Hard Drive Rescue with a Raspberry Pi

LINUX JOURNAL

Since 1994: The Original Magazine of the Linux Community

WATCH: ISSUE OVERVIEW

SEPTEMBER 2016 | ISSUE 269
http://www.linuxjournal.com

Firewalld in Multi-Zone Configurations

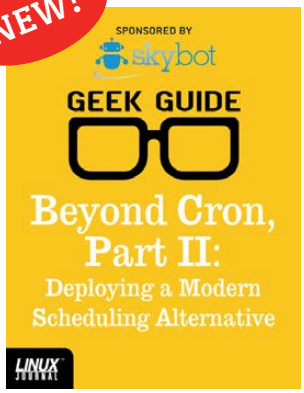
+

Home Networking with Low-Power ARMs

Cool Project: Play Nintendo Using Emulation on an RPi

A LOOK AT SNMP AND ITS FUTURE

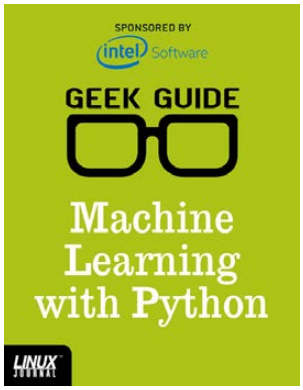
NEW!



Beyond Cron, Part II: Deploying a Modern Scheduling Alternative

Author: Mike Diehl

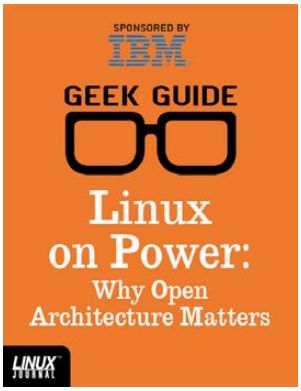
Sponsor: Skybot



Machine Learning with Python

Author: Reuven M. Lerner

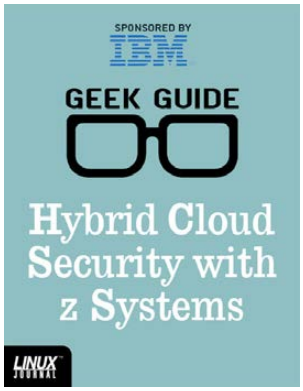
Sponsor: Intel



Linux on Power: Why Open Architecture Matters

Author: Ted Schmidt

Sponsor: IBM



Hybrid Cloud Security with z Systems

Author: Petros Koutoupis

Sponsor: IBM

LinuxONE: the Ubuntu Monster

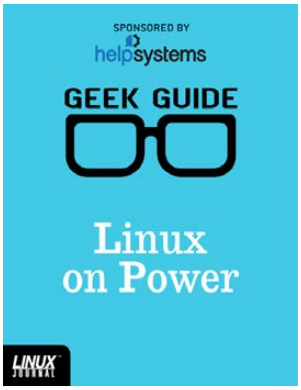
Author: John S. Tonello

Sponsor: IBM

Ceph: Open-Source SDS

Author: Ted Schmidt

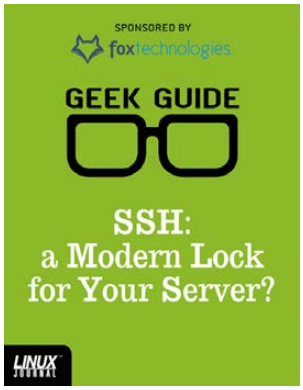
Sponsor: SUSE



Linux on Power

Author: Ted Schmidt

Sponsor: HelpSystems



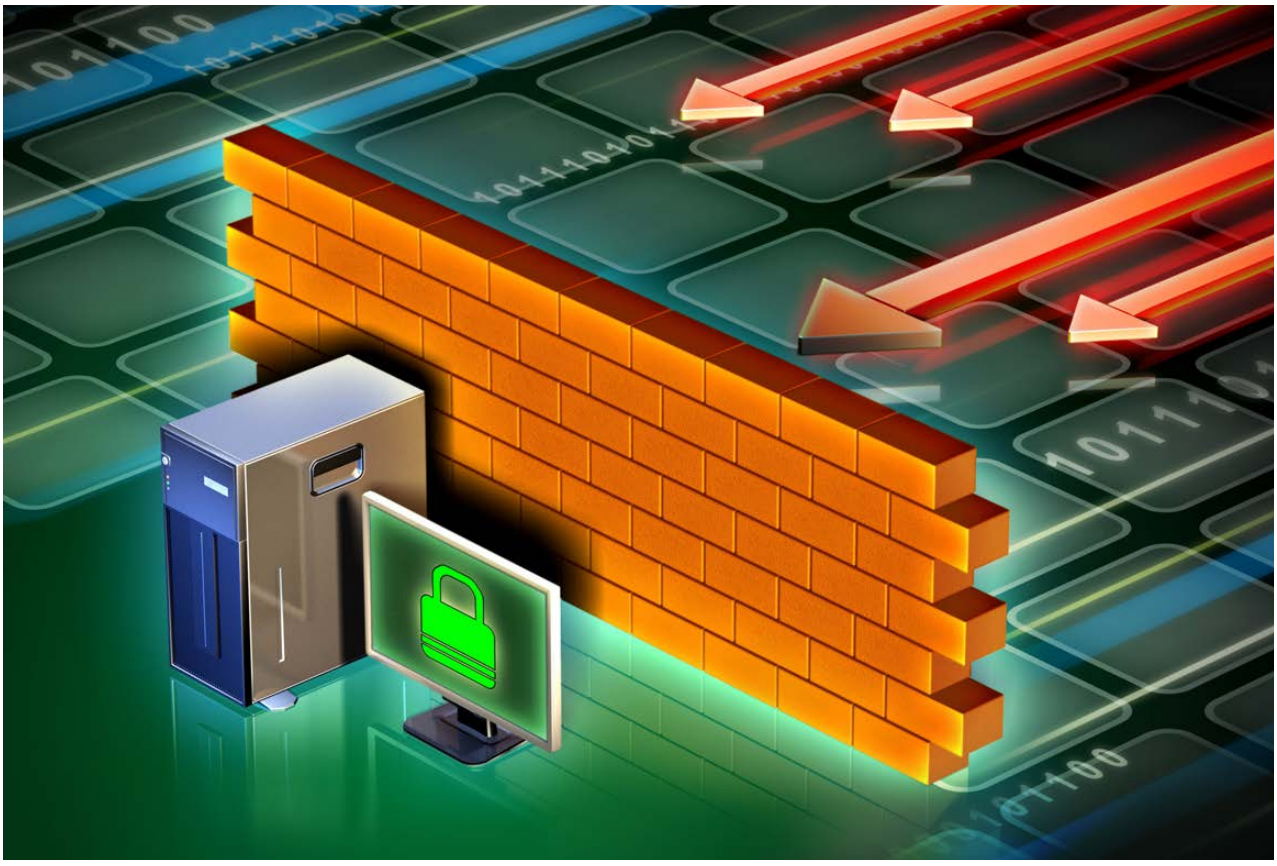
SSH: a Modern Lock for Your Server?

Author: Federico Kereki

Sponsor: Fox Technologies

CONTENTS

SEPTEMBER 2016
ISSUE 269

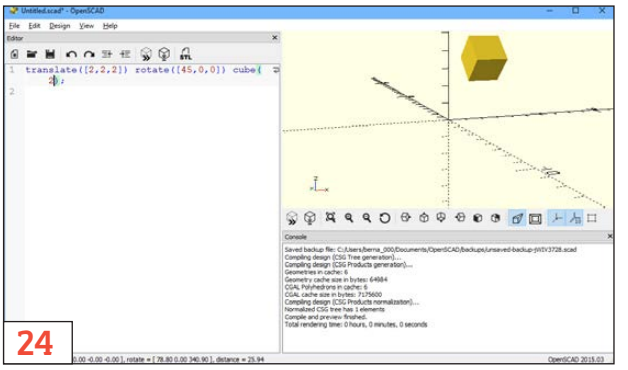


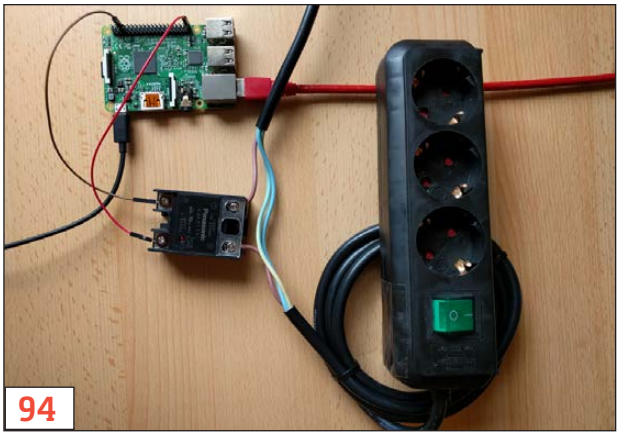
Cover Image: © Can Stock Photo Inc. / Andreus

FEATURES

COLUMNS

IN EVERY ISSUE





ON THE COVER

- Hard Drive Rescue with a Raspberry Pi, p. 94
- Firewalld in Multi-Zone Configurations, p. 80
- Home Networking with Low-Power ARMs, p. 38
- Cool Project: Play Nintendo Unsing Emulation on an RPi, p. 46
- A look at SNMP and Its Future, p. 62

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

LINUX JOURNAL

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

Contributing Editors

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

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

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

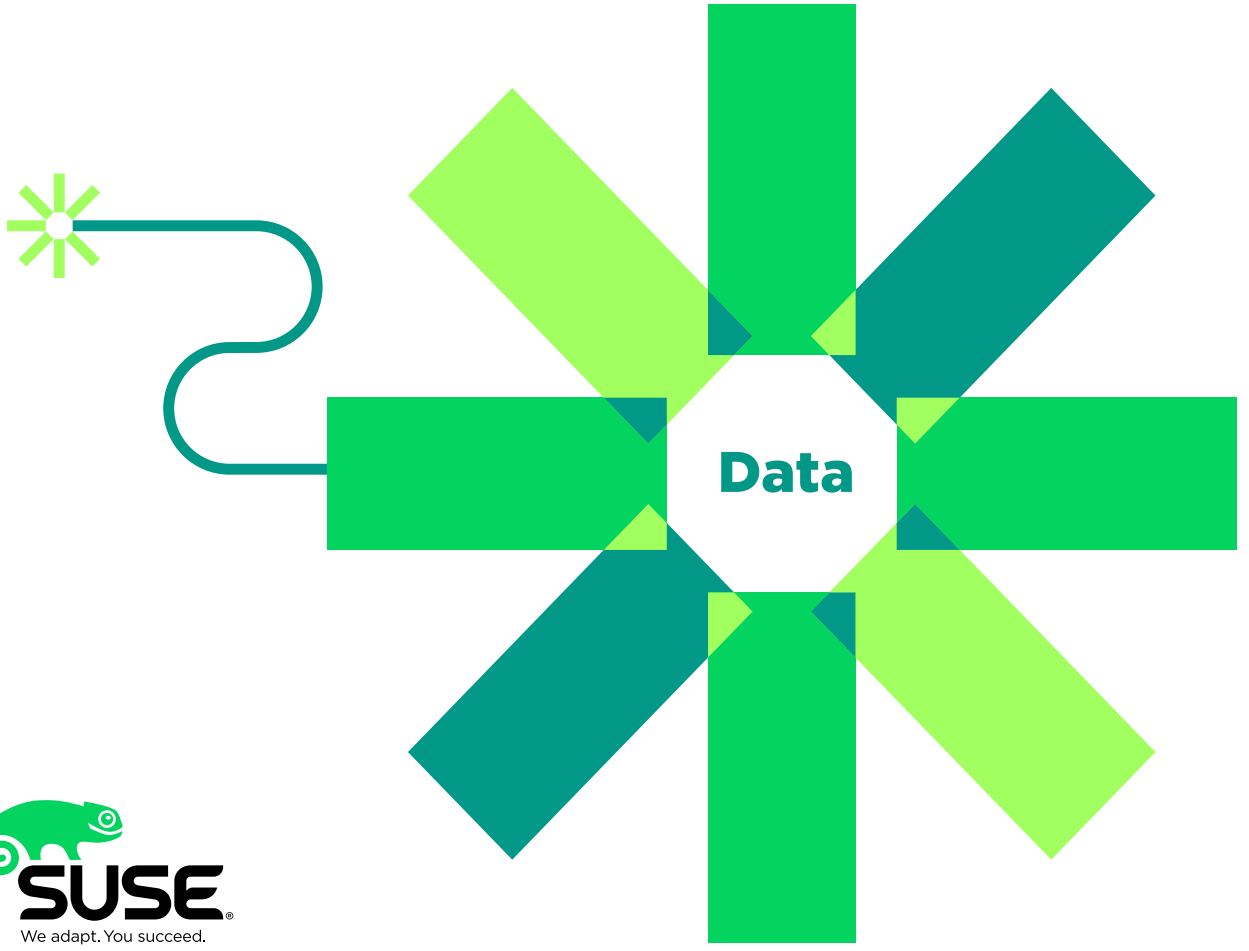
STORAGE
REDEFINED:

You cannot keep up with data explosion.

Manage data expansion with SUSE Enterprise Storage.

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

suse.com/storage



SUSE
We adapt. You succeed.

Doing Old Things in New Ways

SHAWN POWERS

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

Our new house is built into the side of a hill. It's not quite a Hobbit hole, and our doors are rectangular as opposed to round, but it still has its challenges. The first challenge we face is mowing the front yard. There's an approximate 15' drop over the 20' from the house to the road. The safety concerns aside (mowing side to side is dangerous and really difficult, and mowing downhill is asking to get your toes cut off), with such a steep angle I couldn't keep the mower running! I went through several "solutions" before I found one that works. I bought a big weed whacker, thinking I'd mow the lawn that way. Then I bought a plug-in model electric mower, which somehow tries to eat its own cord at every turn. The final solution was a battery-operated mower. It works great, and the batteries are shockingly long-lasting. Ten years ago, the thought of a battery-powered lawn mower would have been ridiculous. As technology advances, the impossible becomes the practical. So, I say this fairly often: "It's so awesome to live in the future!"

Dave Taylor starts off the issue this month by traveling to Mars. Well, that's not entirely accurate. He starts off on the process of coding a text-based game simulating the landing procedure for a spacecraft on the surface of Mars. If you've ever played *Lunar Lander* and wanted to test the math, Dave's column will be right up your alley.

Most of us have a pile of servers somewhere in the house

VIDEO: Shawn Powers runs through the latest issue.

or garage that we use for experiments and to store our files. Kyle Rankin's server is in his garage (I'm jealous, I don't have a garage), and he recently decided it was time to upgrade. Rather than buying a huge server, however, Kyle decided to create a powerful NAS device using the smallest and most power-efficient server possible. This month he walks through his process and explains how he landed on the final solution. It's tinier and far more efficient than anything available just a few years ago. If you want to revamp your server closet or save some electricity by replacing old power hogs, check out his column.

I decided I wanted to do old things in new ways, but the things I wanted to do were all game-related. I spent a shameful number of weekends in my youth playing Nintendo games. Although Nintendo is releasing a console this winter with some classic NES games embedded into the system, there's something special about playing with the original controllers. This month I turned a Raspberry Pi and a cigar box into an emulation machine for Nintendo and Super Nintendo. If you ever played *Super Mario Brothers 3* or had fever dreams about building experience points by killing Slime Molds, come play along.

We have a guest columnist this issue talking about SNMP. Andrew Kirch explains SNMP in simple terms and shares his frustrations over the lack of development it's gotten during the past decade. SNMP is more than just a way to read network data and build throughput charts. It's a two-way protocol that allows for actual management of devices. For a look at how SNMP works and why we should be giving it more attention, be sure to read Andrew's column!

Nathan R. Vance and William F. Polik teach how to build a wall. Specifically, they describe how to use firewalld in multi-zone configurations. Firewalld makes building firewall rules much easier, and that clarity allows for more complex sets of rules that still make sense. Nathan and William take the daunting concept of firewall zones and make it easy to understand. They're followed by another Andrew, Andrew Nii Addo, who gives us a lesson on rescuing hard drives with a Raspberry Pi. For years I kept a full-blown tower on my workbench for the inevitable hard drive rescue operations. With Andrew's guidance, you can use a simple Raspberry Pi to do the same task in a far more convenient way.

The Linux kernel itself was created to do something old in a new way. In fact, pretty much everything we do with technology is designed to do old tasks in more efficient ways. Sometimes it's important to stick with tried and true methods. So we've also included the usual tech tips, product announcements and technology tidbits you've come to expect from every issue of *Linux Journal*. I learned a lot this month, and as a reward, I plan to play Nintendo. Maybe not as long as I did in high school, but I think there are still a few Slime Molds left in the forest who need to taste my sword!■

LETTERS

PREVIOUS
Current_Issue.tar.gz

NEXT
UpFront

Tux in Public!

