Bitwig GmbH's software solution for empowering music enthusiasts to realize their musical ideas at every stage of production. Available for Linux, Mac OS X and Windows, the updated Bitwig Studio 1.3 most prominently features complete multi-touch functionality for quick identification of gestures for the most unique and intuitive workflow ever. With multi-touch, users control multiple faders, knobs and device displays at the same time. Other innovations in v1.3 include a radial menu for quick access to multiple actions, an integrated keyboard that includes independent X/Y axis controls for each finger and a new e-cowbell—because when you've got a fever, the only cure is...more cowbell!
http://www.bitwig.com



FreeFileSync

The new version 7.5 is the latest rendition of FreeFileSync, a free and open-source utility that synchronizes files and folders for "all modern versions" of Linux, Windows and Mac OS X. FreeFileSync is designed to save users' time setting up and running backup and sync jobs while enjoying useful visual input along the way. The software has been optimized for both CPU and file I/O performance, enabling it to scan a hard drive with hundreds of thousands of files in seconds. The fail-safe file copy design includes multiple strategies to prevent data corruption if the synchronization process is interrupted. The latest version 7.5 now supports the SFTP protocol, adding the unique benefit of eliminating the tedious and error-prone task of manually identifying files that have changed on the source computer and moving them to the target computer. Another new feature is the detection of moved files on the source computer, even for targets with no file-id support or with unstable file-id support. Finally, enhanced media transfer protocol (MTP) support makes it even easier for users to synchronize files and folders between their PCs and their other MTP devices.
http://www.freefilesync.org

Jedox 6

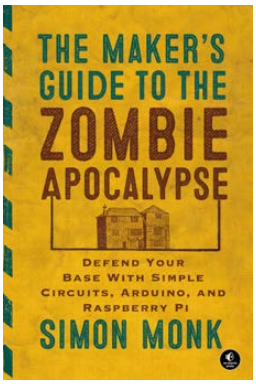
With the new Jedox 6, vendor Jedox says it has managed to distill the essence of its unified business intelligence (BI) and planning platform, accelerate it and make it easier to use. Jedox is a unified planning, analysis and reporting platform that empowers decision-makers from finance, sales, purchasing and marketing to work smarter, streamline business collaboration and make insight-based decisions with confidence. Jedox 6 improves the BI platform by including connectors for Salesforce, SAP HANA, Hadoop and the Qlik social collaboration platform, as well as by providing a completely new mobile experience. Meanwhile, speed, enterprise scalability and usability are enhanced due to the new integrated Dynamic Data Engine. The improved mobile experience stems in large part from the integration of the newly acquired Reboard mobile BI platform. Because Jedox 6 is available as SaaS and on-premise, it completely supports companies transitioning to the cloud.
http://www.jedox.com

New Spaces AG's Gravit.io

The Principality of Liechtenstein may have fewer people than the pueblo of Bountiful, Utah, but it can boast innovations like New Spaces AG's Gravit.io, "the world's first Web-based pixel design tool". Gravit offers features that will allow anyone—from beginner to professional—to start designing free of charge. Examples include a company logo, Facebook timeline or graphics for a Web site. The site has basic features that cater to those who are unfamiliar with design software as well as advanced features that a professional would be happy to use for everyday work. This new version 2.0 makes it easier for users to create, edit and manage their designs, and it includes additions like the professional Bezigon Pen and more than two dozen photo filter options. Gravit's developers say that its creation is unlike other design software and services in that it is more than a pixel design tool. It also is a cloud-based design platform that allows users to design, share their work and interact with other designers and community members as part of their regular workflow.
http://about.gravit.io

Varnish Software's Varnish Cache

Headlining the innovations in the new Varnish Cache 4.1 open-source HTTP engine are improved security and proxy support. These advances enhance the recently added streaming architecture, which cuts down delivery times for larger objects and decreases latency when accessing content through cache hierarchies. A noteworthy new security feature is support for different kinds of privilege separation methods, collectively described as jails. On the topic of proxy protocol support, version 4.1 provides socket support for PROXY protocol connections, whereby PROXY defines a short preamble on the TCP connection where typically an SSL/TLS terminating proxy can signal the real client address. Varnish Software notes that more than 2.2 million Web sites use its HTTP accelerator, including 14% of the top global 10,000 sites.
http://www.varnish-software.com



Simon Monk's *The Maker's Guide to the Zombie Apocalypse* (No Starch Press)

Author, hardware hacker and zombie anthropologist Simon Monk wants to know where his fellow makers will be when the zombie apocalypse hits. Trapping yourself in the basement? Roasting the family pet? Beheading re-animated neighbors? "No way!" offers Monk. With his guidance, you'll be building fortresses, setting traps and hoarding supplies, because you, savvy survivor, have snatched up your copy of *The Maker's Guide to the Zombie Apocalypse* before it's too late. Subtitled *Defend Your Base with Simple Circuits, Arduino, and Raspberry Pi*, this indispensable guide to survival after Z-day will teach apprentice zombie anthropologists how to generate their own electricity, salvage parts, craft essential electronics and out-survive the undead. Readers will learn myriad survival skills. They will take charge of their environment—for example, by powering zombie defense devices with car batteries, bicycle generators and solar power. They will escape imminent danger—for example, by repurposing old disposable cameras for zombie-distracting flashbangs. And they will communicate with other survivors—for example by passing silent messages with two-way vibration walkie-talkies. Survival of the zombie apocalypse calls for these and many more essential, life-saving measures.
http://www.nostarch.com

Neverware's CloudReady

Helping schools extend the life of their hardware is one key objective of Neverware and its Google-supported CloudReady OS, which turns almost any existing PC or Mac into a fully functional Chromebook. The new free version of CloudReady has been certified by Neverware on nearly 200 computer models and can be used on hundreds of others. Automatic updates are included, and simple installation occurs from a USB thumb drive. Because CloudReady is based on the same code that powers Chromebooks, it offers complete and secure integration with Google Apps and other Google services. A previously released paid version of CloudReady also exists, adding a complement of dedicated support and integration with Google's device management console. Neverware says that numerous school districts have expressed satisfaction that CloudReady has appreciably increased their device-to-student ratios.

http://www.neverware.com

Red Hat Software Collections

In order to provide developers with ready access to applications needing software components in order to utilize their newest features, Red Hat maintains Red Hat Software Collections. This Red Hat offering, now at version 2.1, is a package of essential open-source Web development tools, dynamic languages, databases and a variety of development and performance management tools. The content is either more recent than equivalent versions included in the base Red Hat Enterprise Linux (RHEL) system or is new to RHEL. Red Hat Software Collections 2.1 includes Red Hat Developer Toolset 4, which has the latest stable open-source C and C++ compilers and dev tools. For developers following the rapid development and deployment cycles inherent to Linux containers, many of the most popular Red Hat Software Collections also have been made available as Dockerfiles and/or Docker-formatted container images via the Red Hat Customer Portal.

http://www.redhat.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.

What's in the Box? Interrogate Your Linux Machine's Hardware

Shell and graphic utilities for hardware detection. **FEDERICO KEREKI**

I recently had a problem trying to install the NVIDIA driver for my machine. It seemed the latest driver had stopped supporting my graphics card, and after updating my kernel, I was out of a driver. The question, obviously, was "which card did I have?" But, I didn't remember. If you have to name the chipset of your motherboard, specify the CPU in your box or get any other kind of hardware-related information, Linux provides several utilities to help you. In my case, I quickly could get the full ID of my graphics card, confirm that it really was getting a bit long in the tooth and decide that a newer one wasn't such a bad idea.

In this article, I discuss several ways of getting hardware data for your machine. In the most time-honored Linux shell way, I show how to work with several command-line utilities, but if you prefer using a GUI, I also include some graphical programs. And, if you want to get into the nitty-gritty details, I give some pointers on how to get that information by using the /proc or /sys filesystem.

The `ls` Command Family

Let's start the command-line work with a set of several utilities, whose names all start with `ls` (Table 1). Some of these commands provide overlapping

Glossary

Working with hardware means dealing with several acronyms, and I must admit, I had been using at least a couple of them without remembering precisely what they meant. Here's a list of definitions you'll surely need:

- ACPI (Advanced Configuration and Power Interface): related to power aspects.
- AGP (Accelerated Graphics Port): a channel to allow attaching a video graphics card (not typically seen since around 2008).
- APM (Advanced Power Management): older than ACPI, also related to power issues.
- ATA (AT Attachment): "AT", as in the old IBM AT, a standard to connect storage devices, superseded by SATA in 2003.
- BIOS (Basic Input/Output System): firmware used when booting an Intel-compatible PC.
- DMA (Direct Memory Access): a feature that allows giving hardware access to RAM, independently of the CPU.
- DMI (Desktop Management Interface): a framework for keeping track of devices in a computer.
- IDE (Integrated Drive Electronics): an interface standard that later evolved into ATA.
- IRQ (Interrupt ReQuest): a hardware signal that allows an interrupt handler to process a given event.
- PCI (Peripheral Component Interconnect): a bus standard for attaching varied hardware devices to a computer, created in 1992.
- UEFI (Unified EFI—Extensible Firmware Interface): a 2005 replacement for BIOS, which deprecated the previous 1998 EFI standard.
- USB (Universal Serial Bus): a standard bus defined in 1995 to allow connecting all kinds of peripherals to a computer.
- PATA (Parallel ATA): the new name for ATA, after SATA came out.
- PCIe (PCI Express): a high-speed serial bus that replaced PCI and AGP in 2004.
- RAID (Redundant Array of Independent—originally, "Inexpensive"—Disks): a data storage virtualization technology that combines several drives to work as a single one for performance improvement and/or data redundancy. There are several RAID schemes, including RAID 0 ("striping"), RAID 1 ("mirroring"), RAID 5 ("striping + parity") and RAID 10 ("striping + mirroring").
- SATA (Serial ATA): a bus interface to connect storage devices, currently used in practically all PCs.
- SCSI (Small Computer System Interface—pronounced "scuzzy"): a set of standards for connection of devices and transfer of data between computers and peripherals.

Table 1. The ls* family of commands lets you access all aspects of your hardware.

COMMAND	DESCRIPTION
`lsblk`	Produces information about all block devices, such as hard disks, DVD readers and more.
`lscpu`	Shows information like number of CPUs, cores, threads and more.
`lsdev`	Displays data about all devices of which the system is aware.
`lshw`	Lists general hardware data—gives information on every detail of your hardware.
`lspci`	Displays information about PCI buses in your box and devices connected to them, such as graphics cards, network adapters and more.
`lsscsi`	Provides information on all SCSI devices or hosts attached to your box, such as hard disk drives or optical drives.
`lsusb`	Generates information about USB buses in your box and devices connected to them.

information (`lsdev` or `lshw`, for instance), but by using all of them, you can get a pretty clear idea of whatever may be inside your Linux box.

Let's start with CPU information. The `lscpu` command provides data on the CPUs in your box. You can opt to include all CPUs, whether off-line or on-line, with the `-.all` parameter, or you can select `--online` and `--offline`. The `--parse` option lets you choose what CPU characteristics to list, including number, socket, cache data, maximum and minimum speed (in MHz) and more. In my case, you'll see that my machine has a somewhat old single-socket, four-core, Intel Core 2 Quad CPU, at 2.66GHz:

```
> lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                4
On-line CPU(s) list:   0-3
Thread(s) per core:    1
Core(s) per socket:    4
Socket(s):             1
NUMA node(s):          1
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 23
Model name:            Intel(R) Core(TM)2 Quad CPU Q8400 @ 2.66GHz
Stepping:              10
CPU MHz:               2003.000
CPU max MHz:           2670.0000
CPU min MHz:           2003.0000
BogoMIPS:              5340.67
Virtualization:        VT-x
L1d cache:             32K
L1i cache:             32K
L2 cache:              2048K
NUMA node0 CPU(s):     0-3
```

(Note: you can get most of this information by examining the /proc/cpuinfo file or by browsing the /sys/bus/cpu/ directories; see the DIY with /proc and /sys sidebar for more on this.)

Let's move on to block devices, such as hard disks, or CD and DVD units. The `lsblk` command produces information on all available block devices (see Listing 1 for an example). As you can see, I have three hard disks and a ROM (DVD) device. The three disks are known as /dev/sda, /dev/sdb and /dev/sdc; the ROM device is /dev/sr0. The disks are 466GB, 149GB and 2.7TB in size. You can get a little information about partitioning too; for instance, you can see that the first two disks have a swap area enabled, but the third one doesn't. You also can get the mountpoints (/, /disk-laptop and /disk-data) for the three disks.

Listing 1. The `lsblk` command shows all block (storage) devices. The `--topology` option adds extra details; try `--output-all` for even more.

```
> lsblk --paths
NAME        MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
/dev/sda      8:0    0 465.8G  0 disk
|__/dev/sda1  8:1    0     4G  0 part [SWAP]
|__/dev/sda2  8:2    0 461.8G  0 part /
/dev/sdb      8:16   0 149.1G  0 disk
|__/dev/sdb1  8:17   0     4G  0 part [SWAP]
|__/dev/sdb2  8:18   0   145G  0 part /disk-laptop
/dev/sdc      8:32   0   2.7T  0 disk
|__/dev/sdc1  8:33   0   2.7T  0 part /disk-data
/dev/sr0     11:0    1  1024M  0 rom

> lsblk --paths --topology
NAME   ALIGNMENT MIN-IO OPT-IO PHY-SEC LOG-SEC ROTA SCHED RQ-SIZE  RA WSAME
sda            0    512      0     512     512    1 cfq       128 128    0B
|__sda1        0    512      0     512     512    1 cfq       128 128    0B
|__sda2        0    512      0     512     512    1 cfq       128 128    0B
sdb            0    512      0     512     512    1 cfq       128 128    0B
|__sdb1        0    512      0     512     512    1 cfq       128 128    0B
|__sdb2        0    512      0     512     512    1 cfq       128 128    0B
sdc            0   4096      0    4096     512    1 cfq       128 128    0B
|__sdc1        0   4096      0    4096     512    1 cfq       128 128    0B
sr0            0    512      0     512     512    1 cfq       128 128    0B
```

There are many possible optional arguments, but the most typically used are `--paths`, which produces full device paths, and `--topology`, if you are interested in internal details, such as physical sector size, I/O scheduler name and so on. You can get owner, group and permissions information with `--perm`, as shown below (and, if you really want detailed information, try `--output-all`, which will list about 50 columns' worth of data):

```
> lsblk --perm
NAME      SIZE OWNER GROUP MODE
sda     465.8G root  disk  brw-rw----
|__sda1     4G root  disk  brw-rw----
|__sda2 461.8G root  disk  brw-rw----
sdb     149.1G root  disk  brw-rw----
|__sdb1     4G root  disk  brw-rw----
|__sdb2   145G root  disk  brw-rw----
sdc       2.7T root  disk  brw-rw----
|__sdc1   2.7T root  disk  brw-rw----
sr0      1024M root  cdrom brw-rw----
```

For SCSI devices, you can add `--scsi` to `lsblk`, but there's also the more specific `lsscsi` command. The basic information it produces is shown below, and it includes all available SCSI devices. In my case, it shows the three hard disks and the optical reader I already found with `lsblk`, plus three card readers. Note that you also get more information on specific brands and models. For example, I have two Western Digital hard drives (WD5000AAKS and WD30EZRX), plus a Maxtor laptop drive (STM316021) and a Sony AD-7200S DVD unit:

```
> lsscsi
[2:0:0:0] disk   ATA  WDC WD5000AAKS-0 1D05 /dev/sda
[2:0:1:0] disk   ATA  MAXTOR STM316021 D    /dev/sdb
[3:0:0:0] disk   ATA  WDC WD30EZRX-00M 0A80 /dev/sdc
[3:0:1:0] cd/dvd SONY DVD RW AD-7200S  1.61 /dev/sr0
[4:0:0:0] disk   Sony Card_R/W     -CF 1.11 /dev/sdd
[4:0:0:1] disk   Sony Card_R/W     -SD 1.11 /dev/sde
[4:0:0:2] disk   Sony Card_R/W     -MS 1.11 /dev/sdf
```

Check out all the possibilities of this command with `lsscsi --help`. You'll see that you really can dig down into SCSI devices with it. And, if you're interested, this command works by scanning the /sys filesystem (see Resources, and the DIY with /proc and /sys sidebar for more information).

Now, let's move on to some other commands. `lsusb` provides information on all USB-connected devices; see Listing 2 for an example. (An alternative is `usb-devices`, but it's somewhat more obscure in its output and has no configuration options.) As in most modern computers, you'll probably have a

lot of such devices. In my case, I have a Bluetooth dongle, Webcam, keyboard, mouse and more. You can get information on a specific bus or device with the `-s` option or select a given vendor with the `-d` option; for the latter, check the USB ID repository (see Resources) for vendor/device numbers. Finally, if you want very detailed information, try the `-v` (verbose) option, but be prepared to read a lot. For my machine, `lsusb -v` produces more than 1,300 lines of output.

Another command that can produce a ton of information is `lspci`, which shows all data on

Listing 2. The `lsusb` command reports all USB-connected devices, as a list or in tree form.

```
> lsusb
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation
 ➥2.0 root hub
Bus 005 Device 002: ID 054c:01bd Sony Corp. MRW62E
 ➥Multi-Card Reader/Writer
Bus 005 Device 001: ID 1d6b:0001 Linux Foundation 1.1
 ➥root hub
Bus 004 Device 001: ID 1d6b:0001 Linux Foundation 1.1
 ➥root hub
Bus 003 Device 002: ID 0a12:0001 Cambridge Silicon
 ➥Radio, Ltd Bluetooth Dongle (HCI mode)
Bus 003 Device 006: ID 1e4e:0100 Cubeternet WebCam
Bus 003 Device 005: ID 046d:c317 Logitech, Inc.
 ➥Wave Corded Keyboard
Bus 003 Device 004: ID 04f3:0232 Elan
 ➥Microelectronics Corp. Mouse
Bus 003 Device 003: ID 05e3:0608 Genesys Logic,
 ➥Inc. Hub
Bus 003 Device 001: ID 1d6b:0001 Linux Foundation
 ➥1.1 root hub
Bus 002 Device 001: ID 1d6b:0001 Linux Foundation
 ➥1.1 root hub

> lsusb --tree
/:  Bus 05.Port 1: Dev 1, Class=root_hub, Driver=uhci_hcd/2p, 12M
    |__ Port 2: Dev 2, If 0, Class=Mass Storage,
         ➥Driver=usb-storage, 12M
/:  Bus 04.Port 1: Dev 1, Class=root_hub, Driver=uhci_hcd/2p, 12M
/:  Bus 03.Port 1: Dev 1, Class=root_hub, Driver=uhci_hcd/2p, 12M
    |__ Port 1: Dev 3, If 0, Class=Hub, Driver=hub/4p, 12M
        |__ Port 1: Dev 4, If 0, Class=Human Interface Device,
             ➥Driver=usbhid, 1.5M
        |__ Port 2: Dev 5, If 0, Class=Human Interface Device,
             ➥Driver=usbhid, 1.5M
        |__ Port 2: Dev 5, If 1, Class=Human Interface Device,
             ➥Driver=usbhid, 1.5M
        |__ Port 3: Dev 6, If 0, Class=Video, Driver=uvcvideo, 12M
        |__ Port 3: Dev 6, If 1, Class=Video, Driver=uvcvideo, 12M
    |__ Port 2: Dev 2, If 0, Class=Wireless, Driver=btusb, 12M
    |__ Port 2: Dev 2, If 1, Class=Wireless, Driver=btusb, 12M
/:  Bus 02.Port 1: Dev 1, Class=root_hub, Driver=uhci_hcd/2p, 12M
/:  Bus 01.Port 1: Dev 1, Class=root_hub, Driver=ehci-pci/8p, 480M
```

PCI devices. And, as a matter of fact, this is the actual command I used to remember what kind of graphics card I had:

```
# lspci
00:00.0 Host bridge: Intel Corporation 4 Series
 ➥Chipset DRAM Controller (rev 03)
00:01.0 PCI bridge: Intel Corporation 4 Series
 ➥Chipset PCI Express Root Port (rev 03)
00:1b.0 Audio device: Intel Corporation NM10/ICH7
 ➥Family High Definition Audio Controller (rev 01)
00:1c.0 PCI bridge: Intel Corporation NM10/ICH7
 ➥Family PCI Express Port 1 (rev 01)
00:1c.1 PCI bridge: Intel Corporation NM10/ICH7
 ➥Family PCI Express Port 2 (rev 01)
00:1d.0 USB controller: Intel Corporation NM10/ICH7
 ➥Family USB UHCI Controller #1 (rev 01)
00:1d.1 USB controller: Intel Corporation NM10/ICH7
 ➥Family USB UHCI Controller #2 (rev 01)
00:1d.2 USB controller: Intel Corporation NM10/ICH7
 ➥Family USB UHCI Controller #3 (rev 01)
00:1d.3 USB controller: Intel Corporation NM10/ICH7
 ➥Family USB UHCI Controller #4 (rev 01)
00:1d.7 USB controller: Intel Corporation NM10/ICH7
 ➥Family USB2 EHCI Controller (rev 01)
00:1e.0 PCI bridge: Intel Corporation 82801 PCI
 ➥Bridge (rev e1)
00:1f.0 ISA bridge: Intel Corporation 82801GB/GR
 ➥(ICH7 Family) LPC Interface Bridge (rev 01)
00:1f.1 IDE interface: Intel Corporation 82801G (ICH7
 ➥Family) IDE Controller (rev 01)
00:1f.2 IDE interface: Intel Corporation NM10/ICH7
 ➥Family SATA Controller [IDE mode] (rev 01)
00:1f.3 SMBus: Intel Corporation NM10/ICH7 Family
 ➥SMBus Controller (rev 01)
01:00.0 Ethernet controller: Qualcomm Atheros AR8152
 ➥v2.0 Fast Ethernet (rev c1)
04:00.0 VGA compatible controller: NVIDIA Corporation
 ➥GK107 [GeForce GT 740] (rev a1)
04:00.1 Audio device: NVIDIA Corporation GK107 HDMI
 ➥Audio Controller (rev a1)
```

Try the `-v` or `-vv` options, for verbose and very verbose listings. To get full information on my (current) graphics card, I proceeded as shown in Listing 3. I now have an NVIDIA GeForce 740, and I'm using the nouveau kernel driver, among other internal details. Of course, to understand the produced information fully, you must have a bit of experience with PCI devices. Try the same command with `-vv`, and you'll see what I'm talking about.

If you are even more electronically/digitally minded, `lsdev` produces information about your installed hardware, including interrupts, ports, addresses and all such internal details. This command provides no options, and it's not likely you'll use it unless you are dealing very closely with hardware. Listing 4 shows an abbreviated example of the output. This command scans /proc/interrupts, /proc/ioports and /proc/dma, as described in the DIY with /proc

Listing 3. The `-v` option provides more detailed information; `-vv` goes even deeper.

```
# lspci -v -s 4:00.0
04:00.0 VGA compatible controller: NVIDIA Corporation
 ➥GK107 [GeForce GT 740] (rev a1)
 ➥(prog-if 00 [VGA controller])
    Subsystem: eVga.com. Corp. Device 2742
    Flags: bus master, fast devsel, latency 0, IRQ 27
    Memory at fd000000 (32-bit, non-prefetchable) [size=16M]
    Memory at e0000000 (64-bit, prefetchable) [size=256M]
    Memory at de000000 (64-bit, prefetchable) [size=32M]
    I/O ports at ec00 [size=128]
    [virtual] Expansion ROM at fe000000 [disabled] [size=512K]
    Capabilities: [60] Power Management version 3
    Capabilities: [68] MSI: Enable+ Count=1/1 Maskable- 64bit+
    Capabilities: [78] Express Endpoint, MSI 00
    Capabilities: [b4] Vendor Specific Information: Len=14
    Capabilities: [100] Virtual Channel
    Capabilities: [128] Power Budgeting
    Capabilities: [600] Vendor Specific Information: ID=0001
     ➥Rev=1 Len=024
    Capabilities: [900] #19
    Kernel driver in use: nouveau
    Kernel modules: nouveau
```

and /sys sidebar.