On a recent trip to western New York State, I sat down to use the public computer in the hotel lobby and was shocked to find it running Ubuntu with Unity. It was hooked up to a multi-function printer device like you normally find in hotel business centers, and while not a fan of Unity, I was able to get online easily and it worked fine. This is the first time I've seen Tux in a place so visible to the general public. How great is that?

—Cranky Frankie

Shawn Powers replies: *I had the same experience at a local library here in Michigan. There was an entire room of computers running some sort of Linux with KDE!*

About Let's Encrypt

I've been a *Linux Journal* reader for more than five years, although the English is sometimes strange for me.

After reading Andrei Lukovenko's article "Let's Automate Let's Encrypt" in the June 2016 issue, I decided to write you to congratulate you for all of your great work and, even more, to thank Andrei. I was searching for a way to use SSL on my recently actualized Request Tracker server, and this article came as a ring on my finger! In less than five minutes, my server used a certified certificate, and there were no more warnings from the browser about untrusted sites. *LJ* is a great journal, with great articles!

—Jerome Verleye

Proxmox Kernel Is Not Debian-Based

John S. Tonello's article "The Tiny Internet, Part II" in the July 2016 issue introduces Proxmox VE. However, its kernel is either RHEL or Ubuntu based on a Debian environment, not directly based on Debian. This is how we choose server-grade hardware for running Proxmox VE, according to the certified or compatibility lists from Red Hat or Canonical. FYI.

—Cheng-Han Wu

John S. Tonello replies: *Thanks for the clarification about Proxmox VE. To clarify, is it fair to say the kernel is pure Red Hat or Ubuntu, but the environment is something like Debian 6/7? I wanted to let readers know that the CLI interface is closer to a Debian experience than, say, Ubuntu, and avoid delving too deep into the kernel itself, which is a more advanced topic.*

Thanks for writing. Proxmox VE is a great tool and one more people should know about!

Cheng-Han Wu replies: Here's the wiki about Proxmox VE kernel: https://pve.proxmox.com/wiki/Proxmox_VE_Kernel. I think it's better to say "Red Hat or Ubuntu based", because it's actually not "pure". The team has done some customization themselves, for example, adding container support like OpenVZ.

The CLI part is indeed Debian (5/6/7/8 for Proxmox VE version 1/2/3/4), along with their own repo in APT system: https://pve.proxmox.com/wiki/Package_repositories.

It's really a convenient tool for building up a PaaS virtualization host with Proxmox VE in minutes. It's good to see that *LJ* has so many pages to introduce it!

4232, a Short Animation Film Using Software Libre

My name is Ernesto Bazzano (Bazza), and I'm an Argentinian artist. I'm making a short animated film called *4232*, using libre software with GNU/Linux—software made and supported by communities of

programmers and developers worried about freedom.

Among other software, I use the following:

- GIMP: to clean the scanned images, and for coloring and image processing.
- Synfig Studio: to mount the animation frame by frame.
- My own programs: the drawings of *4232* are hand-made. I develop programs that allow Synfig to incorporate real drawings.

In addition to making my short animated film, I generate programs that can help others artists in their work.

Animation is a very complex activity, thought to be created by a lot of people, in big studios and with big budgets. I'm making applications to speed up and make the whole animation process easier. This way, it's a little bit easier for artists with the desire but with a small budget to make their dream animations.

Here's a synopsis of *4232*.

4232 is a story about a post-apocalyptic future. In a metropolis surrounded by a big dome, the aristocracy has converted poor people into robots, undressing them of all humanity. They are the robot-workers used for exploitation of the last resources on the Earth, which is turning into a big barren desert. Some people have escaped this metropolis and found shelter in small constructions outside the dome, surrounded by desert and pollution.

One day, the inhabitants of one of these shelters finds a robot in the desert—number *4232*. Unlike the other robots, *4232* preserves his consciousness and relates to them how he escaped the metropolis. So the people of the desert start to wonder why *4232* is different and see him as hope for their own robot-ized friends and families to

recover their consciousness.

Go to http://4232.cf for more details.

—Ernesto Bazzano

Shawn Powers replies: *Ernesto, wow! It's great to see open-source software used so extensively. In my quick perusal of your site, I wasn't able to find a sample of the end product, but your process looks fascinating. Please consider pitching an article idea our way, I'm sure our readers would love to learn more about your methods.*

PHOTO OF THE MONTH

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

WRITE LJ A LETTER

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

RETURN TO CONTENTS

PREVIOUS
Letters

NEXT
Editors' Choice

diff -u

What's New in Kernel Development

Intel has been working on some code to support its **ISH** (Integrated Sensor Hub) chips for motion detection and other localized data. Intel's code aims to provide a similar level of Linux support as other sensor hardware, but without changing the **ABI** (Application Binary Interface), so that existing code could run on hardware containing ISH chips, without needing to rewrite the source code.

Srinivas Pandruvada was the Intel engineer to send out the patches and request feedback. He offered the caveat that the code was still in an early state, and he wanted to make sure it was heading in the right direction.

Several kernel developers were happy to see these patches, though they all reported various problems with the code and difficulty performing tests. And while Srinivas wasn't able to produce updated patches during the conversation, it's clear that ISH support will be in the kernel soon.

Patching a running kernel without having to reboot is insanely cool. The way it works is that the patched code and the original code coexist on the system until nothing is using the old code. Then, the calling paths are updated to point only to the new code. Assuming

the new code doesn't do anything fundamentally incompatible with the old, the system should experience no trouble.

One problem with this is that the new code has to wait until no other thread is running in the old code, and only then do the switch. This could take an arbitrary amount of time, during which the kernel must rely on the old code. **Josh Poimboeuf** recently submitted some changes to the **live-patching code** to make the kernel switch to newly live-patched code on a per-process basis. This way, as a given process exited the old code, the next time it needed that code, it would see only the new, patched version. Over time, more and more processes would switch over to the new code, until the old code wasn't used at all.

There are various insane elements to the whole issue. Live patching is nuts. For instance, **Jessica Yu** pointed out that if a process was sleeping in an uninterruptible state, it might never be able to let go of the old code, so that code could never be removed after a patch. And as **David Laight** pointed out, such a process would need special handling by the kernel, so that the normal ways of dealing with hung tasks wouldn't trigger anything bad with the live-patching procedure.

Resource locking is crucial for a multitasking, multi-user system, but it's difficult to get right. Locks need to be as fast as possible, or they risk slowing down the system significantly. They also need to lock only the specific resource they want, or they risk making other processes wait around. That was the problem with the **Big Kernel Lock** (BKL) that has since been replaced with a variety of smaller locks.

Peter Zijlstra recently noticed that some kernel code would check a lock too early and potentially miss important state changes. He posted some code to fix it, but his proposed interface apparently relied a little too heavily on the user understanding the inner workings of those locks. **Linus Torvalds** objected very strongly, saying, "NAK. We don't start adding more of this 'after_ctrl_dep' crap. It's completely impossible to understand, and even people who have been locking experts have gotten it wrong. So it is *completely* unacceptable to have it in drivers. This needs to be either hidden inside the basic spinlock functions, *or* it needs to be a clear and unambiguous interface."

Fortunately, as **Tejun Heo** pointed out, the whole question might turn out to be moot, since the portion of code that contained Peter's problem might not be needed anymore and simply could be removed from the kernel. Peter replied that this would be perfectly acceptable to him, and the thread petered out.

Virtualization is one of the hairiest parts of kernel development. It's the effort to make the kernel appear to be running user code on multiple separate systems, all the while running everything on just one. And, since the virtual systems need to communicate their needs and the users' needs to the real system underneath, it can be hard to give users absolutely every privilege they expect.

Serge E. Hallyn recently tried to address one of these sorts of issues. If users untar a tarball, the **extended attributes** (xattrs) of the extracted files need to be preserved, according to the security privileges of those users. And if the users are on a virtualized system, they may have privileges on that system that appear to be greater than the privileges they'd have on the real system underneath. For example, regular users can be root on their personalized virtual systems, but they're still just regular users underneath.

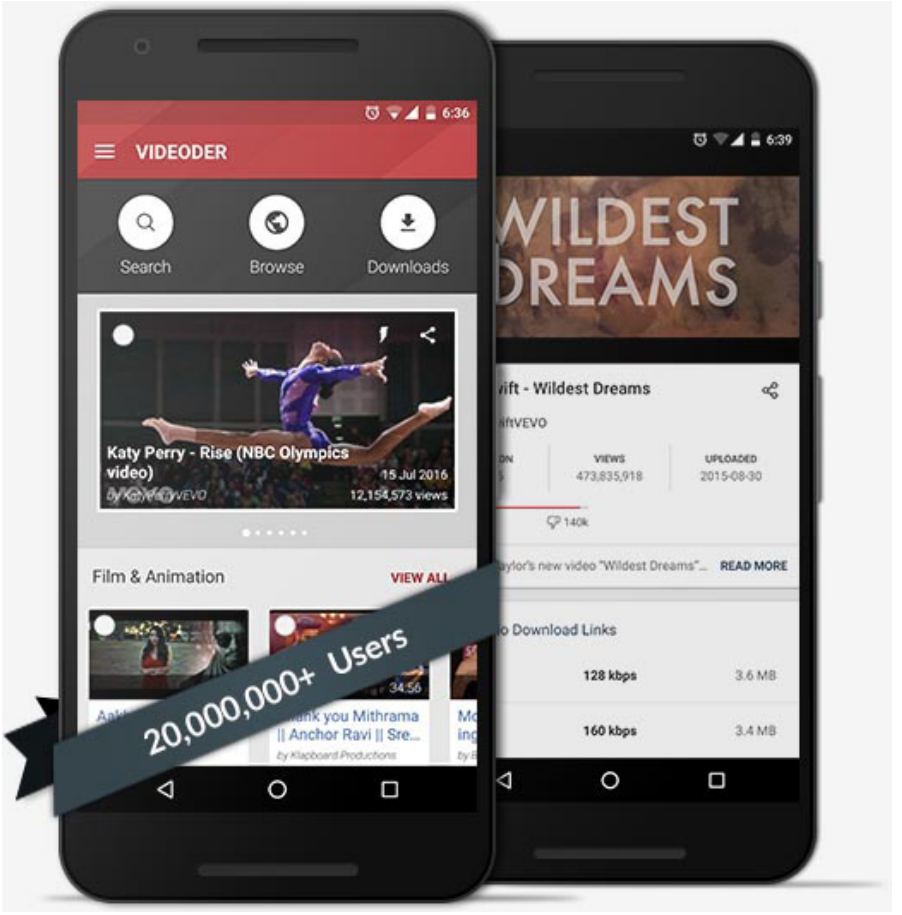
Serge sent in some patches to prevent users from giving files more powerful xattrs than they deserved, by spoofing the real xattr with an alternative that had lesser security privileges. However, as **Eric W. Biederman** put it, creating alternative xattrs actually would have the effect of letting otherwise identical file repositories get out of sync with each other.

Eric suggested simply keeping the xattrs as stored in the tarball (or anywhere else), but having the kernel keep track of the true security level of the user who unpacked it. The kernel could, therefore, prevent the user from improperly gaining any extra security privileges.

Serge probably will implement Eric's idea, although **Mimi Zohar** (and Serge himself) particularly appreciated the sneaky way Serge's original patch would detect the attempt to gain improper security powers and use alternate xattrs to prevent it. But, sometimes the simpler idea is better.—Zack Brown

Android Candy: Did You See That Cat Video?

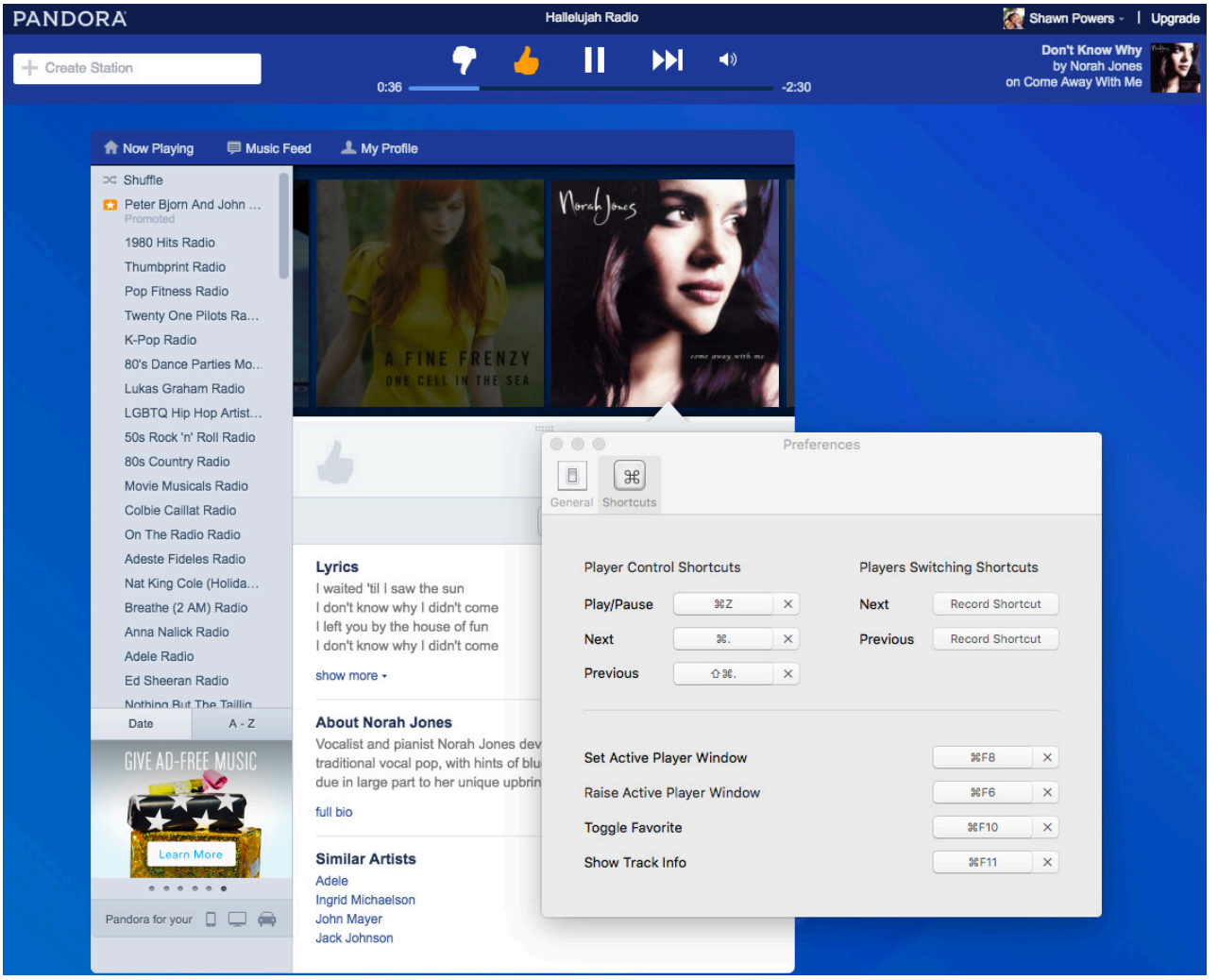
I love cruising the internet looking for amusing videos. I'm pretty sure everyone who works at a computer finds themselves down weird YouTube rabbit holes from time to time. Unfortunately, I never can seem to find the videos I like the best when I'm trying to show someone else. Thankfully, if you're on an Android device, you can save YouTube videos locally.



(Screenshot from https://www.videoder.net)

Mind you, an app for ripping YouTube videos is not something you'll find in the Google Play store. But, if you're okay with violating YouTube's terms of service, or if your country doesn't have legal issues with downloading a copy of YouTube videos, check out Videoder (https://www.videoder.net). It's a downloadable APK installer that will allow you to save a local copy of any YouTube video you happen across! —Shawn Powers

Non-Linux FOSS: Control Web-Based Music!



I like Pandora. I like it because it doesn't require me to know anything other than whether I like the current song. I'm sure other music services offer more features or a larger catalog, but Pandora is simple. So am I.

One of the frustrating things about using the web-based version of Pandora is that getting back to the proper tab can be frustrating. Thanks to an oddly named open-source tool for OS X, when I'm on a Mac, I can map keyboard shortcuts to the web player even if it's in the background.

The BeardedSpice program is open source and available to download at http://beardedspice.github.io. At first, I wasn't sure how to configure it, but that's because there's very little to configure. Once I set the keyboard shortcuts I wanted, it's just a matter of pressing them while the music is playing! It's simple, effective and it's free (both kinds of free). If you use OS X, grab a copy today!—Shawn Powers

(Screenshot from http://www.minetest.net)

Wish *Minecraft* Were Open Source?

Minecraft is still a huge hit and loved by millions all around the world. I personally don't really understand its popularity, but I can understand a love for open-source software. The folks over at http://www.minetest.net have created a *Minecraft*-like software package that is fully open source.

Minetest supports multiple players, single-player mode and also has tons of mods that are free to download. You also can make your own mods and implement them along with other mods. If you truly understand the passion people feel for *Minecraft* but are hesitant to use it because of its closed-source nature, check out *Minetest*. The graphics aren't very detailed, but that's by design! —Shawn Powers

peer1 hosting

Where every interaction matters.

break down
your innovation barriers

power your business to its full potential

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

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

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

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

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

3D CAD Modeling

Linux has several options available for handling CAD (computer-assisted drafting) projects. In this article, I cover OpenSCAD (http://www.openscad.org), which is available on Linux, Mac OS X and Windows.

Most Linux distributions should have it—for example, you can install it on Debian-based distributions with:

```
sudo apt-get install openscad
```

This will install all of the relevant binaries, along with several example files.

Many other CAD systems, such as Blender, are focused on the graphics of the rendering in order to produce very pretty pictures. OpenSCAD focuses more on the practical side of 3D modeling. It also isn't designed to do interactive model construction, but instead uses CAD specification files and then renders the final object, almost like compiling an executable.

You can start OpenSCAD from a menu entry in your desktop environment or by executing the `openscad` command from a terminal window. When it first starts, you'll see an initial window where you can select from a list of recent files or choose from the list of installed example files.

To get started, let's look at one of the example files—in this case, let's look at the first one: CSG.scad. The default layout consists of three separate panes. The main pane on the left-hand side is the main editor where you can define all the parts of your 3D design. The right-hand side is divided into two more panes. The top half is where the final 3D-rendered object is displayed. The bottom half is a console pane where messages are displayed. These messages could include messages from the rendering section. The viewing pane is a standard 3D viewing pane.

You can click and drag the display to rotate the view of your 3D object. There is also a set of icons at the bottom of the viewer where

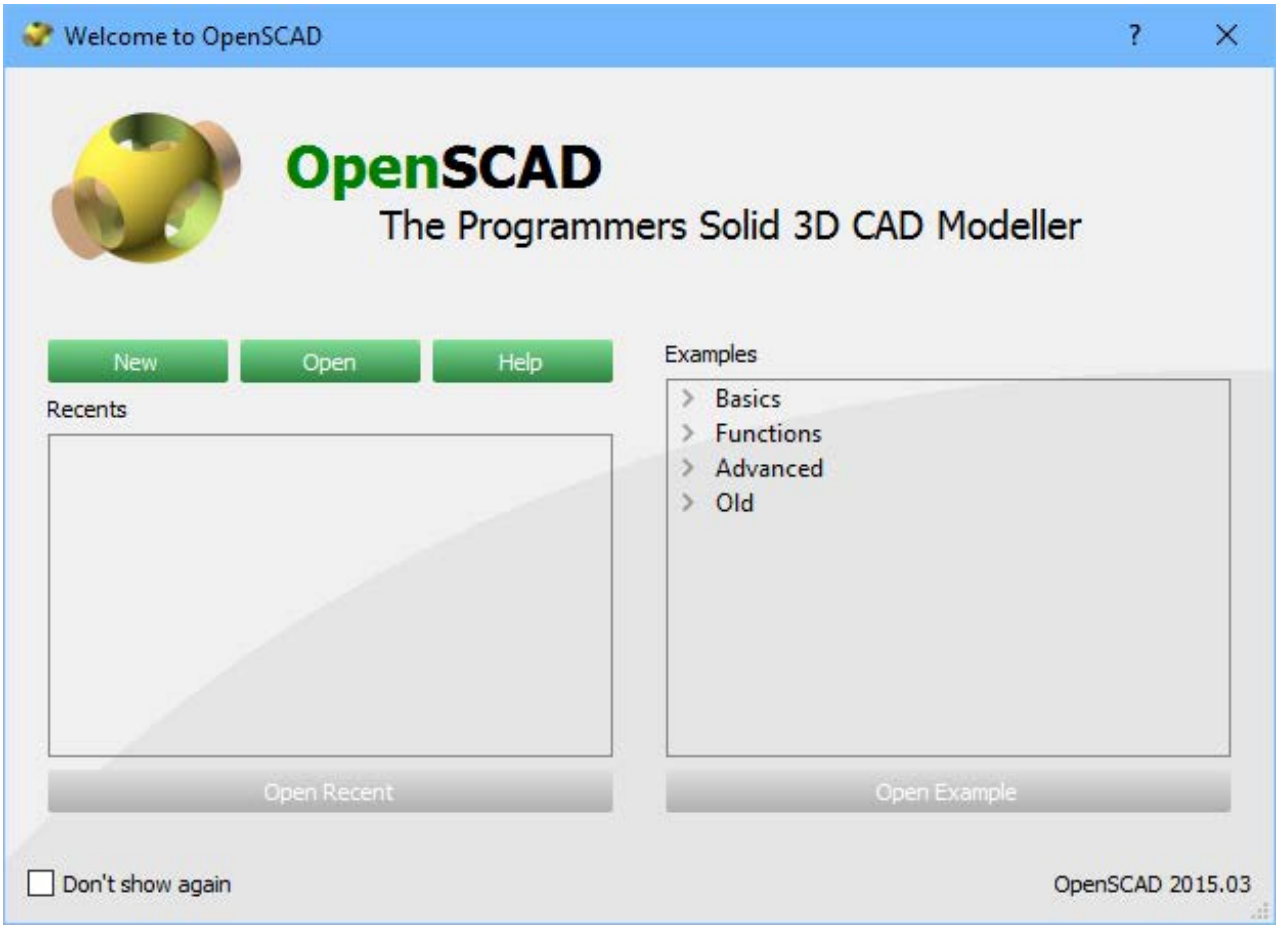


Figure 1. When you first start OpenSCAD, you'll see a listing of the most recent projects along with a selection of examples.

you can zoom in or out, rotate or select one of the standard views. Additionally, a set of icons at the top of the editor window gives you access to the most common functions. Here, you can open or save projects, as well as preview or render your object.

When you are ready to start a new project, you can click on the menu item File→New or press Ctrl-N. This will pop up a new window with an empty editor. Unlike many other CAD systems, OpenSCAD doesn't include a toolbox of objects that you can click and drag to build up your object. Instead, you need to type in the specifications for each of the elements for your design. As a simple example, you could add a cube with dimensions 2x3x4 by typing:

```
cube([2, 3, 4]);
```

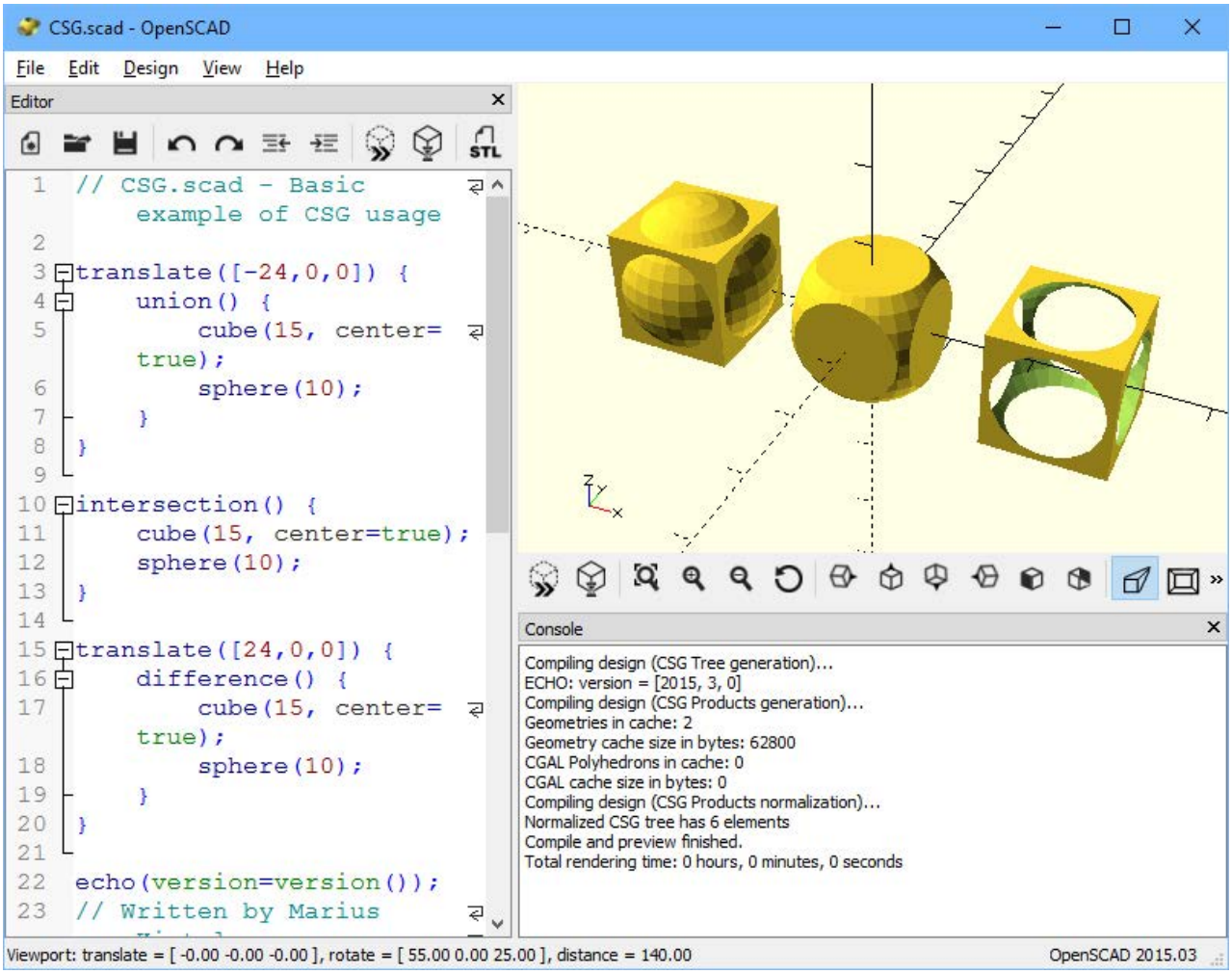


Figure 2. The main window for OpenSCAD, where you can define the objects to be rendered.

You won't see anything at this point within the viewer pane. In order to trigger a rendering, you can do a preview by clicking the menu item Design→Preview or pressing F5.

A number of basic objects are available, such as spheres, cylinders and polygons. In order to do things with those basic units, OpenSCAD provides a number of transformational functions that you can apply to them. For example, you can move an object with the translate function:

```
translate([5,0,0]) {
    sphere(1, center=true);
}
```

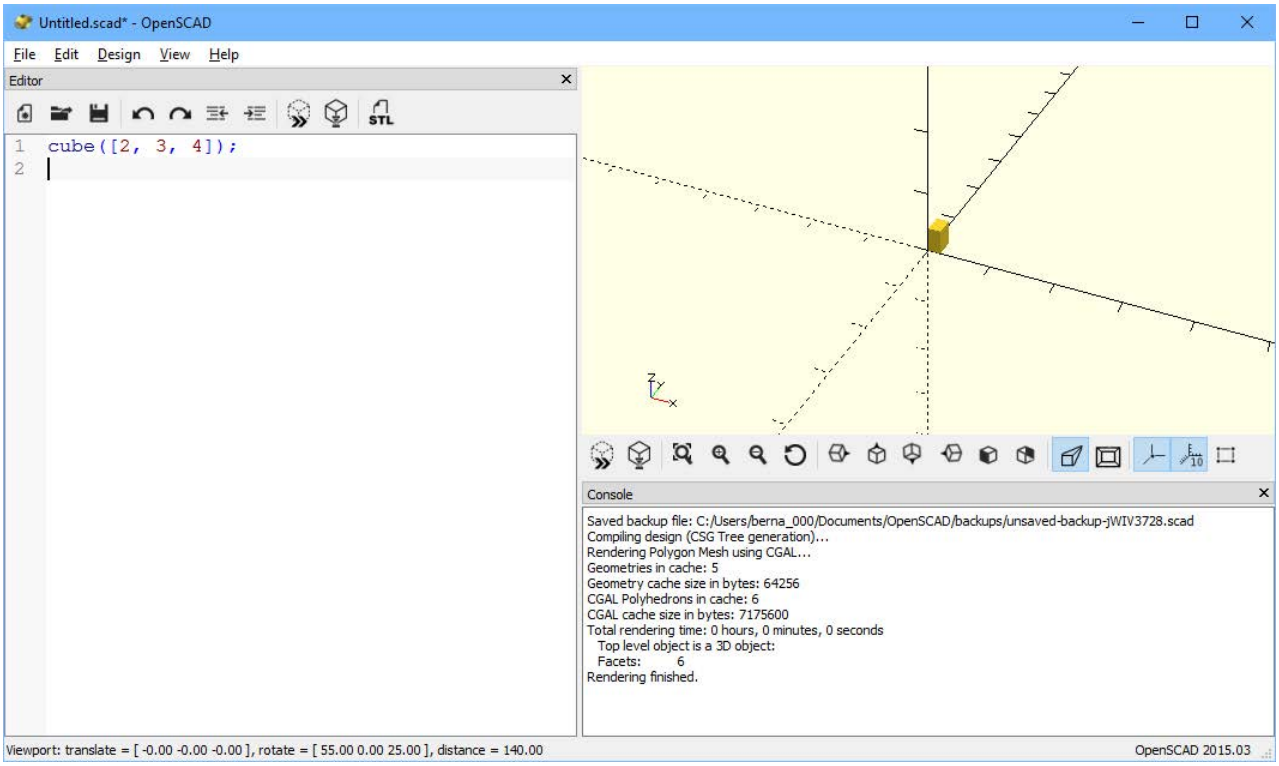


Figure 3. As a "Hello World" example, you can draw a basic cube with a single command.

OpenSCAD projects are based on a tree structure. So transformational functions, like `translate`, apply to the children of the function. These children are all grouped together within curly braces.

You also can apply a number of other transformations, such as mirroring, scaling, coloring, rotating or offsets. You can combine multiple transformations simply by adding them one after the other on a single line. For example, you could move and then rotate a cube with this command:

```
translate([2,2,2]) rotate([45,0,0]) cube(2);
```

In this case, you also could place the cube command on the same line, since there is only one child.

Along with 3D elements, you can build your object out of 2D elements. Several basic elements are available, such as circles, squares, polygons and even text elements. You can use them to build

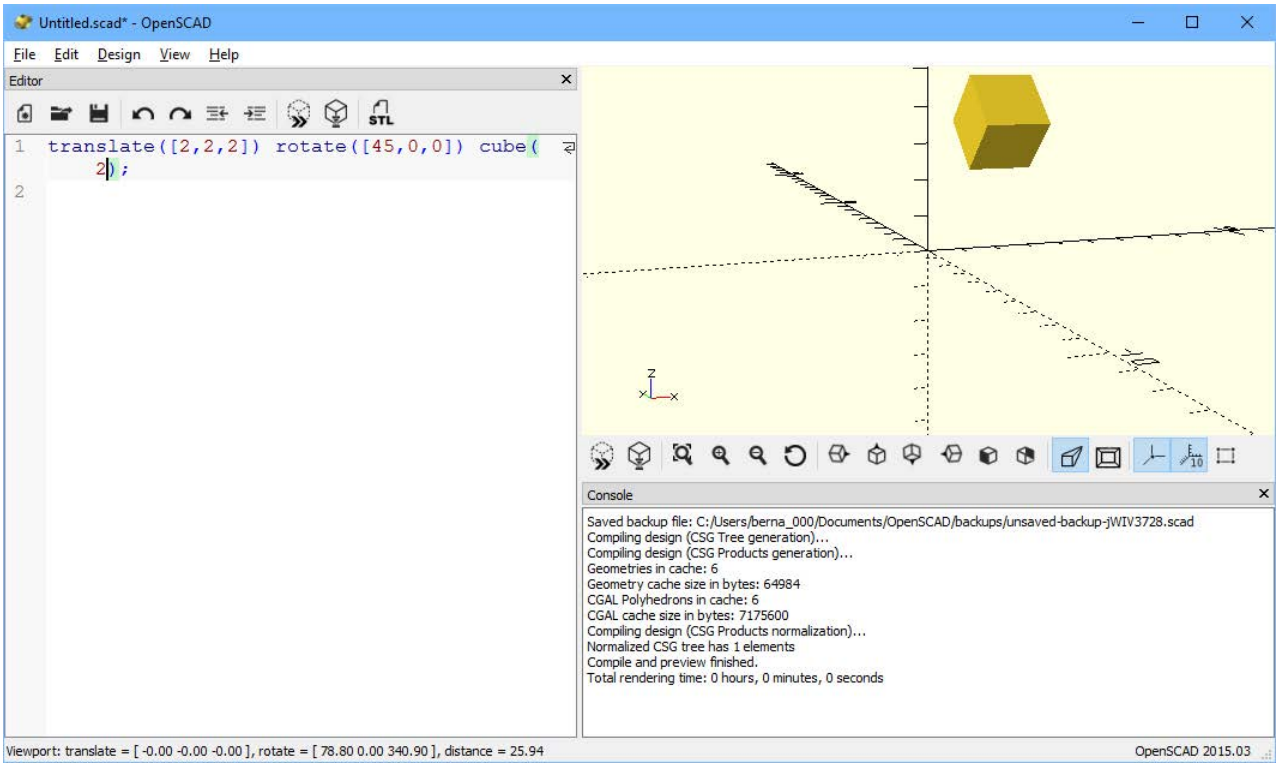


Figure 4. You can apply transformations to your project elements.

up the surfaces of your project directly.