Finally, if you've made it this far, the `lshw` command is a sort of catch-all that can produce lots of information on all of your installed hardware. The `-short` option provides a (somewhat) abbreviated listing of everything in your box (see Listing 5, and note some interesting lines, "To Be Filled By O.E.M.", which show that someone was careless when setting up my motherboard). With this command, you get information on the system, buses, memory, processor, display, network and everything else.

Notice the "class" column in Listing 5. You can get a hint of the full information that `lshw` can provide by using the `-class` parameter to limit

Listing 4. The `lsdev` command provides information on interrupts, ports and direct memory access.

```
> lsdev
Device            DMA   IRQ  I/O Ports
------------------------------------------------
                          7
0000:00:1d.0                   c480-c49f
0000:00:1d.1                   c800-c81f
0000:00:1d.2                   c880-c89f
...
... (several lines snipped out)
...
eth0                      29
fpu                            00f0-00ff
gpio_ich                       0480-04bf 04b0-04bf
i801_smbus                19   0400-041f
i8042                   1 12
iTCO_wdt                       0830-0833 0830-0833 0860-087f 0860-087f
keyboard                       0060-0060 0064-0064
...
... (several lines snipped out)
...
timer                      0
timer0                         0040-0043
timer1                         0050-0053
uhci_hcd                       c480-c49f c800-c81f c880-c89f cc00-cc1f
uhci_hcd:usb2             23
uhci_hcd:usb3             19
uhci_hcd:usb4             18
uhci_hcd:usb5             16
vesafb                         03c0-03df
```

output. For example, see below the detailed specs on my network card; it shows the vendor, model and plenty of other details (warning: this is the kind of output you get if you don't restrict the command with `-short`; for my machine, `lshw` with no extra options produces a listing more than

Listing 5. The `lshw` command includes information on all your hardware."

```
# lshw -short
H/W path               Device      Class          Description
=============================================================
                                   system         To Be Filled
 ➥By O.E.M.
/0                                 bus            G41M-VS3.
/0/0                               memory         64KiB BIOS
/0/4                               processor      Core 2 Quad (To Be
 ➥Filled By O.E.M.)
/0/4/5                             memory         128KiB L1 cache
/0/4/6                             memory         4MiB L2 cache
/0/d                               memory         4GiB System Memory
/0/d/0                             memory         4GiB DIMM SDRAM
 ➥Synchronous
/0/d/1                             memory         DIMM [empty]
/0/100                             bridge         4 Series Chipset
 ➥DRAM Controller
/0/100/1                           bridge         4 Series Chipset
 ➥PCI Express Root Port
/0/100/1/0                         display        GK107 [GeForce
 ➥GT 740]
/0/100/1/0.1                       multimedia     GK107 HDMI Audio
 ➥Controller
/0/100/1b                          multimedia     NM10/ICH7 Family
 ➥High Definition Audio Controller
/0/100/1c                          bridge         NM10/ICH7 Family
 ➥PCI Express Port 1
/0/100/1c.1                        bridge         NM10/ICH7 Family
 ➥PCI Express Port 2
/0/100/1c.1/0          eth0        network        AR8152 v2.0 Fast
 ➥Ethernet
/0/100/1d                          bus            NM10/ICH7 Family
 ➥USB UHCI Controller #1
/0/100/1d/1            usb2        bus            UHCI Host Controller
/0/100/1d.1                        bus            NM10/ICH7 Family
 ➥USB UHCI Controller #2
/0/100/1d.1/1          usb3        bus            UHCI Host Controller
/0/100/1d.1/1/1                    bus            USB2.0 Hub
/0/100/1d.1/1/1/1                  input          OM
/0/100/1d.1/1/1/2                  input          USB Multimedia
 ➥Keyboard
/0/100/1d.1/1/1/3                  multimedia     USB2.0 Camera
/0/100/1d.1/1/2                    communication  Bluetooth Dongle
 ➥(HCI mode)

...several lines snipped out...

/0/1                   scsi2       storage
/0/1/0.0.0             /dev/sda    disk           500GB WDC
 ➥WD5000AAKS-0
/0/1/0.0.0/1           /dev/sda1   volume         4102MiB Linux
 ➥swap volume
/0/1/0.0.0/2           /dev/sda2   volume         461GiB EXT4 volume
/0/1/0.1.0             /dev/sdb    disk           160GB MAXTOR
 ➥STM316021
/0/1/0.1.0/1           /dev/sdb1   volume         4094MiB Linux
 ➥swap volume
/0/1/0.1.0/2           /dev/sdb2   volume         145GiB EXT3 volume
/0/2                   scsi3       storage
/0/2/0.0.0             /dev/sdc    disk           3TB WDC
 ➥WD30EZRX-00M
/0/2/0.0.0/1           /dev/sdc1   volume         2794GiB EXT4 volume
/0/2/0.1.0             /dev/cdrom  disk           DVD RW AD-7200S
```

500 lines long):

```
# lshw -class network
  *-network
    description: Ethernet interface
    product: AR8152 v2.0 Fast Ethernet
    vendor: Qualcomm Atheros
    physical id: 0
    bus info: pci@0000:01:00.0
    logical name: eth0
    version: c1
    serial: bc:5f:f4:12:e0:f1
    size: 100Mbit/s
    capacity: 100Mbit/s
    width: 64 bits
    clock: 33MHz
    capabilities: pm msi pciexpress vpd bus_master
➥cap_list ethernet physical tp 10bt 10bt-fd
➥100bt 100bt-fd autonegotiation
    configuration: autonegotiation=on broadcast=yes
➥driver=atl1c driverversion=1.0.1.1-NAPI
➥duplex=full latency=0 link=yes multicast=yes
➥port=twisted pair speed=100Mbit/s
    resources: irq:29 memory:fcfc0000-fcffffff
➥ioport:dc00(size=128)
```

The `lshw` command has several other interesting options. For example,

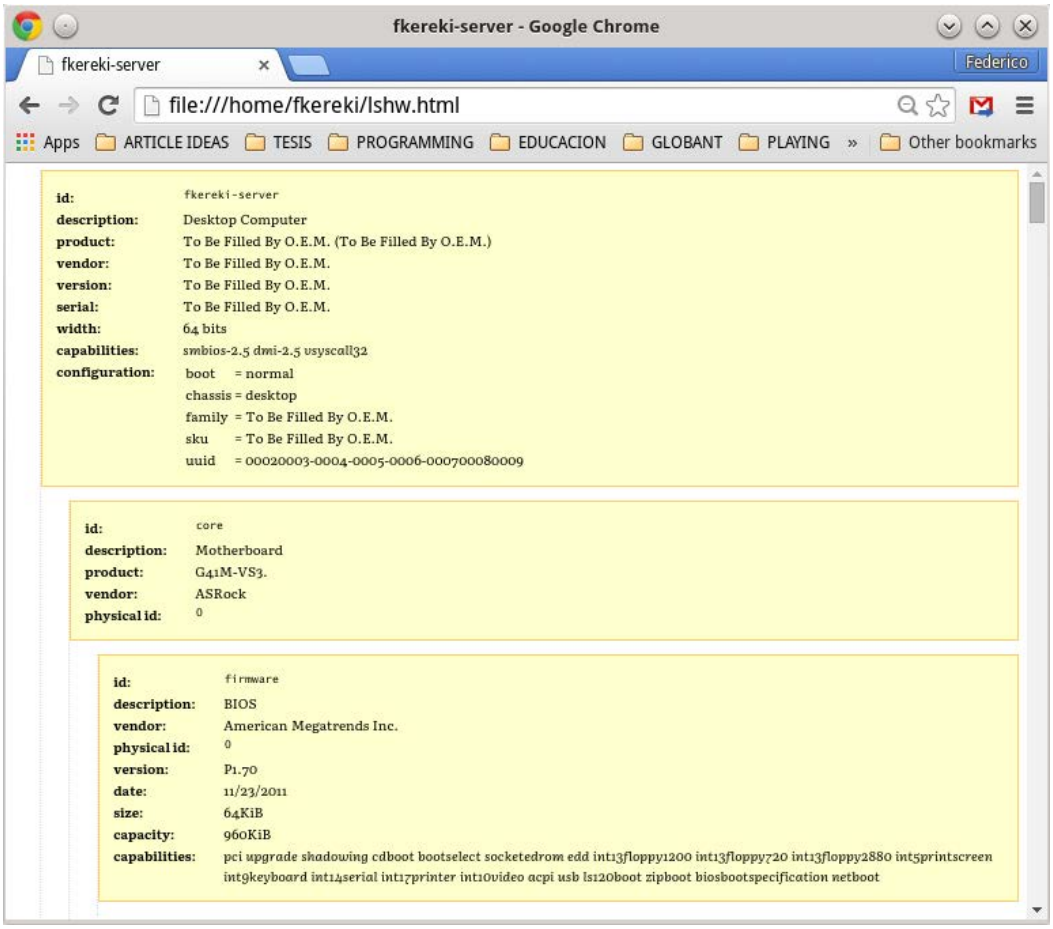


Figure 1. The `lshw` command also can produce HTML or XML output; the former is shown here.

it can produce either HTML or XML output (add the `-html` or `-xml` options); the former is appropriate for showing in a browser, while the latter is useful if you want to store or process your hardware information. See Figure 1 for just a small part of the full hardware description of my box. For security purposes, the `-sanitize` option removes sensitive information, such as serial numbers. There's even an `-X` option to use a graphical interface (I'll get to that later).

So far, I've discussed several ls* commands, and even if they are not actually a "family", they are my favorite tools. It's easy to remember them by typing `ls` and letting type-ahead suggest the rest. However, there are more command-line possibilities, so let's take a look.

More Command-Line Options

Let's start with some general commands. The first, `dmidecode`, allows you to dump the computer's DMI (or SMBIOS; see the What's SMBIOS? sidebar) in a more readable format. If the table is found, its contents are dumped record by record, similar to this:

```
# dmidecode -t 6
# dmidecode 2.12
SMBIOS 2.5 present.

Handle 0x0009, DMI type 6, 12 bytes
Memory Module Information
    Socket Designation: DIMM0
    Bank Connections: 0 1
    Current Speed: Unknown
    Type: DIMM SDRAM
    Installed Size: 4096 MB (Double-bank Connection)
    Enabled Size: 4096 MB (Double-bank Connection)
    Error Status: OK
```

What's SMBIOS?

How does Linux recognize what devices are installed? Since 1995, the SMBIOS (System Management BIOS) specification has provided this kind of information, doing away with the need for potentially worrisome operations like hardware probing. This standard (used by DMI) is geared to the Intel 32- and 64-bit processor architecture systems. Basically, it defines a structure with appropriate data for each kind of device, such as CPU, RAM, system slots and more. On principle, you could parse and decode this table by yourself, but several of the commands shown in this article already do that job. If you are curious about the specifics of the standard, see the Resources section.

Table 2. SMBIOS has several record types that you can select with `dmidecode`.

TYPE	DESCRIPTION
0	BIOS
1	System
2	Baseboard
3	Chassis
4	Processor
5	Memory Controller
6	Memory Module
7	Cache
8	Port Connector
9	System Slots
10	On-board Devices
11	OEM Strings
12	System Configuration Options
13	BIOS Language
14	Group Associations
15	System Event Log
16	Physical Memory Array
17	Memory Device
18	32-bit Memory Error
19	Memory Array Mapped Address
20	Memory Device Mapped Address
21	Built-in Pointing Device
22	Portable Battery
23	System Reset
24	Hardware Security
25	System Power Controls
26	Voltage Probe
27	Cooling Device
28	Temperature Probe
29	Electrical Current Probe
30	Out-of-band Remote Access
31	Boot Integrity Services
32	System Boot
33	64-bit Memory Error
34	Management Device
35	Management Device Component
36	Management Device Threshold Data
37	Memory Channel
38	IPMI Device
39	Power Supply
40	Additional Information
41	Onboard Devices Extended Information
42	Management Controller Host Interface
126	Disabled Entry
127	"End-of-Table" Special Marker
128–255	OEM-specific Data

If I were to give an award for "Most Talkative Command", surely it would go to `hwinfo`, another command that can dump all the hardware information on your computer.

```
Handle 0x000A, DMI type 6, 12 bytes
Memory Module Information
   Socket Designation: DIMM1
   Bank Connections: 4 5
   Current Speed: Unknown
   Type: DIMM SDRAM
   Installed Size: Not Installed
   Enabled Size: Not Installed
   Error Status: OK
```

If you don't want to list the entire table (several hundred lines in my computer), you can restrict the output to a specific type of entry, according to SMBIOS definitions (Table 2).