A second way of building 3D objects from 2D elements is through the process of extruding. Extrusion essentially takes a 2D shape and extends it through the third dimension. An example would be getting a cylinder by extruding a circle. This is called linear extrusion. Rotational extrusion takes a 2D object and rotates it around some axis in order to generate a 3D object. Taking the circle example, you could rotate around one of the axes in order to generate a donut shape, or torus. You can apply the same types of transformations that I described for 3D objects too.

Using just the above examples, you already could build rather complex objects, but another class of functions is available that allows you to combine multiple objects in other ways. You can merge multiple overlapping objects together with the union transformation. You can get only the parts that overlap with the intersect transformation. You even can slice away pieces by using the difference transformation to remove any overlapping sections.

There even are program control structures, such as for loops and if-else conditionals. Using all of these available combinations can lead to rather complex behaviors.

Once you have a project properly defined, you can try rendering it fully. There are a few different ways to initiate this. You can click on the Design→Render menu item, or more directly, you can press the F6 key.

The console window will give you diagnostic information about what was done during the rendering, including data like the number of edges, vertices and facets that were used.

If you're happy with the way it turned out, you can export your project a few different ways. You can export an image of your project as either a PNG file or SVG file by clicking File→Export and selecting the file format. You also can export it into one of several different formats used in other CAD systems. One of the formats, STL (STereoLithography), is used in several different systems including 3D printing. Because of its ubiquity, it has earned its own button at the top of the editor pane.

Although OpenSCAD isn't designed to generate the prettiest rendered images, hopefully this article has shown you enough so you can see where it might fit in your workflow. It is a very good program for handling more practical designing of real objects in a simple way. I was able to cover only a small amount of the available functionality, so don't be afraid of digging into the OpenSCAD manual to see all the other things you can do with this software.—Joey Bernard

THEY SAID IT

He who would travel happily must travel light.
—Antoine de Saint-Exupéry

In the absences of a decent time machine, fiction remains the most sturdy vehicle for visiting other eras.
—Tom Nolan

Never help a child with a task at which he feels he can succeed.
—Maria Montessori

If the universe is bigger and stranger than I can imagine, it's best to meet it with an empty bladder.
—John Scalzi

If the wind will not serve, take to the oars.
—Latin Proverb

PREVIOUS
UpFront

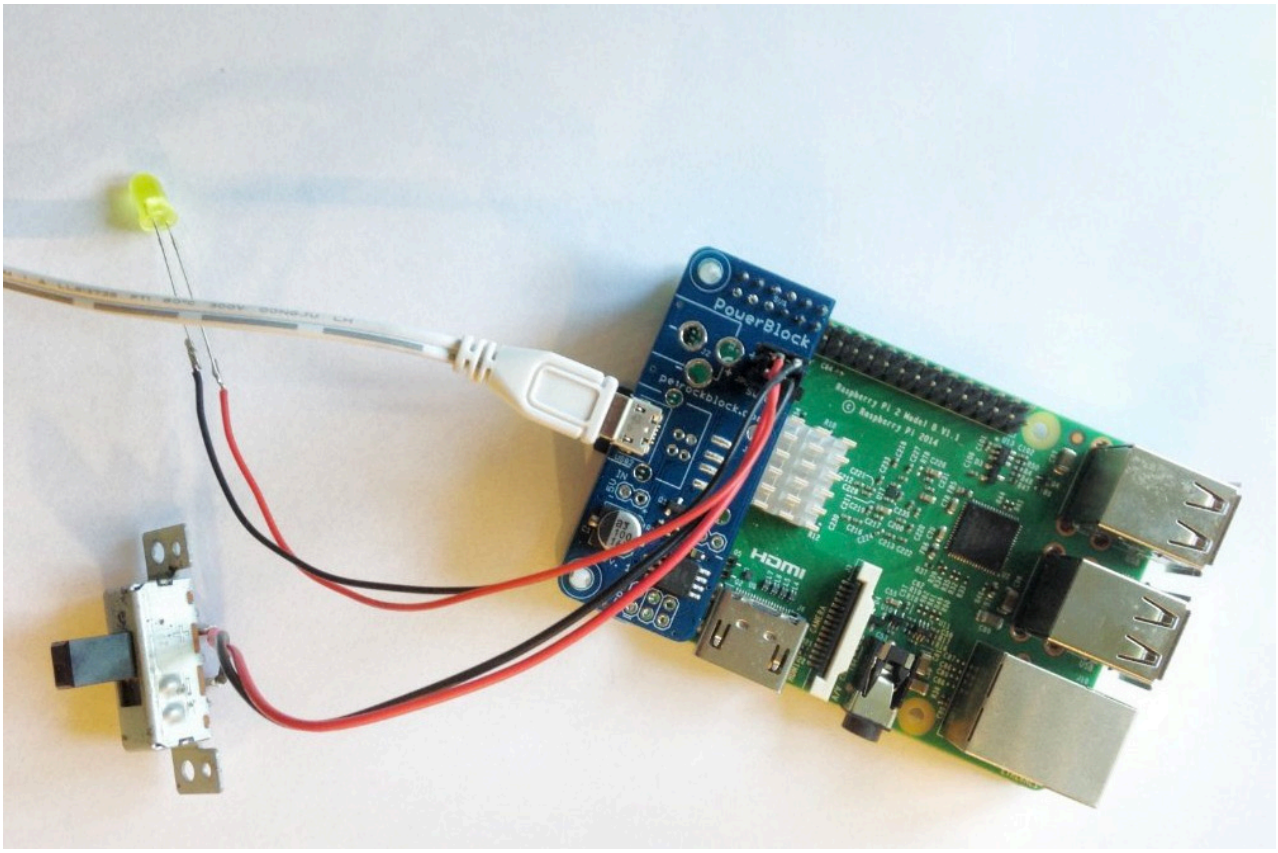
NEXT
Dave Taylor's
Work the Shell

A Switch for Your Pi

In my Open-Source Classroom column this month, I talk about an add-on card for the Raspberry Pi called the ControlBlock. It allows game controllers to be connected as regular joystick devices, but it also has a really incredible power switch feature. The folks at http://blog.petrockblock.com have created an add-on board for the Raspberry Pi that strictly does the power feature for a cheaper price!

The PowerBlock is a tiny, $22 circuit board that provides a few useful features:

- Power can be supplied to the Raspberry Pi via any 5V source, not just MicroUSB. (MicroUSB still is supported though.)
- By connecting a toggle switch, the Raspberry Pi can be turned on. When the switch is turned off, rather than cutting the power to the Raspberry Pi, the PowerBlock initiates the shutdown command, and then powers down the system.
- There are connectors for an LED (either embedded into the switch or separately, like in the photo) that show the progress of the power on/off process.



Basically, when switched on, the PowerBlock starts the Raspberry Pi. The LED blinks slowly until the RPi is completely booted, then the LED stays lit solidly. When the switch is turned to the off position, the LED blinks rapidly while it goes through the shutdown procedure. Then after the RPi is shut down, it powers off the device and the LED.

We typically give the Editors' Choice award to software, but this month, the award goes to the PowerBlock. We love Raspberry Pi projects so much that anything making those projects better deserves our attention. The PowerBlock doesn't do anything the more expensive ControlBlock doesn't already do, but if your project doesn't require the game controller support of the ControlBlock, the PowerBlock is perfect. For more details, head over to http://blog.petrockblock.com.—Shawn Powers

RETURN TO CONTENTS

Let's Go to Mars with Martian Lander

Figuring out all the formulas needed for a simple physics game on Mars.

DAVE TAYLOR

Dave Taylor has been hacking shell scripts on Unix and Linux systems for a really long time. He's the author of *Learning Unix for Mac OS X* and the popular shell scripting book *Wicked Cool Shell Scripts* (new edition coming soon!). He can be found on Twitter as @DaveTaylor, and you can reach him through his tech Q&A site: http://www.AskDaveTaylor.com.

PREVIOUS
Editors' Choice

NEXT
Kyle Rankin's Hack and /

REMEMBER THAT I SAID "LET'S WRITE ANOTHER GAME" IN MY LAST COLUMN? Well, this is the beginning of a series of articles where I develop a variation on the classic lunar-lander game themed around the planet Mars. To do this in three dimensions can be rather complicated, so in the spirit of the original arcade game (that I became rather obsessed with, I should admit), I'm going to tackle the simplified two-dimensional problem. I also am going to discount terrain issues, although clearly landing on the very edge of one of the mysterious Martian canals would be more tricky than a flat plain in the Schiaparelli crater.

Oh also, I'm not going to have any graphics at all. This is going to be a game where you enter thrust commands second by second and either shoot off into orbit and land smoothly on the Martian surface or crash into the planet. You want to add graphics? Excellent. But I'm going to leave that as an exercise for you, the reader, as that's pretty far afield for this shell scripting column.

Gravitational Mathematics

I can't begin Martian lander without talking about physics, because it's Newton's laws (they're not just a good idea!) that describe the process of an object coming into the gravitational field of another and being pulled toward its center.

The Newtonian gravitational formula is *F = G m1 m2 / r2*, and the big idea is that every object in the universe attracts every other object with a force that is proportional to the product of their masses and inversely proportional to the square of the distance between the two objects.

I'm not going to worry about other planets, however, because the gravitational force that far distant objects have on a ship attempting to land on Mars is quite negligible (to say the least). The difference in mass between a planet and a spaceship are enormous too, allowing me to simplify the formula: *velocity = gravity * time*.

If I drop something out of a stationary helicopter (or off the side of a building), in second 0, it'll be falling at 0 ft/s. After one second, it'll be traveling 32 ft/s, and after ten seconds, it'll be going 320 ft/sec. I'm discounting air resistance, and so on, but this example is regarding landing a spaceship on Mars, so there really isn't much atmosphere to worry about here.

The next question is how far has the object fallen in a specified number of seconds? This is a more complex equation: *distance = (gravity * time**2) / 2*.

So after those same ten seconds, the object will have fallen *(32 * (10**2))/2 = 1600 feet*. If I begin the Earthly descent one mile above the surface, that means that without any rockets to slow things down, it'll take just more than 18 seconds to crash.

Mars, however, has a different gravitational force than Earth does. Earth is 32.1 ft/s, while Mars, with only 15% of the mass of our home

planet, has a gravity of 12.1 ft/s.

This means that from a one mile orbital trajectory, it would take a lot longer—almost 30 seconds to crash into the surface of the planet. That's a lot more time to wonder why you forgot to add thrusters to your Martian lander, for sure!

There's a third equation also needed for this game: horizontal velocity. The idea is that the lander will begin in orbit, so it'll have a starting horizontal velocity but no vertical velocity. The rocket boosters will be able to be turned to a specific angle and fired, so with just the right effort, you can stop the horizontal motion entirely, allowing the craft to descend straight onto the planet—a good thing, because hitting the surface with lots of horizontal motion is going to be a crash!

In this case, the formula is really easy, because there's no force "pulling" the craft ever faster around the planet nor any force (such as wind resistance) that's slowing it down either. So if the craft starts with a 100 ft/sec horizontal velocity, it'll land with exactly the same velocity if the thrusters don't slow it down.

For simplicity, the formula I'll use is *speedH = initial speed – thrust*. So to stop all horizontal speed, a burst that exactly matches the current speed is all that's required. I'll assume this all happens essentially instantaneously.

But, the thrusters can operate within a 90° rotation, ranging from straight downward (all vertical thrust, no effect on horizontal speed) to straight forward (all horizontal thrust, no effect on vertical drop). How do you model that? Here are my shots at the formulas, with zero degrees being down and 90 degrees being forward:

- *thrustH = thrust * (angle / 90)*

- *thrustV = thrust * (1 – (angle / 90))*

So a thrust of 100fps at 45° should balance out, and *100*(45/90) = 50fps* and *100 * (1 – (45/90)) = 50*. Good. What about a 10° thrust? ThrustH = 11.11 fps and ThrustV = 88.88 fps.

Building the Program from the Math

Now I have the basic mathematics required, remembering that each second of time adds (gravity) vertical acceleration toward the Martian surface. To make this more fun, I'm going to assume that the one-mile mark is the very edge of the Martian gravity, so if at any point the craft goes farther than a mile from the surface, it's lost in deep space.

You simply could burn the retro rockets at exactly the force of gravity for as long as it takes to land on the surface, but of course, you don't have that much fuel (predictably!). There are constraints to the rocket boosters too: no burst greater than 100fps is allowed, or it'll tear the lander apart, which is definitely not a desired outcome!

That means you can't wait until the last second and slam a *huge* burst of rocket power just before you crash—not to mention that the g-forces would be more than a bit problematical!

Let's start pulling things together. At any given second, you'll have the options of firing the thruster, how much to fire it and at what angle it should be fired. Against that, at any given second, you'll have both horizontal and vertical velocity and vertical gravitational pull. Like this:

- *thrustH = thrust * (angle / 90)*
- *thrustV = thrust * (1 – (angle / 90))*
- *velocityH = velocityH – thrustH*
- *velocityV = (velocityV + gravity) – thrustV*

The initial values are thrust = 0, angle = 0, velocityH = 100 (everything will be in feet per second for simplicity) and velocityV = 0. Gravity on Mars = 12.1.

There's another formula required, and that's height. At any given second then, *height = height – (velocityV)*.

If the craft starts at 5280 feet off the Martian surface, then in

second zero:

- thrustH = 0
- thrustV = 0
- velocityH = 100
- velocityV = 0
- height = 5280

And, in second one, assuming you fire the thrusters at 20fps for one second at 45 degrees:

- *thrustH = 20 * (45/90) = 10*
- *thrustV = 20 * ((1 – (45/90)) = 10*
- *velocityH = (100 – 10) = 90*
- *velocityV = (0 + 12.1 – 10) = 2.1*
- *height = (5280 – 2.1) = 5278.9*

See how that works? It's not too bad once you get through all the basic calculations.

Having gone through all of this physics and mathematics, in my next column, I'll jump into the coding, because it's going to be pretty straightforward. So stay tuned!

Note: thanks to my friend Brad Waller for pointing out all the major oversimplifications in my physics modeling. My defense is that it's just a game, and I'm sticking with that, so take a deep breath, physics nerds.■

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

RETURN TO CONTENTS

Papa's Got a Brand New NAS

Racks of x86 servers are so 2005! Now is the age of low-power ARMs.

KYLE RANKIN

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.

PREVIOUS
Dave Taylor's Work the Shell

NEXT
Shawn Powers' The Open-Source Classroom

IT USED TO BE THAT THE TRUE SIGN YOU WERE DEALING WITH A LINUX GEEK was the pile of computers lying around that person's house. How else could you experiment with networked servers without a mass of computers and networking equipment? If you work as a sysadmin for a large company, sometimes one of the job perks is that you get first dibs on decommissioned equipment. Through the years, I was able to amass quite a home network by combining some things I bought myself with some equipment that was too old for production. A major point of pride in my own home network was the 24U server cabinet in the garage.

It had a gigabit top-of-rack managed switch, a 2U UPS at the bottom, and in the middle was a 1U HP DL-series server with a 1U eSATA disk array attached to it. Above that was a slide-out LCD and keyboard in case I ever needed to work on the server directly.

The 1U server acted as my primary server for just about everything. It was the gateway router, local mail relay and secondary MX for my personal domains, DNS server, DHCP server, and with the eSATA array, it became our home NAS (Network Attached Storage) array that we used for general file storage and backups. Everything generally worked well, and if you ignored the power bill and the space it took up, it was quite the impressive setup in its day.

The key phrase here is "in its day", because these days, a combination of virtualization and cloud computing means you are more likely to see a Linux geek with a laptop than a pile of servers. As computers have become faster, smaller and cheaper, my 1U server was starting to show its age. Given that all of my eggs were in this basket, I started wondering about what I'd do if one of the expensive components on the server failed. Although the server had been stable up to this point, I realized it wouldn't last forever, and if it did break, I could buy modern hardware for the cost of replacing, for instance, one of its fancy serial-attached SCSI drives. I ended up researching a lot of different options, and in this article, I describe how I ended up replacing that 24U cabinet and all the hardware in it with something that's smaller than a shoe box, much lower-powered and relatively cheap.

It's an ARM's World

At the beginning of my search, I started down a more traditional route with a cheap 1U server and a modern motherboard, but I quickly started narrowing down the motherboards to small, lower-power solutions given this machine was going to run all day. As I started considering some of the micro ATX solutions out there, it got me thinking: could I use a Raspberry Pi? After all, the latest iteration of the Raspberry Pi has a reasonably fast processor, a decent amount of RAM, and it's cheap, so even if one by itself wasn't enough to manage all my services, two or three might do the trick and not only be cheaper than a standard motherboard but lower power as well.

Unfortunately when you are talking about a home server, in particular a NAS, even recent Raspberry Pis have some limitations.

If you have been reading my column through the years, you'll know I'm no stranger to solving problems with Raspberry Pis—whether it's controlling the temperature of a beer fridge, creating a gaming media center, flashing coreboot onto an X200 or controlling my 3D printer. Unfortunately when you are talking about a home server, in particular a NAS, even recent Raspberry Pis have some limitations.