You also can use specific keywords to restrict the output to a few types (Table 3).

If I were to give an award for "Most Talkative Command", surely

it would go to `hwinfo`, another command that can dump all the hardware information on your computer. On my machine, running `hwinfo` without any parameters produces more than 12,000 lines, including several memory dumps of the SMBIOS table. You can produce a much more compact version with the `--short` option (Listing 6).

Table 3. You also can use special keywords to get related information from SMBIOS.

SMBIOS Keyword	SMBIOS Types
bios	0,13
system	1,12,15,23,32
baseboard	2,10,41
chassis	3
processor	4
memory	5,6,16,17
cache	7
connector	8
slot	9

Listing 6. The `hwinfo` command can be quite talkative; using the `--short` option makes it more manageable.

```
# hwinfo --short
cpu:
          Intel(R) Core(TM)2 Quad CPU Q8400 @ 2.66GHz, 2670 MHz
          Intel(R) Core(TM)2 Quad CPU Q8400 @ 2.66GHz, 2336 MHz
          Intel(R) Core(TM)2 Quad CPU Q8400 @ 2.66GHz, 2670 MHz
          Intel(R) Core(TM)2 Quad CPU Q8400 @ 2.66GHz, 2670 MHz
keyboard:
          Logitech USB Multimedia Keyboard
mouse:
          Elan Microelectronics OM
monitor:
          SAMSUNG SA300/SA350
          SAMSUNG S20B300
graphics card:
          nVidia VGA compatible controller
sound:
          Intel NM10/ICH7 Family High Definition Audio Controller
          nVidia GK107 HDMI Audio Controller
storage:
          Intel 82801G (ICH7 Family) IDE Controller
          Intel NM10/ICH7 Family SATA Controller [IDE mode]
network:
  eth0    Atheros AR8152 v2.0 Fast Ethernet
network interface:
  lo      Loopback network interface
  eth0    Ethernet network interface
disk:
  /dev/sda   WDC WD5000AAKS-0
  /dev/sdb   MAXTOR STM316021
  /dev/sdc   WDC WD30EZRX-00M

...etc (rest of the listing, snipped out)
```

You can restrict `hwinfo` to a specific type of hardware by adding an option, such as `--monitor` or `--printer`. Get the whole list of options with `hwinfo --help`. For instance, I can dump the optical unit data with `hwinfo --cdrom` (Listing 7). The `--listmd` option lets you include RAID devices, which usually aren't included in the standard output.

Of the command-line programs I'm covering in this article, `inxi` is more colorful, even if only moderately (Figure 2).

If invoked with no parameters, it will just produce a line like the

Listing 7. The `hwinfo` command can restrict its output to specific hardware, as the cdrom device, for example.

```
# hwinfo --cdrom
25: SCSI 301.0: 10602 CD-ROM (DVD)
  [Created at block.249]
  Unique ID: KD9E.SGHalmfn+h9
  Parent ID: w7Y8.xyd+qedQTr5
  SysFS ID: /class/block/sr0
  SysFS BusID: 3:0:1:0
  SysFS Device Link: /devices/pci0000:00/0000:00:1f.2/
➥ata4/host3/target3:0:1/3:0:1:0
  Hardware Class: cdrom
  Model: "SONY DVD RW AD-7200S"
  Vendor: "SONY"
  Device: "DVD RW AD-7200S"
  Revision: "1.61"
  Driver: "ata_piix", "sr"
  Driver Modules: "ata_piix", "sr_mod"
  Device File: /dev/sr0 (/dev/sg3)
  Device Files: /dev/sr0, /dev/cdrom, /dev/cdrw,
 ➥/dev/disk/by-id/ata-Optiarc_DVD_RW_AD-7200S,
 ➥/dev/disk/by-path/pci-0000:00:1f.2-ata-2.1,
 ➥/dev/dvd, /dev/dvdrw
  Device Number: block 11:0 (char 21:3)
  Features: CD-R, CD-RW, DVD, DVD-R, DVD-RW, DVD-R DL,
 ➥DVD+R, DVD+RW, DVD+R DL, DVD-RAM, MRW, MRW-W
  Drive status: no medium
  Config Status: cfg=no, avail=yes, need=no, active=unknown
  Attached to: #14 (IDE interface)
  Drive Speed: 48
```

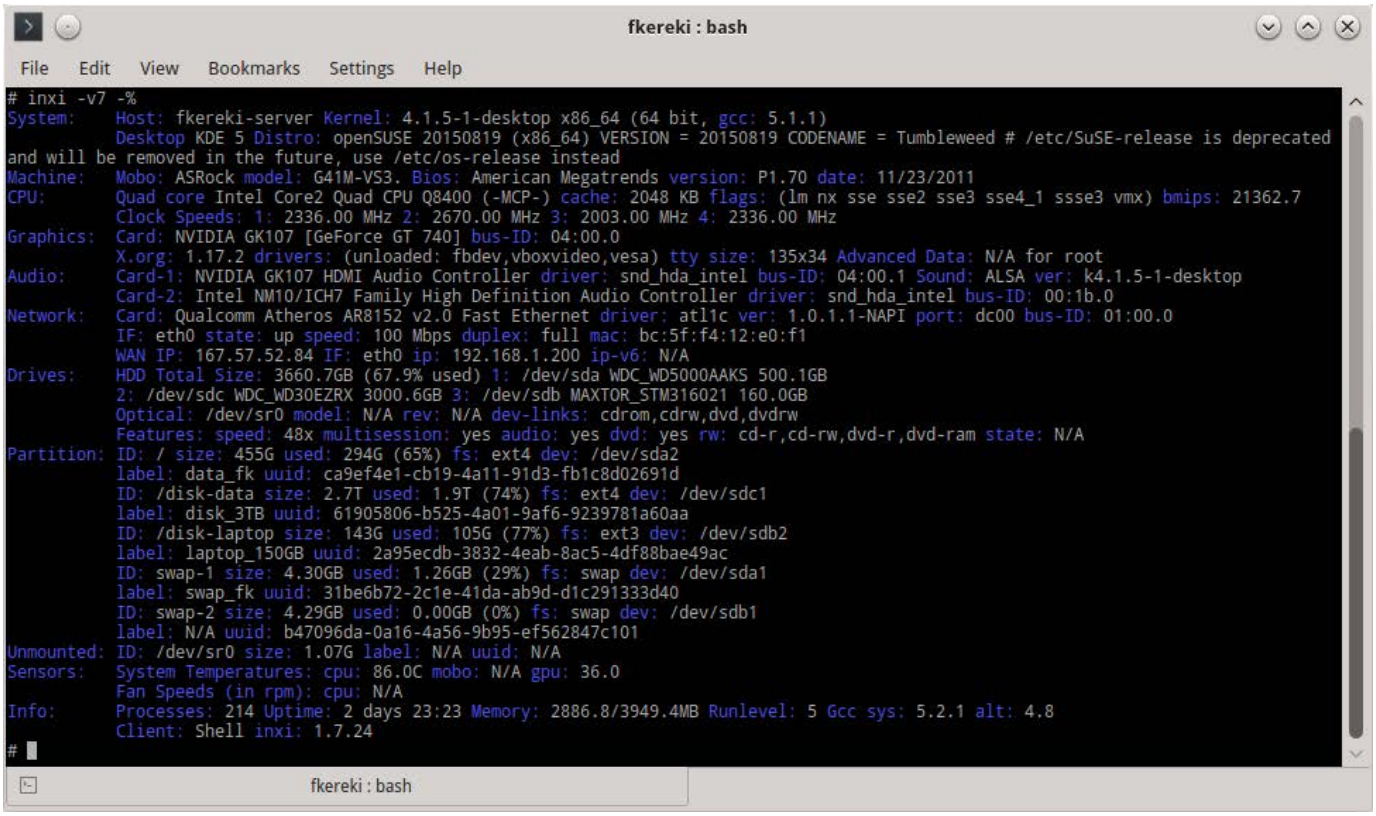


Figure 2. inxi, even if only a command-line tool, at least tries to use some colors.

following, showing CPU, kernel, uptime and a few more details:

```
CPU~Quad core Intel Core2 Quad CPU Q8400 (-MCP-)
 ➥clocked at 2003.000 Mhz Kernel~4.1.5-1-desktop
 ➥x86_64 Up~2 days 23:24 Mem~2377.4/3949.4MB
 ➥HDD~3660.7GB(67.9% used) Procs~202 Client~Shell
 ➥inxi~1.7.24
```

However, you can use lots of options to get specific data. For example, you can set the verbosity level with options `-v0` (minimum) through `-v7` (maximum verbosity). The `-x` option allows including extra information for some hardware. Check out `inxi -h` to get all possible options. For instance, you can get audio information with `inxi -A` or graphics card data with `inxi -G` and so on:

```
# inxi -A
Audio:     Card-1: NVIDIA GK107 HDMI Audio Controller
 ➥driver: snd_hda_intel Sound: ALSA ver: k4.1.5-1-desktop
           Card-2: Intel NM10/ICH7 Family High
 ➥Definition Audio Controller driver: snd_hda_intel
```

Now, let's finish with some GUI options.

The GUI Way

To start with, `usbview` is a rough graphic equivalent of `lsusb` or `usb-devices`, which I discussed earlier. It's quite simple to use, with no options or parameters. It shows two columns: the left one is a tree of all available USB devices, and the right one gives the full details. Figure 3 shows details on my USB keyboard.

Let's move on to a command I already discussed, which shares the display style: `lshw -X`. Instead of producing a listing (as shown previously), the `-X` option produces a graphic interface with several columns on the left to let you choose what hardware to inspect. An area to the right shows the full hardware details for the chosen device. Figure 4 shows the result of analyzing my optical DVD reader/writer unit; the provided information includes other details, such as the logical unit name, its

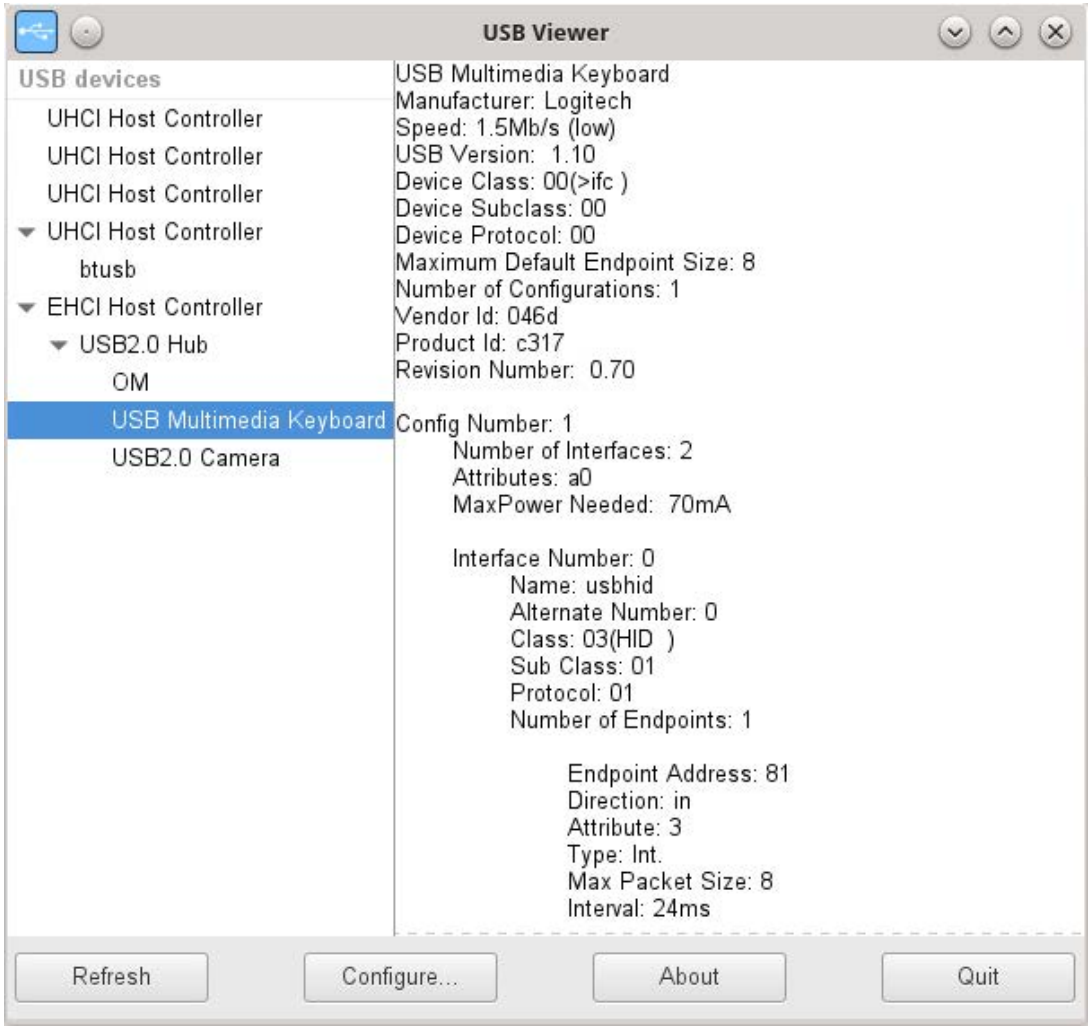


Figure 3. The `usbview` command shows the details of all USB devices in tree form.