The first limitation with a Raspberry Pi isn't the CPU or the RAM, but the network card. A 10/100 network card is fast enough for some services around the house, but if you are setting up a NAS these days, those big media files demand a gigabit network, and the USB2 port on a Raspberry Pi isn't fast enough to drive a USB gigabit NIC.

The second limitation is disk I/O. Even if a Raspberry Pi had a gigabit NIC, your storage options are limited to what you can fit on a microSD card or a hard drive hanging off one of the USB2 ports, and USB2 is just too slow for a modern NAS. If a Raspberry Pi had USB3, you could bypass the network limitations with a USB3 gigabit NIC to it, but as it stands, the I/O is just too slow to replace even an old 1U server that has eSATA disks on a gigabit network.

So a Raspberry Pi was out of the race, but that got me thinking—I knew there were other cheap ARM single-board computers out there that had different hardware options. If I could find one with decent network and storage I/O, maybe it could be a contender.

I've Got That Feeling: Banana Pi

One of the first places I ended up when searching for a Raspberry Pi with faster I/O was the Banana Pi. Despite the similar name, this project is completely unrelated to Raspberry Pi, even though the board is a similar size and price ($35), and it has a dual-core 1GHz ARM Cortex A-7 processor in it with 1GB RAM, so even its specs were

similar to some of the older Raspberry Pi revisions.

The big difference between a Banana Pi and Raspberry Pi though was the fact that it touted both a gigabit network and a SATA2 port! That meant I could hang one of the large hard drives from my old NAS off of a Banana Pi. This got me thinking. My current solution had a number of large hard drives in a software RAID5. While any individual drive wouldn't be large enough for my files, I could buy one of the newer larger drives out there, and that still would be cheaper than some of the traditional solutions.

Of course, a single Banana Pi with a single drive wouldn't solve my fault-tolerance problems. If the drive failed, I would be sunk. Given how cheap the Banana Pis were, I started applying some cloud computing approaches to my home network. Instead of worrying about an individual, potentially unstable server, what if I split the load and fault tolerance across multiple Banana Pis? In a prior article, I walked through creating a GlusterFS cluster on Raspberry Pis, and I realized that Banana Pis would work even better in that case.

I still wasn't sure how well it would work, but I was sure enough that I first bought one Banana Pi to put it through its paces, and when I was reasonably satisfied with its performance, I bought a second one and set them up in a two-node GlusterFS cluster. I've used GlusterFS for a number of years, and what I've learned during those years is that setting up GlusterFS is easy; it's the maintenance—particularly when dealing with faults on an unreliable network—that's hard.

I started to realize that if I really wanted a chance at this cluster being reliable, I would need to add a third Banana Pi to the cluster. Among other things, a third node would help combat split brain—a scenario when each member of a two-node cluster thinks it's the master—and it would help distribute the load. Although a Banana Pi has enough power to run a 2.5" hard drive off the motherboard, the 3.5" hard drives I wanted to use required a separate power supply. As I started adding up all of the Banana Pis, this arrangement was beginning to look kind of clunky, and I started worrying about the overall network load when a new large file was uploaded to the server and it had to be sharded and shared. I started wondering if I was making this a bit too complicated.

I Feel Good: ODROID XU4

It was around this point that I started researching other small ARM computers that might offer more ports or more resources, and I ran across the ODROID XU4. I had been somewhat familiar with the ODROID line of computers in the past, but up to this point, I hadn't really had a need for one.

The ODROID XU4 in particular caught my attention, because although it was between two to three times as much as a Banana Pi (the cheapest I found was $75), it had double the RAM (2GB) and an *eight* core 2GHz ARM processor. Now this is some pretty respectable hardware that had a better chance of handling all of the load from my previous four-core AMD 1U server. It also had a gigabit NIC, and although it didn't have any SATA ports, it did have some USB3 ports, which offer similar bandwidth both to the SATA2 port on the Banana Pi and the eSATA port I was using on my old 1U server.

Since the ODROID XU4 used USB3 instead of SATA, I started pricing out standalone USB3 enclosures for my existing hard drive. While I was looking up 3D-printed cases for the ODROID XU4, I noticed one particular project on Thingiverse where someone had come up with a design for a case that mounted your ODROID XU4 on an existing USB3 drive enclosure. This case was designed for the Mediasonic ProBox HF2-SU3S2, which was a four-drive USB3 and eSATA enclosure that ran about $100—around the amount of money I was going to spend on a few standalone USB3 enclosures for my 3.5" SATA drives.

More important with this enclosure than the price was that I realized since the drives would be presented to the computer individually, even though they were connected through a singular USB3 port, I could maintain the existing software RAID I had in place! No multiple levels of backups and restore or Linux software RAID voodoo to go through—I could just plug in and go like with my 1U eSATA array. That ease of migration pushed me over the edge, so even though I had a couple Banana Pis in the house, I decided to order an ODROID XU4 and the Mediasonic enclosure. I ended up printing out the special Mediasonic ODROID XU4 case while I was waiting on the hardware to arrive.

The Payback

The good thing about most ARM boards these days is they all tend to

The eight-core processor so far has been more than enough for all of the standard tasks I put my home server through, and this machine serves as a DNS server, secondary MX, home NAS and a number of other services without skipping a beat.

provide at least standard Debian images, so once my board arrived, it was simple to set up a Debian server similar to my existing one and port over all of my configuration files. The day I set up the big server move was actually pretty uneventful. Because my RAID configuration file already was copied over to the new server, it was just a matter of moving the drives to the new array and mounting it. The rest of my services worked out of the box after I copied the configuration files over.

The eight-core processor so far has been more than enough for all of the standard tasks I put my home server through, and this machine serves as a DNS server, secondary MX, home NAS and a number of other services without skipping a beat. I'm sure if I did a lot of media transcoding or something I might notice some slowdown with ARM versus a classic Intel processor, but after a few months with this new set up for all of the things I do, it seems more than adequate.

Really the main thing I miss on this new setup is virtualization. I have a particular image gallery I use to share pictures with my family, and it hasn't kept up with the times, so I've found myself having to run it in a virtual machine on an old version of Ubuntu Server. Also, I really wanted to set up a separate backup server instead of running my backups on the same main machine. Of course, I had those Banana Pis just lying around, so I was able to use one for my classic image gallery and attach a SATA drive to the other one and turn it

Figure 1. My New Server Rack

into a nice standalone backup server for my important files.

The real payback on this solution though is in power savings. These ARM processors sip power compared to my old server, and I figure I'll be able to pay for the whole upgrade in power savings alone during the year. Plus, I ended up selling my old server cabinet and freed up a lot of space in my garage. This new solution fits on a shelf or two off in the corner. All in all, it's been nice to get with the times and use new, small, low-power hardware. Plus, I know if I have a hardware problem now, replacement hardware is low-cost and easy to come by.■

RESOURCES

ODROID XU4 Information Page: http://odroid.com/dokuwiki/doku.php?id=en:odroid-xu4

Mediasonic ODROID XU4 3D-Printed Case: https://www.thingiverse.com/thing:1125745

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

RETURN TO CONTENTS

My Childhood in a Cigar Box

What's better than playing Nintendo on a 65" screen? Nothing!

SHAWN POWERS

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

PREVIOUS
Kyle Rankin's Hack and /

NEXT
Under the Sink

I GREW UP IN THE 1980S. That meant we drank far too much Kool-Aid, and on Saturday mornings, we got up early to watch cartoons. It also was the heyday of arcades, but I lived in the ghetto of Detroit and couldn't afford quarters to play games. Plus, there were none anywhere near the neighborhood where I lived. For me, the first real video-game experience was the Atari 2600. I played a lot of *Frogger*, *Pac-Man* and *Yars' Revenge* in middle school. The first system really to impact me, however, was the original Nintendo Entertainment System (NES).

My family moved to northern Michigan when I was in eighth grade, and I worked all summer to save for a used NES from one of the kids who got a brand-new Super Nintendo from his parents. I was a poor nerdy kid who moved in the middle of eighth grade, so my group of friends was fairly small. I had exactly one

friend. There happens to be two controllers with a Nintendo, so it worked out perfectly for Pete and me. While the arcade system I built back in 2007 (my first *Linux Journal* article) may have been to relive the 1980s, this article's project is really a better look at my actual childhood. And this article's project is *awesome*!

The Goal

My end goals for this project are the following:

- Play Nintendo and Super Nintendo games using emulation on a Raspberry Pi.
- Fit the project into a wooden cigar box (because I already have a cool wooden cigar box).
- Use original NES and SNES controllers, not USB knockoffs.
- Boot up, select and play games using nothing more than the controller for navigating menus.
- Plug controllers into emulation machine using either original connectors or RJ-45 plugs.
- Have a good way to turn the machine on and off, not just unplug it.
- Support HDMI, because that's what all televisions and projectors use now.
- Support game state saves and restores. (Yes, it's cheating, but I'm more than 40 years old, so if I want to save myself 40 hours of play every time I get to a boss level, I'm gonna do it!)

Thanks to the size of the Raspberry Pi, it's possible to build a project like this into just about anything. I don't have an NES case anymore, but if I did, I'd probably build it inside one for added nostalgia.

I decided to use RetroPie as the distribution for my project.

Figure 1. This is the completed system. Notice the working power LED and both types of controllers plugged in. The screen shows Emulation Station's front end.

The great thing about using RetroPie is that it basically solves all the issues on my list. It has the "Emulation Station" front end built right in (Figure 1), which supports navigation via controller. It also has emulators already installed, waiting for ROMs to be added. Truly, using RetroPie as my base saved at least one article on software alone!

Also, since I'm using the incredible RetroPie software, I could easily add more platforms to my emulator. If you were a Sega Genesis fan, for instance, you could add those ROMs and get back a slice of your

own childhood. RetroPie supports somewhere around 20 different platforms! I just wanted Nintendo and Super Nintendo, but obviously you can support whatever games you want.

The Really Long List of Parts

First off, it's important to note that I've been building this machine on and off for months. I didn't come up with a list of items all at once; rather, while I was building, I'd realize I needed something and order it. I also decided to do this build "right" versus how most of my projects go. In a huge paradigm shift for me personally, you won't find any duct tape in the box. That said, please don't buy all these parts just because I did. Your build will look different, especially at first. Duct tape is perfect for the trial stage, and often, there's no need to get past the trial stage!

Here's my list:

- Cigar box (Figure 2): I bought this at a sidewalk sale for $2. It didn't contain any cigars, thankfully.

- Raspberry Pi 3: it's taken me so long to complete this project, I had to buy a new RPi twice, because new iterations kept coming out! I started with the original RPi B, then bought version 2, and recently, version 3 with Wi-Fi and Bluetooth.

- ControlBlock from http://blog.petrockblock.com (Figure 3): this is a new device, and it is amazing. You can use the GPIO pins directly if you want, but buying this device makes the project simple and incredibly awesome. It's $39, and worth it.

- Original SNES and NES controllers: I got these from Goodwill and eBay. I have had about a 50% success rate with the eBay controllers. I suspect they're just really worn out. I haven't tried the aftermarket ones on Amazon, but they might work.

- Extension cables for SNES and NES: I used these to make connectors and adapters. They're from Amazon.

- HDMI/USB mountable extension cables (Figure 4): this allowed me to connect HDMI and USB from the outside of the box. I like this particular one because it has a round mounting hole. Round holes are easier to make than square ones!

- Flush-mount power socket: I'm just using 5V for the project, so anything that supplies 5V would work, even a hefty phone charger (2–3amps).

- Soldering equipment: I suck at soldering, so I got a fancy device for holding things while soldering. Be sure to get rosin core solder as well.

- LED switches (toggle), nylon standoffs (for RPi mounting), wires, resistors, shrink tubes for wires, hole saws, hot glue and probably 50 other things I can't remember—also probably duct tape, but I honestly don't think so!

Figure 2. This is the cigar box I used. It was surprisingly sturdy, but obviously not the only option for building something like this.

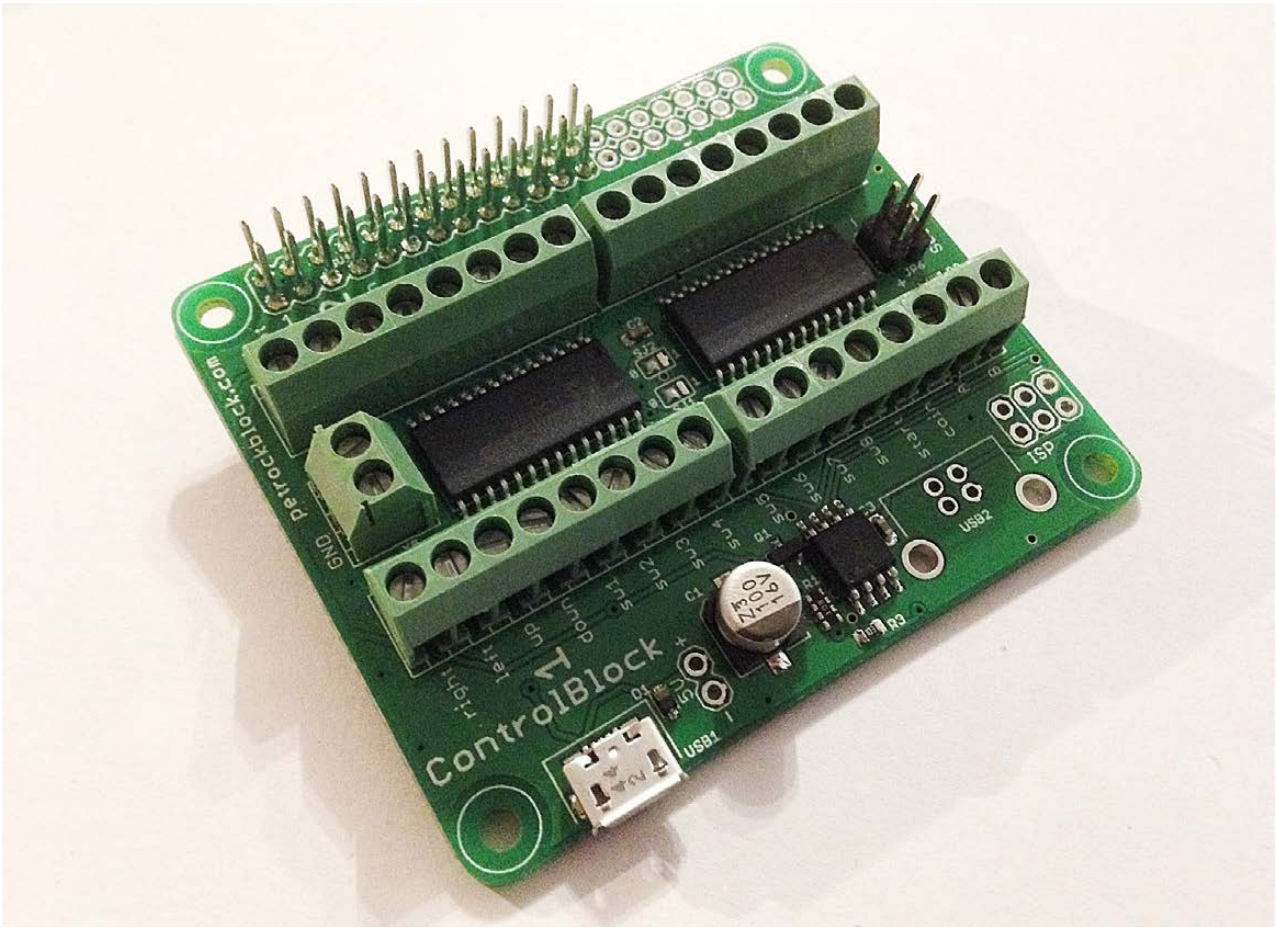


Figure 3. The ControlBlock from petRockBlock is amazing. Truly, it took this project to the next level. I can't recommend it highly enough.

Figure 4. This adapter was from Amazon, and it is my way of cheating. I'm not good at soldering, so prebuilt cables are awesome. Unfortunately, 3-feet long was the shortest option.

The Procedure—Controllers

Wiring the controllers is the most difficult part, so I started there. The cool thing about NES and SNES controllers is that they use the exact same wiring. The SNES just has more buttons. That means you can plug them in to the same wiring harness, and it will work the same. Unfortunately, the NES and SNES have different connectors, even though their pinouts are compatible (Figure 5). That means if I wanted to have swappable controllers, I had to make them both plug in to an identical port. I considered using RJ-45 connectors. I still think it's a pretty good idea, but the thought of cutting all those

The cool thing about NES and SNES controllers is that they use the exact same wiring.

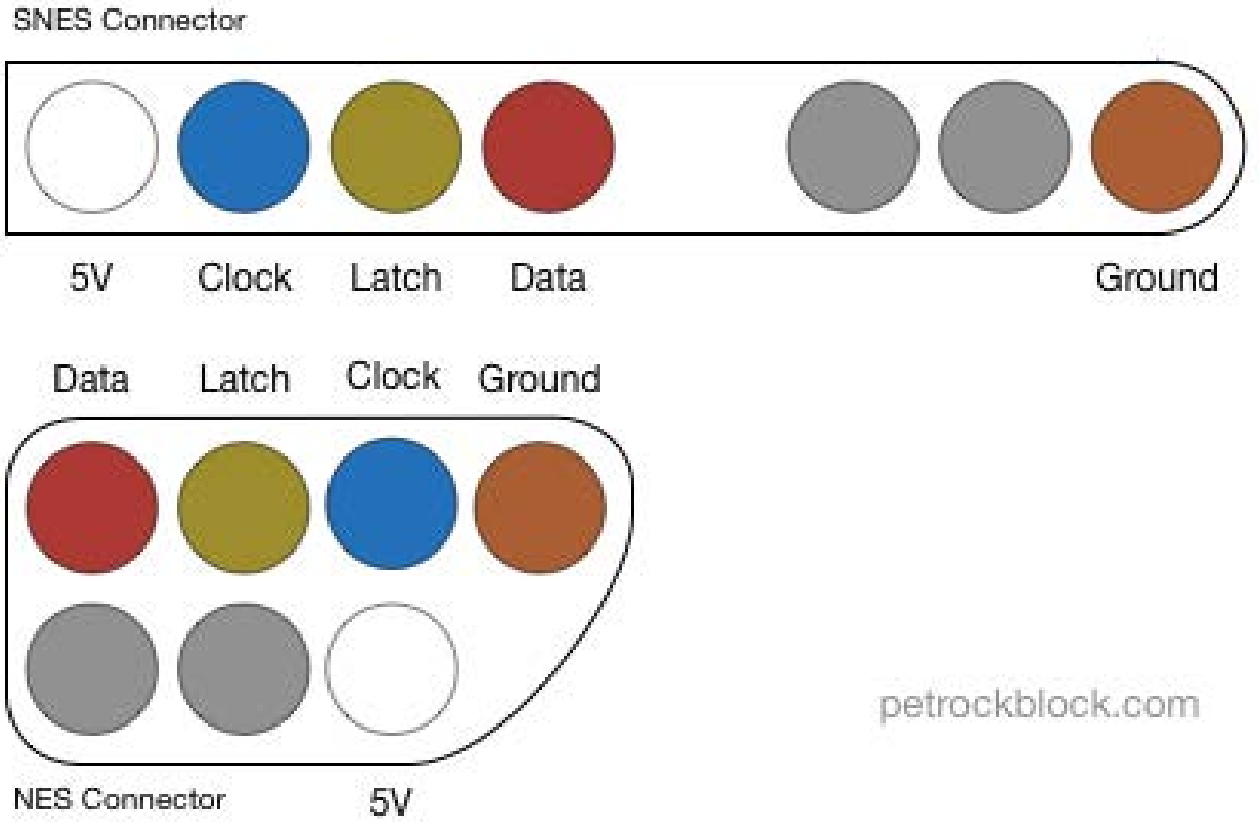


Figure 5. Although shaped differently, the NES and SNES controllers use the exact same wiring. That makes the controllers interchangeable too, which is awesome.

Figure 6. The mounting holes are ugly, but hot glue covers a multitude of sins.

original controllers broke my heart. (I snipped an NES controller and almost cried.) That's when I came up with the idea to use cheap extension cables to make both controllers work. Figure 6 shows the front and back of my cigar box with the SNES extension cable "console side" mounted. That allows me to plug the SNES controller directly in to the box, like an original SNES console.

In order to plug in the NES controllers, I bought an NES extension cable (Figure 7), and I'll use the remaining half of the SNES extension to make an adapter cable. I'll be able to plug the adapter cable in to the cigar box and then the NES controller in to the other end of the adapter cable. I'll never need to snip an actual controller, just the cheap extension cables! (Note: to make the cables, look at the

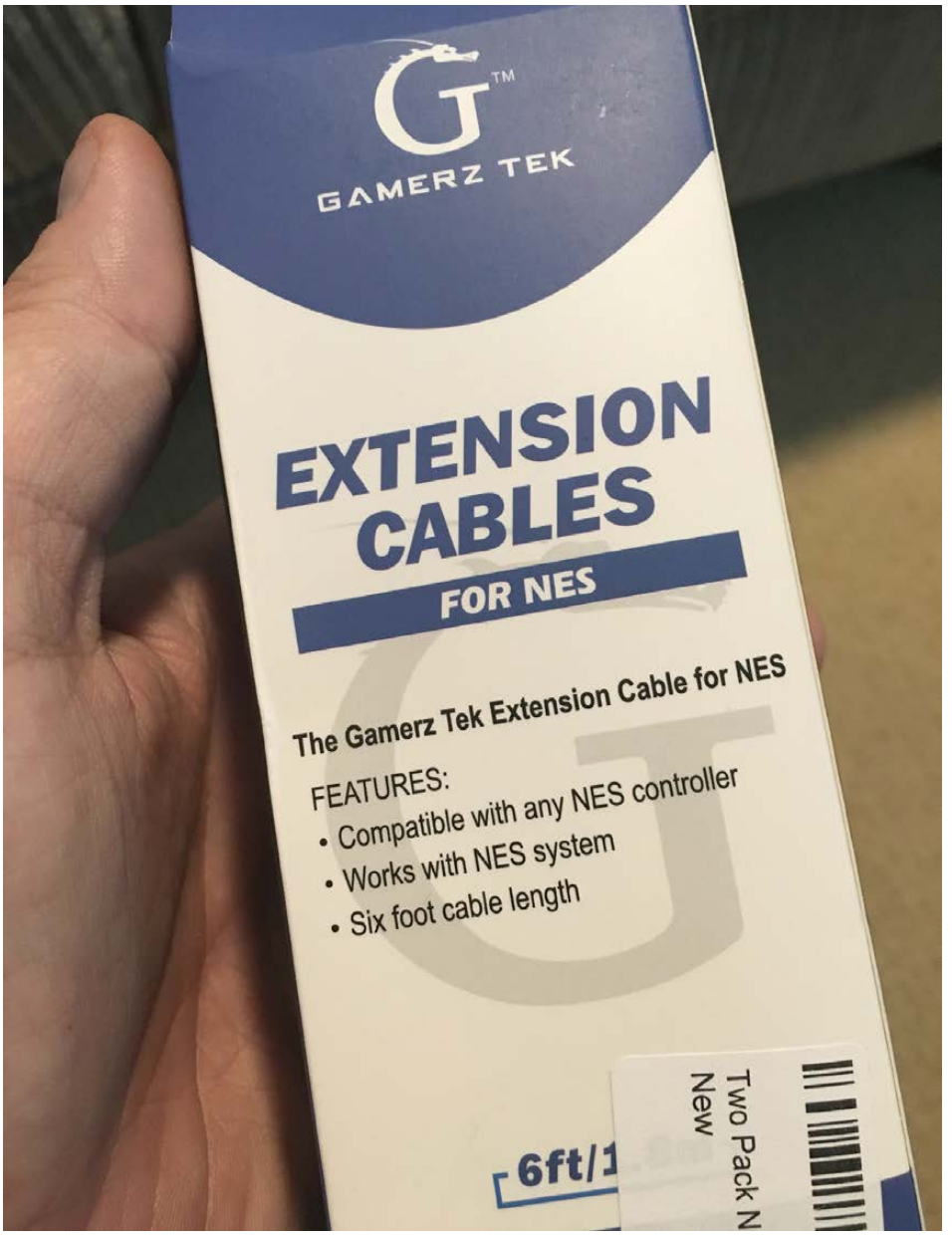


Figure 7. I have no problem cutting up $7 extension cables. I still wish I hadn't cut the end of one of my NES controllers though—that still hurts.

pinouts in Figure 5 and connect the appropriate wires from each extension half together. I recommend soldering those connections and sealing them in shrink tube.)

The Procedure—Raspberry Pi

Normally, I'd just use double-sided sticky tape to attach the RPi. For some reason, I thought using nylon standoffs (from Amazon) would be a good idea. If you look closely (Figure 8), you'll see my first drilled holes didn't work, because the HDMI cable connector was too big. I had to drill new holes that allowed more room for the HDMI

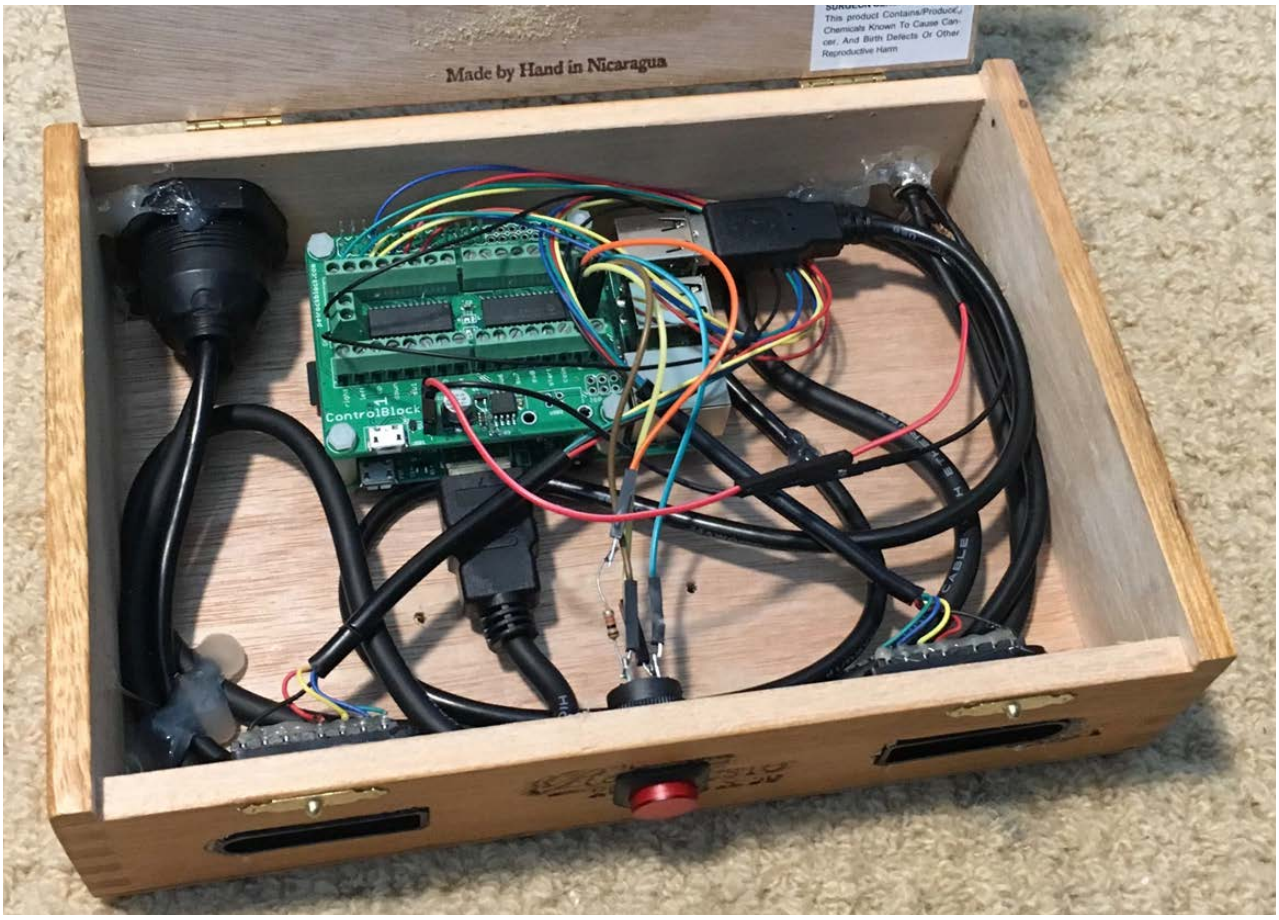


Figure 8. If you look closely you can see my poorly drilled holes. You also can see why I had to move the RPi. The HDMI connector was just too big.

connector and accompanying USB connector. So, I have four extra holes in the bottom of the box. Maybe I'll fill them with wood putty, but probably not.

The ControlBlock fits right onto the Raspberry Pi's GPIO pins. You don't have to use nylon standoffs, but since I had a box of them, I used them to mount the ControlBlock securely on top of the Raspberry Pi. You'll notice the ControlBlock has really nice screw-mounts for wires. That means connecting the controllers doesn't take solder. *Sweet!* The pinout for connecting the SNES (or NES, they're the same) controllers is on the petRockBlock website. (It's shown in Figure 9 as well.) It's important to note that where the controller pinout says 5V, when connecting to the ControlBlock, you use the VCC pins, which are actually 3.3V. (The Raspberry Pi uses 3.3V on its GPIO pins, so the

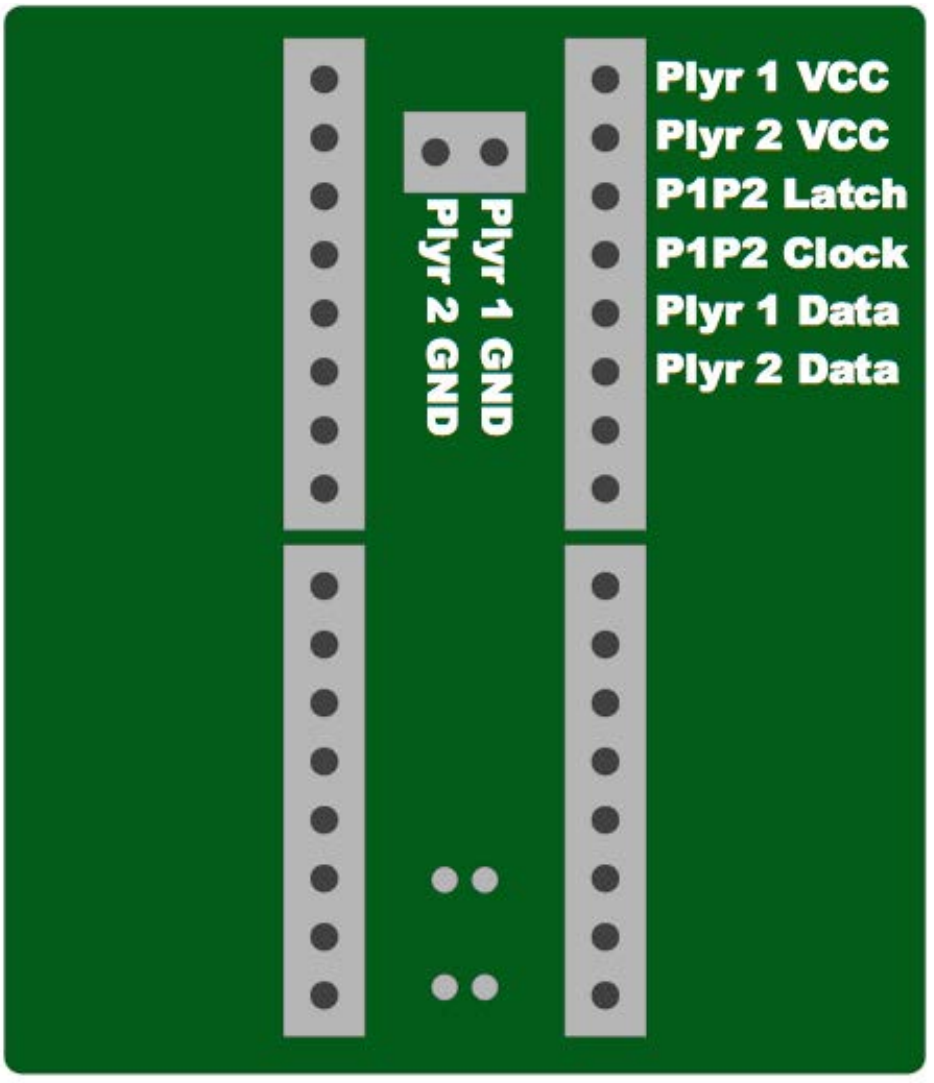


Figure 9. The SNES/NES controllers use only a fraction of the GPIO pins. If you're using the ControlBlock for an arcade setup, it supports far more buttons.

ControlBlock uses that for signaling. Apparently it's enough for the controllers, because it seems to work!)

The other cool feature of the ControlBlock is that it has a really awesome method for powering the Raspberry Pi on and off. You supply power to the ControlBlock (via MicroUSB or via soldered 5V pins, which I did) and then attach a toggle switch. When the toggle switch is "on", the Raspberry Pi boots up. When it's "off", the Raspberry Pi shuts down properly, going through the entire shutdown process. Plus, if you attach an LED, it will flash slowly while booting up and stay solid while the RPi is booted. Then it will flash quickly as it powers down and turn off when the Raspberry Pi turns off. It beats the heck out of just unplugging the unit to turn it off!