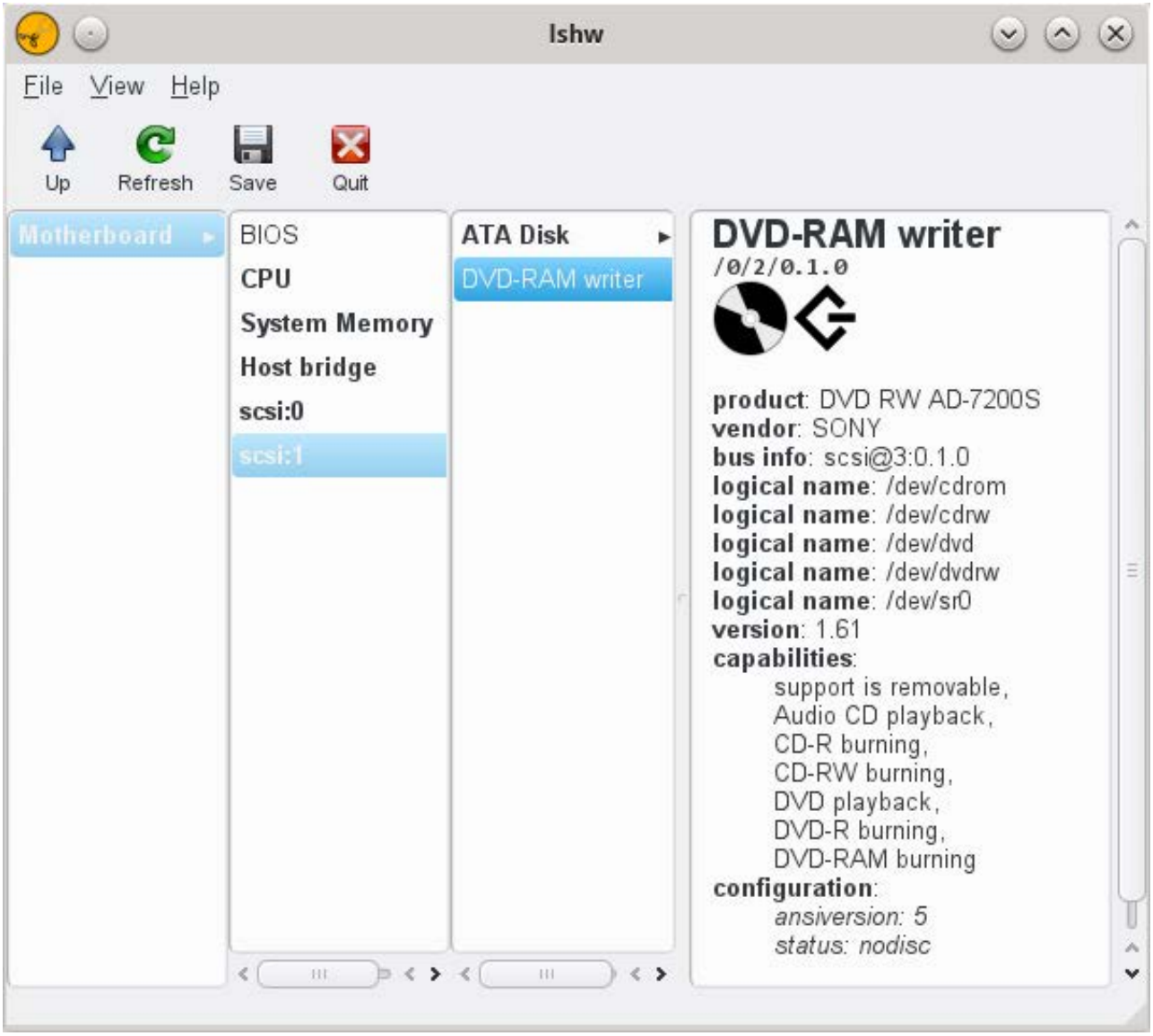


Figure 4. The `lswh -X` command produces a graphic interface that lets you browse all hardware devices.

capabilities and more.

Another interesting program is `hardinfo`, which "is not dead, but needs a maintainer", according to its GitHub page (see Resources.) This program shows a tree structure to the left with four main branches:

- Computer shows lots of details about your machine: some are related to software and not to hardware.
- Devices includes all devices in your box, grouped by category.

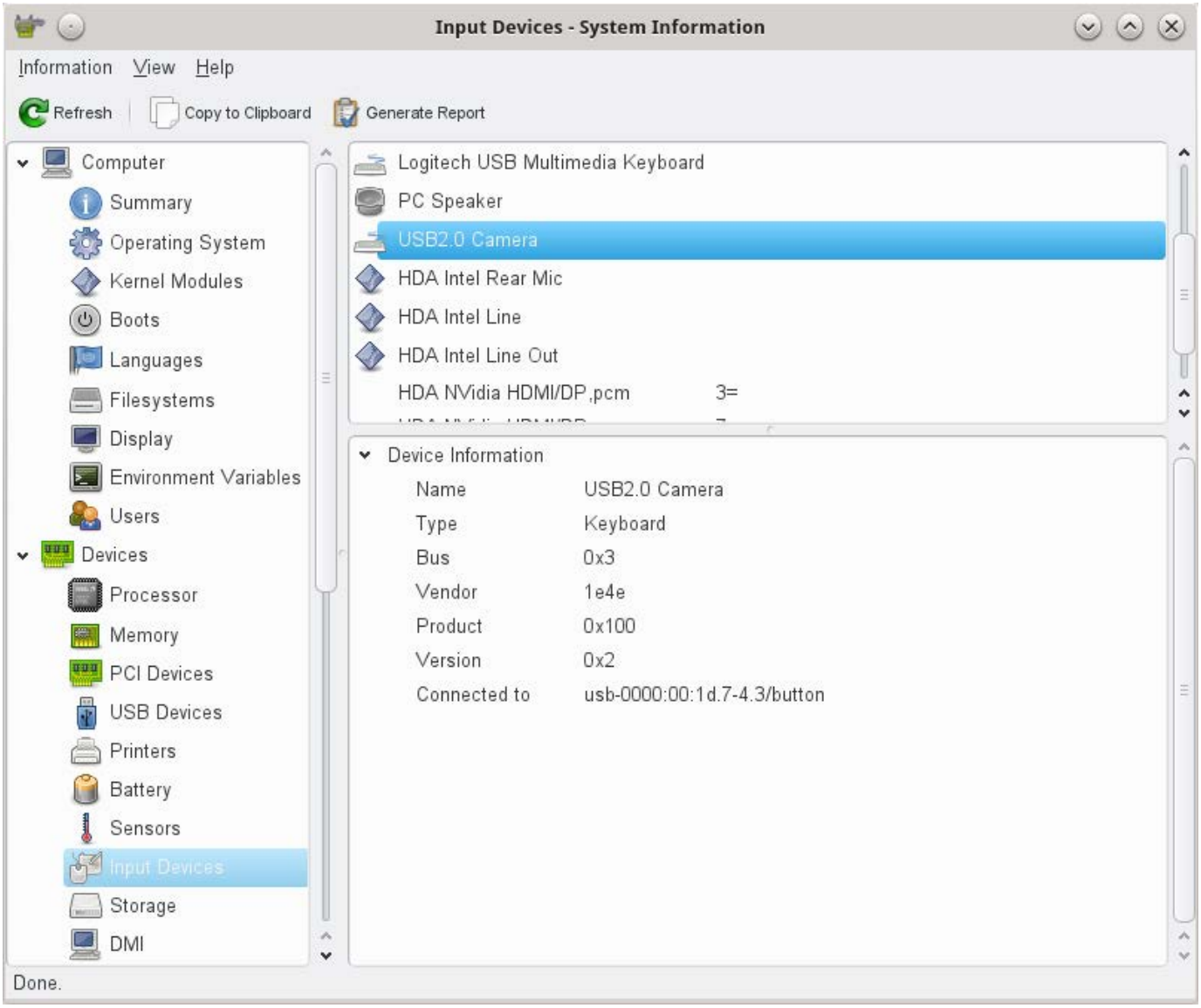


Figure 5. The `hardinfo` command includes several extra pieces of data, not limited only to hardware.

- Network not only shows network card details, but also some other aspects, such as DNS servers or routing.
- Benchmarks lets you see how your machine fares against other computers, but because of the lack of updates, the comparisons are against old CPUs.

Figure 5 shows sample output. There are two more options. The "Information" menu entry allows

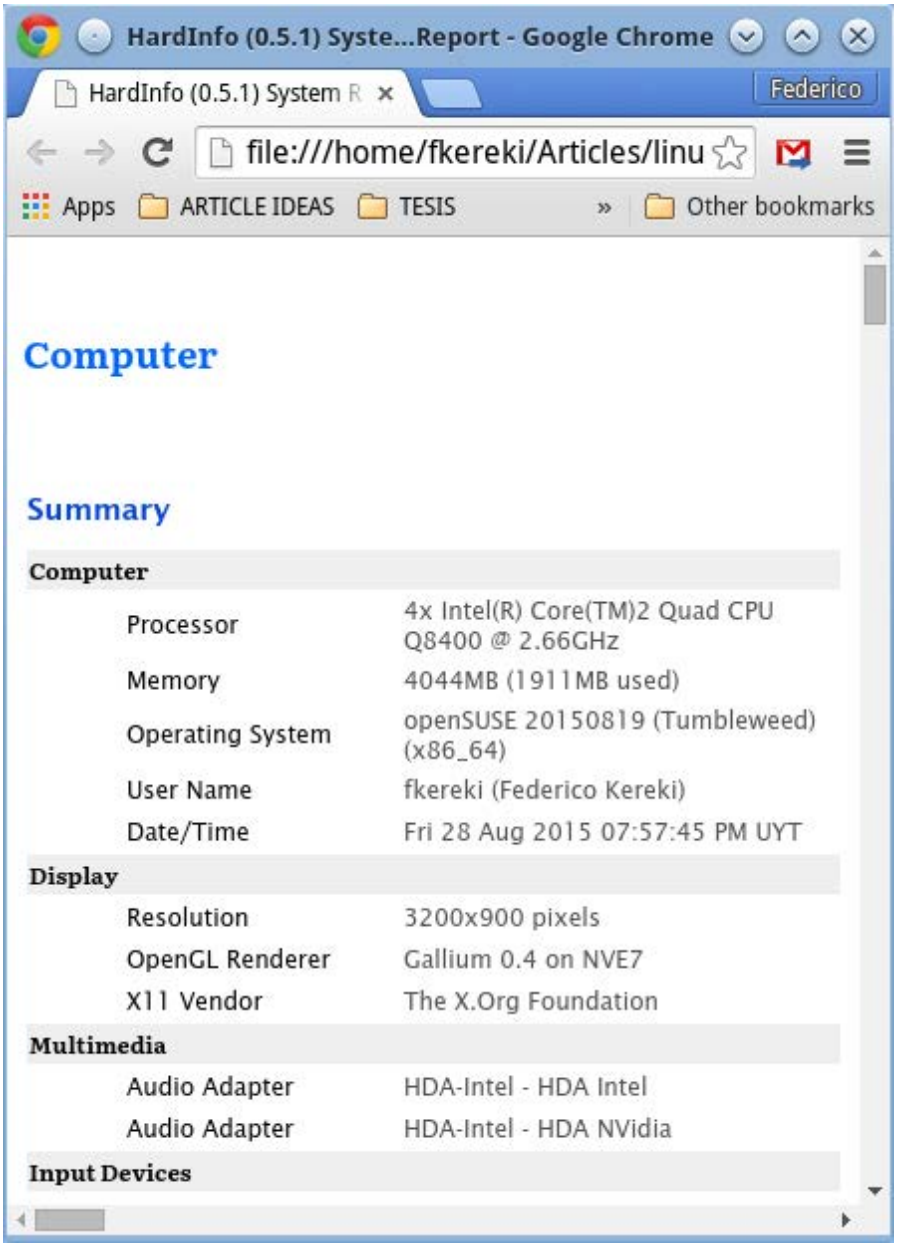


Figure 6. The `hardinfo` command can produce an HTML or text report describing your complete system.

you to produce a report, in either HTML or plain-text format, choosing whichever parts interest you. The "Network Updater" should let you update the internal program data, including more recent benchmark results, but when I tried to run it, I got a "Contacting HardInfo Central Database (failed)" message. See Figure 6 for a example of the produced HTML report.

Let's end with KDE's own `kinfocenter`. This utility (see Figure 7, which shows RAM details for my machine) is similar to the previous tools I've been describing, and it

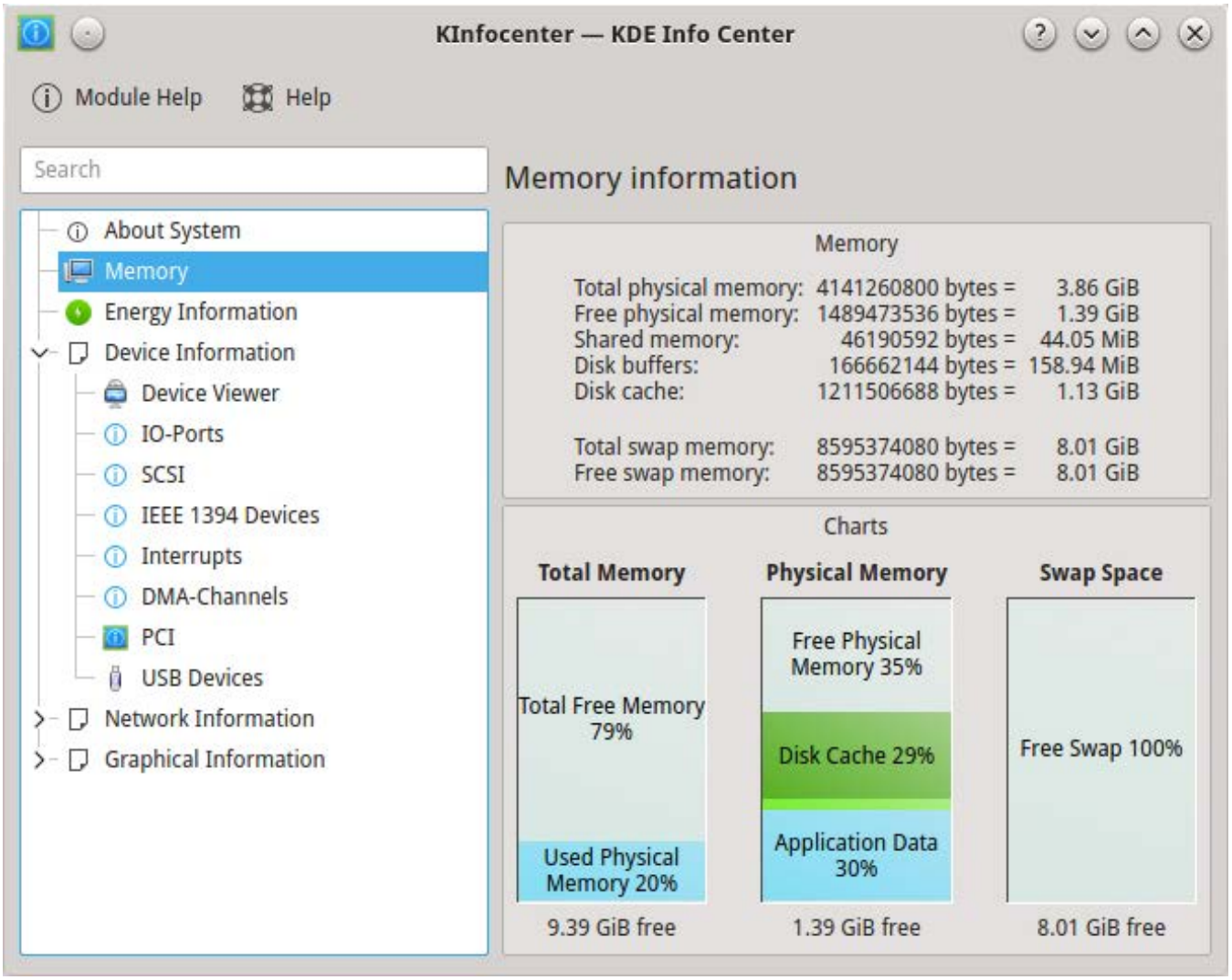


Figure 7. KDE's own kinfocenter shows not only hardware details, but plenty of other system data as well.

offers a left pane with a tree with all available options and a right pane with more details on the chosen option at the left.

The program doesn't restrict itself to hardware details, but shows all kinds of other information, such as "Samba Status", "Energy Information" or "X-Server", just to mention a few.

Conclusion

I've covered a lot of commands that let you query your Linux machine and learn, in more or less detail, what's exactly in it. And if you need to, you even can get at the base data by yourself and whip up your own hardware-inspection tool.■

DIY with /proc and /sys

Linux is full of directories and files, but the /proc and /sys directories are really strange. They don't actually exist, but they allow you to browse them. They are full of zero-length empty files, but you can open and view them. The /proc directory preceded /sys, and it has basically all details about running processes (hence, the /proc name). Over time, more files were added to it, mostly "virtual" ones, which don't actually exist, but are created on the fly if you try to access them. (Most virtual files sport a current timestamp, which shows that they constantly are kept up to date and their contents are the latest possible.) The /sys directory is more modern. It appeared around the time of the 2.6 kernel to introduce more order and a better structure than provided by the older /proc, which had just grown in a sort of haphazard way. Many of the files (but not all) in /proc are duplicated in /sys, and whenever possible, you should pick the files in the latter directory. The /sys directory has several subdirectories:

- block/ has an entry pointing to each block device.
- bus/ has directories for each bus type, and within each, two subdirectories: devices/ and drivers/. The former has a directory for each device, pointing to the device's directory in /root, and the latter has a directory for each driver that was loaded for devices on the given bus.
- class/ has directories for each type of object; some examples are block/, graphics/, net/, sound/ and so on.
- dev/ provides directories for each type of device (for example, dev/block/ or dev/char/), each with directories for each appropriate device.
- devices/ contains the global device hierarchy, with every physical device in your system.
- firmware/ includes directories for firmware-specific objects; for example, acpi/ or memmap/, but the particular directories in your own machine depend on the firmware drivers in your kernel.
- fs/ has a directory for each filesystem type in your machine, each with further directories for each specific device; for example, I have /sys/fs/ext4/sda2, because the disk mounted as /dev/sda2 uses ext4.
- kernel/ has several files related to the currently loaded kernel.
- module/ has a subdirectory for each and every module loaded into the kernel.
- power/ represents the power subsystem.

When you get to the deepest levels of any branch, you may find any number of individual files, which you can read to get attributes of the given object. What files? That's a hard question to answer, because it depends on which specific branch you are visiting, so you'll have to do a bit of work before you get to extract information from the /sys directory. (See Resources for some pointers about this.) Also, be aware that you can write to some of the files, and that will imply modifying the corresponding parameter—be warned: do this with care! However, if you keep at it, you'll be able to duplicate the functionality of most of the tools shown in this article, which often work the same way.

Resources

Read about the SMBIOS standard at http://www.dmtf.org/standards/smbios. At the time of this writing, the latest version is 3.0.0, dated 2/15/2015.

You can find information on `sysfs` at https://www.kernel.org/doc/Documentation/filesystems/sysfs.txt and more specific documentation at https://www.kernel.org/doc/Documentation/ABI/stable.

Regarding the older `procfs`, check https://www.kernel.org/doc/Documentation/filesystems/proc.txt.

The USB ID repository at http://www.linux-usb.org/usb-ids.html has the full list of all known IDs used in USB devices.

The PCI ID repository at http://www.pcidatabase.com provides a centralized list of PCI device IDs.

The `lscpu` and `lsblk` commands are part of the util-linux package, available at https://www.kernel.org/pub/linux/utils/util-linux. For documentation, check out http://linux.die.net/man/1/lscpu and http://linux.die.net/man/8/lsblk, respectively.

Read about `lsscsi` options at http://sg.danny.cz/scsi/lsscsi.html and find a manual page at http://linux.die.net/man/8/lsscsi.

For the `lsdev` man page, see http://linux.die.net/man/8/lsdev.

The `lshw` home page is at http://www.ezix.org/project/wiki/HardwareLiSter, and its manual page is at http://linux.die.net/man/1/lshw.

See `lsusb` in the "usbutils" page at https://github.com/gregkh/usbutils, and get more information at http://linux.die.net/man/8/lsusb.

You can find `lspci` at http://mj.ucw.cz/sw/pciutils (home of the "PCI Utilities") and the man page at http://linux.die.net/man/8/lspci.

Check out `usbview` at http://www.kroah.com/linux/usb and its man page at http://linux.die.net/man/8/usbview.

The `hardinfo` source repository is at https://github.com/lpereira/hardinfo, but first check your distribution's repositories; it's likely to already be there. Note that the program's last update was more than two years ago, and no further maintenance has been done.

You can find KInfoCenter at https://www.kde.org/applications/system/kinfocenter.

Federico Kereki is a Uruguayan systems engineer with more than 25 years of experience doing consulting work, developing systems and teaching at universities. He is currently working as a UI Architect at Globant, using a good mixture of development frameworks, programming tools and operating systems—and FLOSS, whenever possible! He has written several articles on security, software development and other subjects for *Linux Journal*, IBM developerWorks and other Web sites and publications. He also wrote the *Essential GWT* book. You can reach Federico at fkereki@gmail.com.

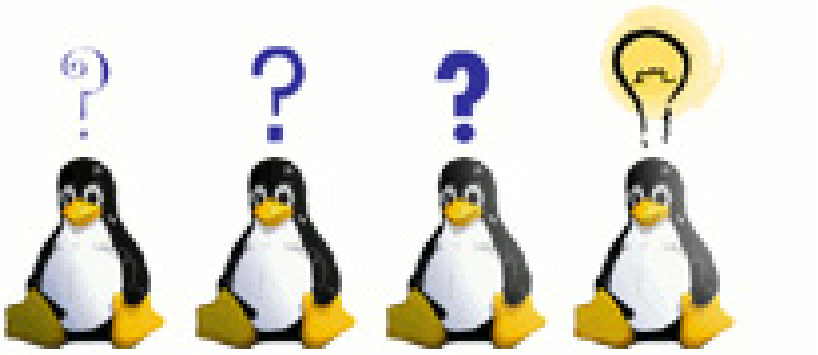
Send comments or feedback via http://www.linuxjournal.com/contact or to ljeditor@linuxjournal.com.

2015 *Linux Journal* Archive

2015 Archive Coming in December!

LinuxQuestions.org

Not Your Average Linux Forum



Boasting more than a half-million users worldwide, welcome to the Linux forum that really feels like a community: LinuxQuestions.org.

BRIAN CONNER

For many of us, our introduction to computing is being placed in front of a machine where the only challenge is figuring out the Windows user experience paradigm. Getting started with Linux, on the other hand, requires a bit more effort, a fair amount of trial and error, and perhaps some colorful language along the way.