The Procedure—Other Stuff

If you look back at Figure 8, you'll see the other things I connected. The power wires are soldered to the ControlBlock, but a MicroUSB power cable could be used instead. I have that HDMI/USB extension mounted to the back of the cigar box (Figure 10), and the wires plugged in to the RPi. The cable was 3-feet long, which was inconvenient, but rather than cut and solder them shorter, I just coiled up the excess and hot-glued it to the case. If I were better at soldering,

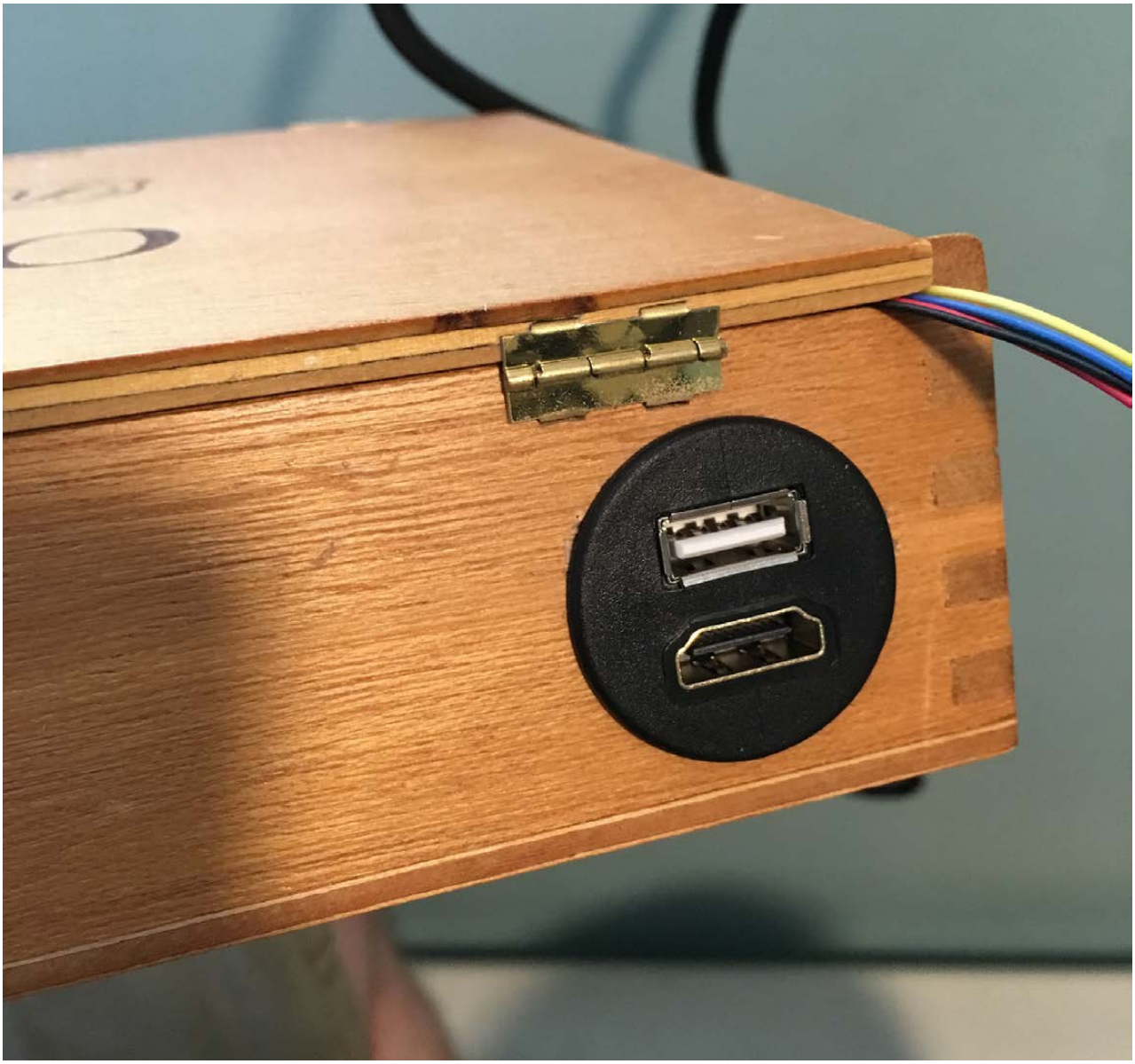


Figure 10. Another cheat: the HDMI/USB flush-mount extension required me to drill a big round hole instead of chipping out a square mounting hole.

I'd get real flush-mount HDMI and USB ports, and just wire them that way. This was a compromise that I'm happy with.

The toggle switch I purchased was from Adafruit. It has an LED built in, so I was able to wire the LED and the toggle switch directly to the ControlBlock. I didn't realize the LED was only a 2 or 3 volt LED, so I burned out several before I realized I had to put a resister in series with the LED. I bought an entire box of resisters, but really needed only a single 1000 Ohm for the project. The mounting for the switch was round, so again, I could just drill a hole to get it mounted. If you look at my SNES controller mounts, you'll see that I don't really have the tools for anything odd-shaped, and if it weren't for copious amounts of hot glue, the SNES controller adapters would never stay! Figure 11 shows most of the junk I ended up purchasing for the build.

Figure 11. This is most of the stuff I gathered to finish this project. That Raspberry Pi is one of the older ones. I'll re-use it for something else!

Software

I mentioned earlier that I used RetroPie as the distribution for this emulation machine. It's based on Raspbian Linux, but it boots directly into Emulation Station, and it includes the binaries for tons of emulators. The only customization you need to do is to download, compile and install the ControlBlock software. Like I said, it's possible to create one of these without the ControlBlock, especially if you're going to use USB controllers. But for me, the $39 was well worth it. Plus, the money goes to a single person who develops them and gives away the software. I bought three ControlBlocks, and I'll be using the others for a new arcade machine. I won't go through the process for setting up the ControlBlock software, but if you go to http://blog.petrockblock.com, you'll find very simple directions that work perfectly with RetroPie. (Just a note: when it comes time to configure the ControlBlock config file, you'll use "SNES" as the controller type, even if you're using NES—remember they're the same as far as the software knows; the NES controllers just have fewer buttons.)

The only other software-related work to do is to install the ROMs. If you have a Raspberry Pi with network capability, it sets up a writable SMB guest share called "ROMS" that you can connect to and upload the ROM files directly. I can't tell you where to find ROM files for console systems, but Google is your friend. Getting those old ROM files can be a legally questionable procedure, but there are some free ones available at the very least.

Once you upload the ROM files into the appropriate folders, when you restart the RPi, the appropriate emulators will appear for you to select games. On the first run, it will have you set up your controllers, but that just requires the controllers to be plugged in; the rest is self-explanatory.

What Now?

Now you play games like it's 1989—or perhaps the early 1990s, depending on your platform of choice. The emulation is amazing, and if you look closely at the configuration options, you'll see there are "shift" keys for the controllers. That means while you're playing, you

can hold down the select key and then press various buttons on the controller to perform system-level actions like resetting the game, or saving and restoring save-game states. It's really like the original consoles, but better. I can't explain how awesome it is to play these old games using the original controllers, but on a huge LCD screen instead of an old 19" television.

My next gaming project will be to create a new arcade system. My original arcade machine didn't survive our recent moves, so I have to (get to?) start from scratch. This time, I'm using a Raspberry Pi and a ControlBlock to emulate those old arcade classics. Until then, I'm quite happy with my cigar box!■

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

RETURN TO CONTENTS

DRUPALCON

DUBLIN

26-30 SEPTEMBER 2016

Great ideas can come from anyone, anywhere, at any time.
They just happen more often at DrupalCon.
Come for the collaboration, tips, and new tools.
Stay for the community.

events.drupal.org/linux

Under the Sink

SNMP is a powerful, nearly ubiquitous tool for network management. Read on to find out how to use it and why it's threatened.

PREVIOUS
Shawn Powers' The Open-Source Classroom

NEXT
New Products

GUEST COLUMNIST

ANDREW KIRCH

Andrew Kirch has more than ten years of experience working as a systems/network administrator, with specializations including DevOps, SNMP and NMS. Andrew is Senior Solutions Architect at GoVanguard, a managed IT and DevOps services firm in New York. Prior to working at GoVanguard, Andrew was the Community Manager at Zenoss. In his spare time, he puts computer crackers in prison, flies airplanes and keeps honeybees. He graduated in the class of 2000 from Howe Military School.

How would you find out how much RAM is free on your Linux desktop? That's a really easy question with a lot of answers—`free`, any of the implementations of `top` and Glances all are valid responses. How would you find out how much RAM is free on 200 Linux instances, which are running on a mixture of real and virtual hardware, in dozens of physical locations spread out around the globe? That's a much bigger problem, and there is a tool to make the job easier. However, the lack of upkeep on the standards and lack of development support for the Linux implementation are resulting in proprietary standards creeping in where there once was a more open standard.

SNMP (Simple Network Management Protocol) was designed in 1990 to read and write structured data on devices attached to a network, such as how much free

RAM there is. Yes, and this is important, the M in SNMP really does stand for "Management", not "Monitoring". Although SNMP is usually used to request operational status information, the SNMP "write" functionality can be used to change the configuration on remote devices. Given the lack of security and authentication in the SNMP protocol, SNMP "write" functionality almost always is disabled on the modern internet, and I will not be discussing it here.

History of SNMP

The original IETF (Internet Engineering Task Force) RFC (Request for Comments) standard for SNMP v1 was published by the IETF in 1990. SNMP v2 was published in 1994–1996 as a series of RFCs and included the first effort to secure SNMP. This effort proved unpopular due to the load it placed on network hardware, which, at the time, had very low performance CPUs. This performance issue exists today and still can cause problems for administrators attempting to secure SNMP. Due to the performance problems, SNMP v2c (SNMP v2 with SNMP v1 communities) became the standard. Concurrently with the release of SNMPv2c, the public began to access the internet, and during the next decade, security would become a serious problem with SNMP since SNMP v2c was entirely unencrypted. SNMPv3 came along in 2003 and added TLS to the previous implementation of SNMP v2c. If all of this seems a bit complicated and unnecessary, it's important to know that many implementations of SNMP still ship with support for SNMP v1, v2c and SNMP v3. This means you're likely to see all of them in the wild.

How Is SNMP Used?

One of the challenges on a modern network is scale, and achieving scale requires managing resources. SNMP provides an agent, which listens for incoming SNMP requests on each host, and a standard communications protocol allowing a central collection system called a Network Management System (NMS) to collect data. NMS is outside the scope of this article, but there are many good open-source NMSes, including Zabbix, OpenNMS, Nagios and Zenoss. The data collected by each NMS is pretty standard, and it includes basic systems information like CPU, memory, network and storage utilization.

SNMP Data Structure

SNMP isn't just an agent, it's also a data structure. Each object in the data structure has an Object IDentifier, or OID. Each OID belongs to an MIB, or Management Information Base. These object identifiers and the hierarchical structure function as a tree. Each sequential number is a branch and has a meaning, and each branch is separated by periods (.), somewhat like an IPv4 address. This means that the meaning of an OID can be decoded very simply.

Given an example OID, `1.3.6.1.2.1.1.1.0`, each number has the following meaning:

- 1 = iso
- 3 = org
- 6 = dod
- 1 = internet
- 2 = IETF Management
- 1.1 = SNMP MIB-2 System
- 0 = sysDescr

From the decoded values, it can be determined that this OID is from the IETF standard MIB (more on MIBs later in the article), and it provides a system description of some sort. Let's look at a real-world example from a CentOS 6 box:

```
1.3.6.1.2.1.1.1.0 = STRING: "Linux foo.example.lan
 ➥2.6.32-573.1.1.v6.i686 #1 SMP Fri Aug 21 14:37:07 MDT 2015 i686"
```

From this description, you can determine that the system this agent is running on is running Linux, 2.6.32, and is 32-bit.

Nearly every OID starts with "1.3.6.1", and the reason for this should

be obvious. The modern public internet originally was created by the United States Department of Defense, and at one time, TCP/IP was called the "DOD Model". Since these values are in every OID, they aren't all that useful for identifying what that OID does, and they generally can be ignored.

After 1.3.6.1, there are more types of OID. If the MIB continues with 1.2, as with the example above, the description of the OID can be found in the standard IETF MIB. If it continues with 1.4, the MIB is "private", and you will need to get the MIB from your hardware vendor. Despite being called "private", these MIBs are almost always available.

What Types of OIDs Are There and How Is Each Used?

There are many different types of OIDs so that SNMP can provide an extensive and extensible variety of information. The example from the previous section, 1.3.6.1.2.1.1.1.0, is a `STRING`. You can tell because SNMP tells you the type of OID when you retrieve it:

```
1.3.6.1.2.1.1.1.0 = STRING: "Linux foo.example.lan
 ➥2.6.32-573.1.1.v6.i686 #1 SMP Fri Aug 21 14:37:07 MDT 2015 i686"
```

Other types of OIDs exist, and each has a use. The following is a list of common types of SNMP OID:

- Integer/Integer32: signed 32-bit integer—these are commonly used for storing values, such as the amount of available memory and the amount of free memory.
- Uinteger32: unsigned 32-bit integer (fairly rare).
- Octet String: this is a short (255-character) length of binary or text data.
- IP Address: this returns an IP address.
- Counter32: this returns a 32-bit counter that counts up, then wraps around to 0 when it reaches 32 bits in length minus 1 (4294967295).

This is important, because gigabit Ethernet can send far more than that many bits in five minutes, which is a common NMS polling period.

- Counter64: this has a maximum value of 64 bits – 1, which allows for higher speed Ethernet traffic counting and counting of other large numbers.

- Object Identifier: this returns a different OID and functions like a `GOTO`, if that data is in another MIB.

- Bit String: this is the type of string above, and it returns text information.

- Gauge32: this goes up and down, but it never exceeds a maximum value.

- TimeTicks: represents an unsigned integer of time since another time (often used for uptime).

What Is an MIB, and Isn't a Name Better Than a Bunch of Numbers Anyway?

Earlier I looked at an OID with the ID `1.3.6.1.2.1.1.1.0`. It's a pain to remember that every single time a system description is required. The good news is that SNMP avoids having to memorize or even deal with long strings of numbers by using Management Information Bases, or MIBs. MIBs decode the OID's purpose for you, so you don't have to remember all the values.

By installing MIBs, the previous difficult-to-read output:

```
1.3.6.1.2.1.1.1.0 = STRING: "Linux foo.example.lan
➥2.6.32-573.1.1.v6.i686 #1 SMP Fri Aug 21 14:37:07 MDT 2015 i686"
```

becomes much easier to read:

```
SNMPv2-MIB::sysDescr.0 = STRING: Linux foo.example.lan
 ➥2.6.32-573.1.1.v6.i686 #1 SMP Fri Aug 21 14:37:07 MDT 2015 i686
```

The quotation marks also disappear. The MIB not only translates the OID, but the value as well. The MIB already knows that that OID is a string, so the quotation marks go away.

How do MIBs know how to do this? MIBs are human-readable plain-text files, often found in /usr/share/snmp/mibs. For `sysDescr`, the SNMP client looks up the value in the SNMPv2 MIBs and is able to learn the type of OID, the purpose of the OID and whether it can be written to (from NET-SNMP's SNMPv2-MIB.txt):

```
sysDescr OBJECT-TYPE
    SYNTAX      DisplayString (SIZE (0..255))
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "A textual description of the entity. This value should
        include the full name and version identification of
        the system's hardware type, software operating-system,
        and networking software."
    ::= { system 1 }
```

How Does SNMP v1/v2c Work in Linux?

Getting started with SNMP v1 and v2c in Linux is quite simple. The information will be transmitted in plain text, including the SNMP "Community", which is sort of like a password. Using your package manager, install net-snmp. Edit /etc/snmp/snmpd.conf, remove everything in the file, add the following lines, then save and exit:

```
rocommunity public
syslocation Somewhere (In the World)
syscontact Overworked Admin <admin@paymemore.com>
```

Restart snmpd, run the following command from the same system, and you'll again see the example OID this article has used since the beginning:

```
[user@foo mibs]$ snmpget -v2c -c public localhost SNMPv2-MIB::sysDescr.0
SNMPv2-MIB::sysDescr.0 = STRING: Linux foo.example.lan
```

```
➥2.6.32-573.1.1.v6.i686 #1 SMP Fri Aug 21 14:37:07 MDT 2015 i686
```

If you don't know the specific OID you're looking for, you can use `snmpwalk`, which will "walk" the entire MIB and print the value for each OID. This tends to produce a *lot* of output, and you can shorten it with `head`:

```
[user@foo mibs]$ snmpwalk -v2c -c public localhost | head
SNMPv2-MIB::sysDescr.0 = STRING: Linux foo.example.lan
 ➥2.6.32-573.1.1.v6.i686 #1 SMP Fri Aug 21 14:37:07 MDT 2015 i686
SNMPv2-MIB::sysObjectID.0 = OID: NET-SNMP-MIB::netSnmpAgentOIDs.10
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (154) 0:00:01.54
SNMPv2-MIB::sysContact.0 = STRING: Overworked Admin <overworked@admin.com>
SNMPv2-MIB::sysName.0 = STRING: foo.example.lan
SNMPv2-MIB::sysLocation.0 = STRING: Somewhere out there
```

As `snmpwalk` runs, `sysDescr.0` shows up again, then another OID called `SysObjectID`, which refers to yet another OID, `NET-SNMP-MIB::netSnmpAgentOIDs.10`. `snmpwalk` will look up that OID and display its type and value before continuing through the rest of the SNMPv2-MIB tree.