When I got started with Linux (Red Hat 9 specifically) back in 2003, the process was quite involved. Before beginning the installation process, I did significant research and took copious notes (by hand, as I recall) on hard drive repartitioning and configuration of master boot records. The sense of accomplishment that resulted the first time Red Hat booted successfully faded quickly as I realized that much still needed to be done: getting X working on my hardware, configuring audio drivers, getting dial-up networking to work and so on. For each issue, the process was the same. Working under Windows, use the Internet to research the process (and take notes) and download the necessary packages/patches. I then would reboot into Red Hat and attempt what I had researched. When I failed—and I did, many times—I would take careful notes about what I did and what the logs and error messages said. Then I would reboot into Windows and scour the Internet support forums for people having the same problem and repeat the process.

Thanks to the near-ubiquity of the Internet, this process now is much more streamlined. The plethora of really good and free virtualization platforms have eliminated the need to "risk" your primary computer to try something new and different. The willingness of hardware manufacturers to work with kernel developers has led to the major subsystems (audio, video, Wi-Fi and so on) just working in most cases.

The thing that hasn't changed, and probably never will, is the need to look to the community for support and guidance. Even the most comprehensive documentation can't possibly cover every scenario, and so it's through the sharing of our experiences (successes and failures) that we all learn.

Today, if you do a Google search for an error or issue installing or configuring Linux, the results you get will be full of support forum posts from other users who had the same problem and (hopefully) a solution. At least a couple of those links will point to posts at the various forums at LinuxQuestions.org. With more than a half-million registered users around the world and two-dozen forums covering a wide range of topics, LQ is one of the largest and most active user support sites around.

Earlier this year, LinuxQuestions.org celebrated its 15th anniversary, and to commemorate this milestone, I spent some time with its founder and maintainer, Jeremy Garcia.

Brian Conner: Tell me about yourself—where you're from, what you do for a living, hobbies and so on.

Jeremy Garcia: I'm from Buffalo, New York, and in addition to running The Questions Network (currently LinuxQuestions.org, AndroidQuestions.org and ChromeOSQuestions.org), I do consulting around both implementing open source and building sustainable communities. Outside the tech realm, I enjoy running, traveling, local history and am a bit of a foodie. I'm also a Bills and Sabres fan, which has proven challenging during the past decade.

BC: How did you get your start with computers and Linux (the obligatory "what was your first distro" question)? And, what is your current distro?

JG: Computers, programming and technology in general always have fascinated me. As for Linux specifically, while I was in high school, I started working for a local ISP that used UNIX almost exclusively. The "UNIX way" just clicked and made a lot of sense to me. It wasn't long before I wanted to run something similar at home. The ISP used SCO (fairly ironic in retrospect), so home use really wasn't an option for licensing and cost reasons. Searching for an alternative quickly led me to Linux. I purchased *The Linux Bible* from a local bookstore, so my first distro was Yggdrasil. I've used Linux as my main OS ever since. I like to tinker and understand how things work, so the fact that I could get an operating system that allowed me not only to *see* how things worked, but also to *modify* how things worked, enthralled me. I moved to Slackware about a year later and have used Debian, Red Hat, SUSE, Fedora, Mandrake, Conectiva and a few others as my main distro through the years. I currently use Ubuntu on my desktop, but I have been considering alternatives.

BC: For those who don't know, what is LinuxQuestions.org?

JG: LQ is an on-line Linux community. The forum is the most well known aspect, but we have a wiki, tutorials, news and more. More than 30 distributions officially participate, so it has a little different vibe from some of the single distro fora. We focus on being friendly and welcoming to Linux newbies and veterans alike.

BC: What prompted you to start LQ?

JG: I had just gotten my first real Linux job, and I wanted to give something back to the community. I had been using Linux for a while at that point and wanted to offer help to existing and potential users. I figured someday the site would grow to maybe a few hundred people, so to say it has grown far beyond my initial expectations is a monumental understatement.

exhibit at LinuxWorld in New York a few years in, and the feedback we got was really energizing. We had mods fly in from multiple countries, and people from all over the world visited our booth to tell us how much they liked the site, explain how much it had benefited them or just stopped by to say hello because they wanted to meet us. It was a humbling experience, and one that

What I can say is that LQ will do everything it can to remain relevant while staying true to our ethos. We've built up a huge knowledge base through the years, and keeping that available to the community for posterity is important to me.

BC: Earlier this year, LQ celebrated its 15th anniversary. Surely you didn't expect it would become what it has. Do you remember what your expectations were at launch time?

JG: As I mentioned, when I started the site, my initial expectations were fairly modest. I really just wanted to give something back to a community that I felt had given me quite a bit. I put a ton of work into the site, but I did so because I enjoyed it, not because I expected any long-term gains. We got a chance to

has happened many times since. If you'd have told me when I founded the site that I'd have experiences such as that one, I'd certainly not have believed you.

BC: Relating to the last question, would you care to hazard a guess where LQ will be 15 years from now?

JG: Projecting that far out is extraordinarily difficult, especially when the Internet is involved. What I can say is that LQ will do everything it can to remain relevant

while staying true to our ethos. We've built up a huge knowledge base through the years, and keeping that available to the community for posterity is important to me.

BC: Tell us about the hardware and software platform that LQ runs on currently and how it has evolved during the last 15 years.

JG: LQ started on a single dedicated Pentium Pro server co-located at the aforementioned ISP. It's had a couple iterations since, and we're actually working on an infrastructure refresh now. The site currently runs on bare-metal RHEL servers and uses nginx and MariaDB.

BC: In your estimation, what's the biggest achievement (or accomplishment) of LQ so far?

JG: That's a difficult question. I'm proud that we've remained friendly and true to our initial goals despite how much we've grown—that's a testament to our great mod team. I'm also proud of how many people we've helped through the years. I've heard from many members that they would have given up on Linux if it weren't for LQ. That's a testament to our great members.

BC: Surely there have been some low points along the way. Any particular occurrences that you were able to learn from as you moved forward?

JG: We've been lucky in this regard I think. When a fellow Linux forum went under, some in the LQ community felt the sudden influx of new members would impact our culture, but I think it made us stronger. We did have some DB corruption about ten years ago that looked grim, but we were able to recover with minimal data loss.

BC: Has the success of LQ opened any doors that wouldn't have been available to you otherwise (speaking at cons, meeting interesting people)?

JG: Absolutely. I've gotten to meet Linus and a variety of other open-source luminaries. I've gotten to speak at conferences, judge the LinuxWorld awards, be on a variety of panels, be on the board of Linux Fund, have my own magazine column—too much to list really. The Open Source world is full of smart, energetic, talented people. I'm absolutely a better person for having been exposed to it. I've also made quite a few good friends along the way.

Jeremy Garcia with Linus Torvalds at LinuxWorld in 2003

BC: Meeting Linus so early in the life of LQ, must have been a huge thrill. Did you get a chance to talk with him? Does he have an account at LQ (and does he use it)?

JG: Linus stopped by many of the booths in the .org pavilion, and we got to chat with him for a while, so it was definitely an experience. He mentioned that he was familiar with the site and had ended up there a few times after searching for solutions to non-kernel problems, although I don't think he has an account.

BC: With more than 500,000 registered users, LQ has to be one of the largest non-distro-specific Linux user support forums. Why do you think LQ has become so popular? Along those lines, the LQ community is one of the most positive and supportive I've experienced. Any thoughts on how you've managed to avoid the trolling and newb-bashing

that seems to happen on other support forums?

JG: I think our success is very much due to the fact that we *are* friendly, positive and supporting. I explicitly set that welcoming tone very early on, as I knew the Linux community had a reputation among some for being rough on new users. The mods very much picked up on that and have kept that ethos alive during our growth. With that culture now ingrained, new members pick up on it and reciprocate.

is limited to posts from the last 30 days. Was this time frame chosen for practical reasons or philosophical ones?

JG: I absolutely concur with your assessment; if you're an experienced Linux user and are looking for a challenge, that link is a great place to start. Answering "Zero Reply Threads" is something we actively promote, and we'll occasionally have dedicated Zero Reply Drives, which do measurably impact the number of threads that haven't received a response. Returning results from the last month seemed like a reasonable default, but you can arbitrarily change the duration.

Answering "Zero Reply Threads" is something we actively promote, and we'll occasionally have dedicated Zero Reply Drives, which do measurably impact the number of threads that haven't received a response.

BC: One of my favorite things about LQ, which I suspect often is overlooked, is the direct link to "Zero Reply Threads" in the right-side menu. I feel this is a great way for an experienced Linux user to jump in and help those users who are, perhaps, most in need. Do you keep an eye on the traffic to this page? And, has it had the desired effect? Also, I noted that the listing

BC: The landscape of Linux is in a perpetual state of change—new and changing distros, applications being forked, applications being abandoned—how is this constant

Jeremy Garcia on a Panel at SCALE 13x

flux reflected in the LQ community, if at all? And, does it provide any challenges to you and the maintainers of LQ?

JG: I don't think it presents us with any specific challenges. I think the ever-changing landscape of the Open Source world is one thing that keeps it fresh and interesting.

BC: Could you pull back the curtain a little bit on the annual LQ Members Choice Awards? Linux users as a group tend to be very loyal to their distro and preferred apps, so I imagine the voting and comments can get heated.

JG: With our large and varied member base, the MCA discussions certainly can get heated—especially when it comes to certain categories, such as desktop distro, text editor and a couple others. I started the annual polls on a bit of a whim, and it's been really interesting to watch them grow each year. It's also been rewarding to see how enthusiastic some of the past winners have been.

BC: In addition to LQ, you are one

of the principles of the *Bad Voltage* podcast. Please tell us about *Bad Voltage* and how it came to be.

JG: *Bad Voltage* is a podcast that myself, Jono Bacon, Bryan Lunduke and Stuart Langridge have been doing for almost two years. Every two weeks, we deliver an amusing (sometimes NSFW) take on technology, Linux, open source, politics, music and anything else we think is interesting. I've been friends with Jono for many years after seeing him at one too many conferences and was introduced to Stuart through him. We tossed the idea around for a bit after a suggestion from Jono and agreed that Bryan complimented the team nicely. We recorded a pilot episode to see if the idea was worth exploring further, and all of us knew it had some potential right then. I think we've really hit our stride, and we recently performed our first live show at SCALE in Los Angeles. We have another live show coming up in Fulda, Germany. Check out http://badvoltage.org for more.

BC: What's the best way for an experienced Linux user to get involved with LQ community? And, what is your advice to someone just getting started with Linux?

JG: We're always looking for experienced Linux users to help answer questions. There is a continual stream of new and interesting challenges to solve in the forum, so if you want to help the Linux community, I think LQ is a great place to do so. As for those wanting to get started with Linux, I'd say jump right in. Linux distributions have never been more approachable, and if you get stuck, you can always head over to LQ for help.■

An Internet junkie working as a Web developer and all-around "IT guy" for a small nonprofit in central Maryland, Brian likes to relax with a craft beer while poking and prodding at his favorite distro: Slackware. When not in front of a monitor, Brian's time is spent with his two beautiful daughters (and his beautiful wife), reading or enjoying college football.

Resources

LinuxQuestions.org:
http://www.linuxquestions.org

Jeremy's Blog:
http://jeremy.linuxquestions.org

Zero Reply Threads:
http://www.linuxquestions.org/questions/lqsearch.php?do=noreplies

Bad Voltage Podcast:
http://www.badvoltage.org

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

LINUX JOURNAL

on your

e-Reader

e-Reader editions FREE for Subscribers

Customized Kindle and Nook editions now available

LEARN MORE

WEBCASTS

Maximizing NoSQL Clusters for Large Data Sets

Sponsor: IBM

This follow-on webcast to Reuven M. Lerner's well-received and widely acclaimed Geek Guide, "Take Control of Growing Redis NoSQL Server Clusters", will extend the discussion and get into the nuts and bolts of optimally maximizing your NoSQL clusters working with large data sets. Reuven's deep knowledge of development and NoSQL clusters will combine with Brad Brech's intimate understanding of the intricacies of IBM's Power Systems and large data sets in a free-wheeling discussion that will answer all your questions on this complex subject.

> **http://geekguide.linuxjournal.com/content/maximizing-nosql-clusters-large-data-sets**

How to Build High-Performing IT Teams — Including New Data on IT Performance from Puppet Labs 2015 State of DevOps Report

Sponsor: Puppet Labs

DevOps represents a profound change from the way most IT departments have traditionally worked: from siloed teams and high-anxiety releases to everyone collaborating on uneventful and more frequent releases of higher-quality code. It doesn't matter how large or small an organization is, or even whether it's historically slow moving or risk averse — there are ways to adopt DevOps sanely, and get measurable results in just weeks.

> **http://geekguide.linuxjournal.com/content/how-build-high-performing-it-teams-including-new-data-it-performance-puppet-labs-2015-state**

WHITE PAPERS

Comparing NoSQL Solutions In a Real-World Scenario

Sponsor: RedisLabs | Topic: Web Development | Author: Avalon Consulting

Specializing in cloud architecture, Emind Cloud Experts is an AWS Advanced Consulting Partner and a Google Cloud Platform Premier Partner that assists enterprises and startups in establishing secure and scalable IT operations. The following benchmark employed a real-world use case from an Emind customer. The Emind team was tasked with the following high-level requirements:

- Support a real-time voting process during massive live events (e.g., televised election surveys or "America Votes" type game shows).
- Keep voters' data anonymous but unique.
- Ensure scalability to support surges in requests.

> **http://geekguide.linuxjournal.com/content/comparing-nosql-solutions-real-world-scenario**

NEW Forrester Study!

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

Achieving Application Delivery Velocity with a 482% ROI

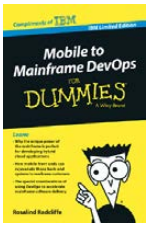
IBM commissioned Forrester Consulting to conduct its Total Economic Impact™ (TEI) study that examines and quantifies potential return on investment (ROI) for IBM UrbanCode Deploy within an enterprise DevOps environment. The study determined that a composite organization, based on the customers interviewed, experienced an ROI of 482%!

Read the Forrester Consulting study and learn learn how these enterprise organizations achieved:

- 97% reduction in the cost of releases.
- Reduction in the risk of failed deployments.
- 75% faster deployment times.

See how IBM UrbanCode brings deployment velocity while reducing release costs.

> **http://devops.linuxjournal.com/devops/total-economic-impacttm-ibm-urbancode**



Mobile to Mainframe DevOps for Dummies

In today's era of digital disruption empowered by cloud, mobile, and analytics, it's imperative for enterprise organizations to drive faster innovation while ensuring the stability of core business systems. While innovative systems of engagement demand speed, agility and experimentation, existing systems of record require similar attributes with additional and uncompromising requirements for governance and predictability. In this new book by Rosalind Radcliffe, IBM Distinguished Engineer, you will learn about:

- Responding to the challenges of variable speed IT.
- Why the mainframe is a unique and ideal platform for developing hybrid cloud applications.
- How mobile front ends can rejuvenate back-end systems to reach new customers.
- And, special considerations for using a DevOps approach to accelerate mainframe software delivery.

> **http://devops.linuxjournal.com/devops/mobile-mainframe-devops-dummies**

BRAND-NEW EDITION!

DevOps For Dummies – New Edition with SAFe®

In this NEW 2nd edition, learn why DevOps is essential for any business aspiring to be lean, agile, and capable of responding rapidly to changing customers and marketplace.

Download the E-book to learn about:

- The business need and value of DevOps.
- DevOps capabilities and adoption paths.
- How cloud accelerates DevOps.
- The Ten DevOps myths.
- And more.

> **http://devops.linuxjournal.com/devops/devops-dummies-new-edition-safe**

DOC SEARLS

Can We Save Wireless from Regulators?

The more expensive regulators make wireless—and for no good reason—the more our lives are lived inside walled gardens.

Linux was born and grew within an ecosystem of norms, not laws. Those norms were those of programming (C), operating systems (*NIX), command shells (bash, etc.), e-mail (SMTP, etc.) licenses (GPL, etc.) and Internet protocols (TCP/IP and the rest).

Had Linux and the Internet been left up to the world's big operating system and network providers, we never would have had either one. Instead, we would have had what business giants and their captive regulators are inclined to believe both actually are: "intellectual property" and billable "services".

"Free" and "open" are the adjectives that best describe the development ethos that allowed Linux and the Internet to happen. Yes, there were regulations around, but Linux and the Net grew up outside the scope of what Bob Frankston calls The Regulatorium (http://www.frankston.com/public/?name=RealityVRegulatorium). To a blessed degree they still do, but that degree is getting narrower and less blessed as more of our computing and communicating moves to mobile devices.

For the most part, those devices are not native to the open Internet. Although they can operate on the Internet (and in the case of Android devices, run on a breed of Linux), they are native by design to the walled commercial gardens of cellular telephone companies. These are regulatory

As artificial scarcities go, spectrum might be the largest and most expensive in world history.

zoo animals that also happen to run the zoo—at least in how they conceive wireless communications and influence regulators.

Perhaps the most expensive and retro conceptual framework for wireless communications is one that carriers and regulators buy completely and the rest of us hardly ever question. That framework is *spectrum*. As artificial scarcities go, spectrum might be the largest and most expensive in world history. And yet, it remains the prevailing frame for understanding wireless, both for regulators and for ordinary muggles.

Take, for example, this story: "A major New York TV station could win $900 million—if it goes off the air. Here's why", by Brian Fung, in the October 16, 2015 issue of *The Washington Post*. In it, he points to this list of opening reverse auction prices on the spectra occupied by TV stations in every US market, including those in its overseas territories: http://transition.fcc.gov/Daily_Releases/Daily_Business/2015/db1016/DA-15-1191A2.pdf. Topping the list is WCBS-TV in New York, better known there as Channel 2 (even though it actually operates on Channel 33, spanning 584–590MHz). Price: $900 million. He explains:

> The figures represent the maximum amount each broadcaster could receive for participating in a never-before-tried auction of wireless airwaves, one that's designed to transfer control of that invisible real estate to wireless carriers such as AT&T and T-Mobile. Cellular providers say they need access to more of the radio spectrum to build out next-generation mobile data networks. (All wireless data, from TV signals to 4G LTE, ride atop spectrum, a finite resource.)

What he's talking about here is auctioning off over-the-air TV channels to wireless carriers, in faith that the wireless carriers can do more with those channels (collections of adjacent frequencies) than the TV stations can—which is probably true. The FCC is also making it possible for stations to bid on other lower-frequency channels that are less

desirable for cellular wireless but just fine for over-the-air TV. Since most people watch cable rather than over-the-air TV, the stations might not want to buy other channels at all. A for-sure end result of all this is that over-the-air TV will end up even more dead than it is already.

But, the framing is what matters most here. "Real estate" and "a finite resource" are taken as givens—independent variables, beyond question.

Yet, what we're also talking about here is what Bob Frankston correctly calls "selling the color blue"—meaning that spectrum shouldn't be for sale at all.

Wireless communications has no more need for "providers" to buy spectrum than any of us have for providers selling us the color blue in order for us to see or use it. Rather than explaining the rest of what that means, I'll turn the floor over to David P. Reed, one of the Internet's founding figures and a scientist of the first water, writing to a list I'm on:

> So there's a fundamental technical question regarding radio systems architectures, which relates to sharing of the medium (colloquially, the airwaves). I've written and spoken about this for decades now, as an engineer. Here's the question:
>
> Is it technically necessary to build regulation into the burgeoning, and highly localized, growth of communications? (And a corollary: why can't computationally powerful radio-networked systems just cooperate, for the mutual benefit of all users of the airwaves?) The answer appears to be No (and They Absolutely Can).
>
> Now the key here is "cooperation".
>
> There's a great incentive for cooperation in communications. That's the entire basis for the success of the Internet! The fact that standardizing on a single abstract framework for communications that is technologically agnostic both in regard to the transport infrastructure (and airwaves) and the applications has actually completely upended the communications industry and how all citizens of the world do their interaction *should* be obvious....
>
> Radio networks can sense the airwaves and modify their behavior—as a network—

cooperatively to make the best of what they are capable of doing. Now that we have software and powerful signal processing in every radio chip, cooperation is not that hard.

And there's a huge return on "effort" to cooperation....We know that what is possible is far better than what is achieved by cooperative systems today....

Cooperation at the local level is easy because people control their own real estate. We don't need the FCC to tell us that we can turn devices on or off if they mess up our local environment. That's the genius of Part 15 (http://www.ecfr.gov/cgi-bin/text-idx?SID=2c611742f6c4a4a15e73ae11cd4dd12a&mc=true&node=pt47.1.15&rgn=div5).

So the combination of incentives to cooperate to get better use out of the airwaves (now technically feasible, far more than ever before, because of Moore's Law, information theory, and digital signal processing, as well as amazing improvements in analog semiconductor technology for sensing and transmitting), AND the desire to make our *local* environments work well *should* be enough.

But in DC...the issue is "who wins?" [When] the real question should be: what serves the public interest?

And in terms of local communications, what serves the public interest is locally managed, freely chosen, technologies that cooperate and *interoperate* to at least some extent.

Everything David talks about here is outside the framework of spectrum. It's just radios talking to radios, the best ways they can. These new-generation radios are like the Internet that way. TCP/IP, the Internet's founding and persistent protocol suite, specifies a "best effort" for getting data from any one end to any other, regardless of who owns and operates the "pipes" (wireless or otherwise) between those ends. The possibilities it opens are boundless. But you can't see them if you remain stuck inside old framings.

On one of the many Geek Cruises that *Linux Journal* hosted back around the turn of the Millennium, we visited the Aricebo Observatory in Puerto

Rico, a primary collection point in SETI, the Search for Extraterrestrial Intelligence. SETI looks mostly in the microwave window of frequencies where the most can be heard across outer space and presumably where the most is being transmitted by distant intelligent beings. In the beginning of SETI, a framing assumption was that extraterrestrials possibly would be radiating radio and television in a manner like unto our own systems of that time.

Back then, TV stations radiated up to five million watts toward the horizon (near the lower-frequency range of that microwave window). The most powerful digital transmitters of today's stations are still one million watts. Meanwhile, cell tower transmissions are a few watts at most, and the phones in

Resources

Bob Frankston: http://www.frankston.com

The Regulatorium: http://www.frankston.com/public/?name=RealityVRegulatorium

"A major New York TV station could win $900 million—if it goes off the air. Here's why": https://www.washingtonpost.com/news/the-switch/wp/2015/10/16/fcc-weve-fired-the-starting-gun-on-a-massive-auction-of-wireless-airwaves

Brian Fung: https://www.washingtonpost.com/people/brian-fung

Reverse Auction Opening Prices: http://transition.fcc.gov/Daily_Releases/Daily_Business/2015/db1016/DA-15-1191A2.pdf

WCBS-TV New York, Channel 33: http://fccinfo.com/CMDProEngine.php?sCurrentService=TV&tabSearchType=Appl&sAppIDNumber=1317336

Television Channel Frequencies, UHF: https://en.wikipedia.org/wiki/Television_channel_frequencies#Americas_.28most_countries.29.2C_South_Korea.2C_Taiwan_and_the_Philippines

David P. Reed: https://en.wikipedia.org/wiki/David_P._Reed

Aricebo Observatory in Puerto Rico: https://en.wikipedia.org/wiki/Arecibo_Observatory

SETI: https://en.wikipedia.org/wiki/Search_for_extraterrestrial_intelligence

Search for Extraterrestrials—Microwave Window: https://en.wikipedia.org/wiki/Search_for_extraterrestrial_intelligence#/media/File:TerrestrialMicrowaveWindow.jpg

our pockets use fractions of a single watt. In other words, all are so weak, on purpose, that even the most sensitive receivers light years away have no hope of detecting them.

Cellular communications, which relies on local and low power transmission, was, and remains, a huge and highly original invention—one that made it possible to crowd many more communication paths into local and worldwide geography than ever would have been possible with brute-force transmissions of the old broadcast school.

What cellular did was free us from believing that bigger was better. Now we need the same kind of liberation from the belief that spectrum is scarce. Because it's not. And selling it makes no sense, except to business as usual, its captive regulators and the billions of people who still don't know better. Let's change that.■

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!