A lot of the information that SNMP can provide is very sensitive, and it really shouldn't be transferred over the LAN or, worse, the public internet unencrypted.

How Does SNMPv3 Work in Linux?

SNMPv3 is very complex compared to SNMPv2, and it requires several steps to set up. If you're curious about your Linux router at home, the above SNMPv2 example probably will suffice, but in almost any other environment, SNMPv3 is a must. To set it up, first create a read-only SNMPv3 user name, with a local password that is encrypted with SHA and that uses AES. This is more secure than the default values of MD5 and DES, but it's still far from perfect (both MD5 and DES can be broken trivially):

```
[root@foo mibs] # service snmpd stop
Stopping snmpd:                                            [  OK  ]
```

```
[root@foo mibs] # net-snmp-create-v3-user -ro -A snmpv3authPass
 ➥-a SHA -X userpass -x AES user
adding the following line to /var/lib/net-snmp/snmpd.conf:
    createUser user SHA "snmpv3authPass" AES userpass
adding the following line to /etc/snmp/snmpd.conf:
    rouser user
```

Now edit /etc/snmp/snmpd.conf as root, and comment out the `rocommunity` line you added earlier:

```
#rocommunity public
```

Restart snmpd, and run `snmpwalk` with your new SNMPv3 credentials:

```
[user@foo mibs]$ snmpwalk -u user -A snmpv3authPass -a SHA -X
 ➥userpass -x AES -l authPriv 127.0.0.1 -v3 | head
SNMPv2-MIB::sysDescr.0 = STRING: Linux clearos65.trelane.lan
 ➥2.6.32-573.1.1.v6.i686 #1 SMP Fri Aug 21 14:37:07 MDT 2015 i686
SNMPv2-MIB::sysObjectID.0 = OID: NET-SNMP-MIB::netSnmpAgentOIDs.10
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (12756) 0:02:07.56
SNMPv2-MIB::sysContact.0 = STRING: Overworked Admin
 ➥<overworked@admin.com>
SNMPv2-MIB::sysName.0 = STRING: clearos65.trelane.lan
SNMPv2-MIB::sysLocation.0 = STRING: Somewhere out there
SNMPv2-MIB::sysORLastChange.0 = Timeticks: (42) 0:00:00.42
SNMPv2-MIB::sysORID.1 = OID: SNMP-MPD-MIB::snmpMPDMIBObjects.3.1.1
SNMPv2-MIB::sysORID.2 = OID: SNMP-USER-BASED-SM-MIB::usmMIBCompliance
SNMPv2-MIB::sysORID.3 = OID:
 ➥SNMP-FRAMEWORK-MIB::snmpFrameworkMIBCompliance
```

You'll notice that you still get the same information, but now it's being transferred via user name/password authentication and 128-bit AES. If you try again with SNMPv2, you'll get a timeout now:

```
[user@foo mibs]$snmpwalk -v2c -c public localhost
Timeout: No Response from localhost
```

This is the most secure SNMP agent configuration currently possible. To increase security, the SNMP port should be firewalled to accept only connections from your NMS.

What Is Happening to SNMP Due to the Lack of Implemented Updates to the Standard Since 2004? Despite being widely adopted, important and incredibly flexible, SNMP is falling by the wayside. SNMPS, SNMP datagrams over TLS, standardized in 2010, has gone mostly unimplemented. SNMPv3 is difficult to use and troubleshoot on devices other than Linux. Microsoft has dropped SNMP support entirely from Windows, replacing it with WMI and then WinRM. Other vendors, and products providing monitoring interfaces, are using an often proprietary API over HTTPS or, worse, unencrypted HTTP that listens to and replies in JSON (Javascript Object Notation) or XML. This balkanization from a single standard has made cohesive monitoring of large networks containing diverse devices more difficult and time consuming.

Linux's NET-SNMP is in even worse shape. With only two contributors since January 1, 2015 (both from VMware) and one project manager, there have been fewer than 30 commits since January 1, 2015. The last stable release of NET-SNMP was in 2014. NET-SNMP has not implemented SNMPS, and there are no apparent plans to do so, leaving that standard dead in the water.

Returning a Value It's unfortunate to see a standard balkanized into hundreds of different proprietary implementations. This wastes time and money and causes aggravation for systems administrators. SNMP and the various standards-compliant implementations of SNMP are still relevant and are in production nearly everywhere. That ubiquity and relevance is on the brink of changing as replacements utilizing proprietary data structures in JSON and XML instead of MIBs and OIDs begin to take over. NET-SNMP and the SNMP standard itself seem to be solid candidates for a rescue. Expanding the IETF MIBs to support newer networked devices, like network attached storage, storage area networks, software-defined networking, containers, cloud, converged and hyper-converged infrastructure will be a must if SNMP is to continue to be relevant.■

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

RETURN TO CONTENTS

Kiwi 2016

PyCon

Kiwi PyCon is a community-focussed conference for the New Zealand python community. A three-day conference of talks, tutorials, and many other activities.

nzpug.org

Location: Dunedin, New Zealand
Conference: 9/10/11 September
Sprints: 12/13 September

Proposals for talks,
tutorials, and posters:
nzpug.org/call-for-proposals

PREVIOUS
Under the Sink

NEXT
Feature: Understanding Firewalld in Multi-Zone Configurations

Ascensio System SIA's ONLYOFFICE

Ascensio System SIA boasts that its ONLYOFFICE office and productivity suite combines the best from the MS Office and Google Docs worlds. ONLYOFFICE is a free and open-source solution and is distributed under the AGPL v.3 license. Ascensio says that its solution trumps Google Docs' collaborative capabilities, allowing users to choose how to co-edit documents—for example, "Fast" (like in Google Docs) or "Strict" (when the changes appear after saving). ONLYOFFICE also out-features MS Office Online, asserts Ascensio, allowing its users to work with auto-shapes, -formulas and -charts online. Regarding file formats, Ascensio claims better support for MS Office formats than any other open-source office suite, and it is fully compatible with OpenDocument formats as well. The recently updated ONLYOFFICE 8.9 features the updated collaboration system called Community Server, which includes mail and calendar integration and mail autoreply. Meanwhile, the updated document editors, aka the Document Server, now offer fast real-time co-editing, commenting and integrated chat, reviewing and tracking changes, and version history.
http://onlyoffice.com

CodeLathe FileCloud Google Chrome Extension

Nearly everyone in today's enterprises is connected throughout the day to a web browser, of which anywhere from 44–71% are Google Chrome. Seeking to make this vast number of users' work more productive is developer CodeLathe, whose new "amazingly easy-to-use" FileCloud extension for Google Chrome enables users to save documents, images and screen captures directly from Chrome to CodeLathe's FileCloud private cloud file-sharing solution. When using the FileCloud for Chrome extension, FileCloud users now can save and share information easily while working within Chrome by selecting content, opening the right-click context menu in Chrome and then selecting the "Save to FileCloud" option. The FileCloud Chrome extension is particularly useful for teams who work closely together in areas like product and market research, web design and others. FileCloud client apps are available for common desktop and mobile platforms including Linux, Mac, iOS, Android, Windows, Windows Phone 8 and now Chrome.
http://getfilecloud.com

Naztech's Roadstar 5 Car Charger

The "5" in Naztech's new Roadstar 5 Car Charger refers to the abundant five ports offered by the device, intended to end in-vehicle debates on who gets to charge their device next. Naztech says that its new charger delivers superior charging power and speed while protecting tablet and smartphone batteries and motherboards. The Roadstar 5 features a compact design with five illuminated USB ports integrated on two units. The main unit that plugs in to the vehicle's electrical outlet offers two ports with the remaining three found on an extendible hub. Connected by a six-foot cable, the hub clips onto the backseat or center console, allowing users to access devices conveniently from anywhere in the car while charging. The Roadstar 5 integrates Naztech's IntelliQ Technology that enables smart communication between the charger and the attached devices, resulting in the optimal and safest delivery of power and current level, as well as short-circuit and overcharge protection.
http://naztech.com

Synopsys' Coverity

The new version 8.5 of Synopsys' Coverity extends the security umbrella of the static analysis tool to mitigate a wider range of security vulnerabilities. Coverity, a core component of Synopsys' Software Integrity Platform, is an automated software testing tool that analyzes source code to detect critical security vulnerabilities and defects early in the software development lifecycle. Coverity 8.5 adds static analysis capabilities for Ruby and node.js web applications, as well as Android mobile applications. In addition, version 8.5 expands security analysis to address a wider range of security vulnerabilities and adds complete support for MISRA C 2012 coding guidelines used in medical device, automotive and other safety-critical industries. This version of Coverity is ISO 26262-certified, demonstrating Synopsys' efforts to address vehicle security and safety in the midst of emerging industry trends, such as connected cars and autonomous driving. To support its growing customer base and expand its software integrity business in the Asia Pacific region, Synopsys now offers a localized version of Coverity 8.5 in simplified Chinese, including a localized user interface, reporting, IDE plugins and documentation.
http://synopsys.com

Nativ Disc

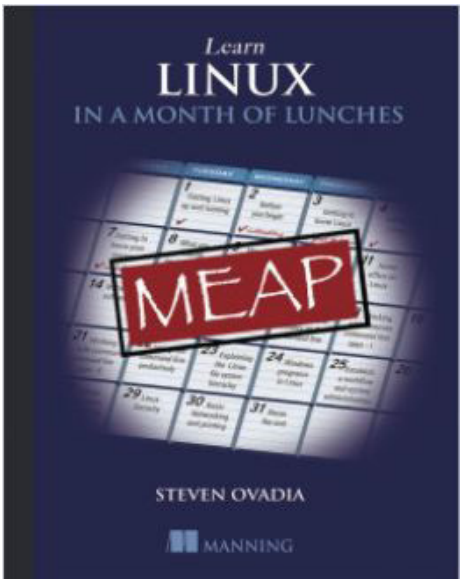
Although most music lovers stream or download music today, the stubborn pre-millennials among us have legacy CD collections at home. This demographic is the perfect target group for Nativ Disc, a bit-perfect CD Ripper that allows users to import up to 12,000 CDs—in lossless FLAC, uncompressed WAV or lossy MP3 format—into their Nativ Vita high-resolution music player. Nativ Disc and Nativ Vita are produced by Nativ, a self-described "nimble and innovative tech startup" that designs audiophile-level components with the latest and greatest in technology by leveraging the power of the crowd through an open platform. To make Nativ Disc the best it can be, Nativ partnered with music-database specialist Gracenote to deliver a more immersive experience and help users re-discover music like never before.
http://nativsound.com

Epiq Solutions' Sidekiq M.2

Following on its resounding success with its Sidekiq MiniPCIe card, wireless communications systems specialist Epiq Solutions recently added the Sidekiq M.2 state-of-the-art, small form-factor, software-defined radio (SDR) card. Epiq Solutions explains that the Sidekiq product line provides a breakthrough small form-factor SDR transceiver solution ready for integration into systems that support either MiniPCIe or now the M.2 card form factors. Compared to the Sidekiq MiniPCIe card, this next-generation product provides benefits such as a 20% size reduction, double the data throughput with its Gen2 PCIe interface, full 2x2 MIMO RF interface and increased FPGA resources with a Xilinx Artix-7 FPGA. Other features of the Sidekiq M.2 card include RF tuning range of 70MHz to 6GHz, up to 50MHz RF bandwidth per channel, flexible RF front end supporting two operating modes, 2.1 W typical power consumption and PDK including software API and FPGA source code.
http://epiqsolutions.com/sidekiq

Senet IoT Foundry

Startup companies and even large enterprises may not be able to harness the full range of skills required to deliver vertically complete IoT solutions to their customers. To assist these companies in getting to market and solving their customers' problems, Senet introduces Senet IoT Foundry, a suite of development services—training, development tools, a network sandbox and technical consulting services—that help IoT solution developers create and launch LoRa-compliant IoT products and applications. Senet calls itself the first and only North American provider of public, LoRa-based, low-power, wide-area networks (LPWANs) for Internet of Things (IoT) applications, putting it in a strong position to support companies in their commercialization of LoRa-based LPWAN products and solutions. Senet is a contributing member of the LoRa Alliance and was the first in North America to gain FCC certification on LoRa-based sensors and gateways. As a result, the company claims to possess a treasure chest of high-level designs, best practices and development tools that can benefit other ecosystem partners. Users of Senet IoT Foundry services can opt to follow the Foundry's four-step development program, or pick and choose the services they need most.
http://senetco.com



Steven Ovadia's *Learn Linux in a Month of Lunches* (Manning Publications Co.)

Yes, Steven Ovadia's new book for Linux "noobs" is titled *Learn Linux in a Month of Lunches*, but readers may need two-hour lunches and weekends to attain the ambitious goal implied in the title. No matter though, because this "study while dining" series of books from Manning Publications offers a fine approach to learning the essentials of our beloved OS, from installation to networking, installing software and securing a system. Readers just curious about Linux or needing to get up and running for their jobs will appreciate how this book concentrates on need-to-know tasks. By digesting targeted, easy-to-follow, compact lessons, readers learn how to use the command line, customize a desktop, print, choose the right application for their needs and more. Readers who make it to the end of the book are treated to topics like filesystems, GitLab and using Linux professionally—for example, certifications. Although new Linux users may be overwhelmed at first, Ovadia's book illustrates how learning Linux doesn't have to be hard, and the payoff is great.
http://manning.com

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

RETURN TO CONTENTS

Understanding Firewalld in Multi-Zone Configurations

Firewalls are essential for system security, but they can be overwhelmingly complex. Firewalld employs the concept of "zones" to organize traffic, greatly simplifying the firewall design process.

NATHAN R. VANCE and **WILLIAM F. POLIK**

PREVIOUS
New Products

NEXT
Feature: Hard Drive Rescue with a Raspberry Pi and Relay

Stories of compromised servers and data theft fill today's news. It isn't difficult for someone who has read an informative blog post to access a system via a misconfigured service, take advantage of a recently exposed vulnerability or gain control using a stolen password. Any of the many internet services found on a typical Linux server could harbor a vulnerability that grants unauthorized access to the system.

Since it's an impossible task to harden a system at the application level against every possible threat, firewalls provide security by limiting access to a system. Firewalls filter incoming packets based on their IP of origin, their destination port and their protocol. This way, only a few IP/port/protocol combinations interact with the system, and the rest do not.

Linux firewalls are handled by netfilter, which is a kernel-level framework. For more than a decade, iptables has provided the userland abstraction layer for netfilter. iptables subjects packets to a gauntlet of rules, and if the IP/port/protocol combination of the rule matches the packet, the rule is applied causing the packet to be accepted, rejected or dropped.

Firewalld is a newer userland abstraction layer for netfilter. Unfortunately, its power and flexibility are underappreciated due to a lack of documentation describing multi-zoned configurations. This article provides examples to remedy this situation.

Firewalld Design Goals

The designers of firewalld realized that most iptables usage cases involve only a few unique IP sources, for each of which a whitelist of services is allowed and the rest are denied. To take advantage of this pattern, firewalld categorizes incoming traffic into zones defined by the source IP and/or network interface. Each zone has its own configuration to accept or deny packets based on specified criteria.

Another improvement over iptables is a simplified syntax. Firewalld makes it easier to specify services by using the name of the service rather than its port(s) and protocol(s)—for example, samba rather than UDP ports 137 and 138 and TCP ports 139 and 445. It further simplifies syntax by removing the dependence on the order of statements as was the case for iptables.

Finally, firewalld enables the interactive modification of netfilter, allowing a change in the firewall to occur independently of the permanent

configuration stored in XML. Thus, the following is a temporary modification that will be overwritten by the next reload:

```
# firewall-cmd <some modification>
```

And, the following is a permanent change that persists across reboots:

```
# firewall-cmd --permanent <some modification>
# firewall-cmd --reload
```

Zones

The top layer of organization in firewalld is zones. A packet is part of a zone if it matches that zone's associated network interface or IP/mask source. Several predefined zones are available:

```
# firewall-cmd --get-zones
block dmz drop external home internal public trusted work
```

An active zone is any zone that is configured with an interface and/or a source. To list active zones:

```
# firewall-cmd --get-active-zones
public
  interfaces: eno1 eno2
```

Interfaces are the system's names for hardware and virtual network adapters, as you can see in the above example. All active interfaces will be assigned to zones, either to the default zone or to a user-specified one. However, an interface cannot be assigned to more than one zone.

In its default configuration, firewalld pairs all interfaces with the public zone and doesn't set up sources for any zones. As a result, public is the only active zone.

Sources are incoming IP address ranges, which also can be assigned to zones. A source (or overlapping sources) cannot be assigned to multiple zones. Doing so results in undefined behavior, as it would not be clear which rules should be applied to that source.

Since specifying a source is not required, for every packet there will be a zone with a matching interface, but there won't necessarily be a zone with a matching source.

Since specifying a source is not required, for every packet there will be a zone with a matching interface, but there won't necessarily be a zone with a matching source. This indicates some form of precedence with priority going to the more specific source zones, but more on that later. First, let's inspect how the public zone is configured:

```
# firewall-cmd --zone=public --list-all
public (default, active)
  interfaces: eno1 eno2
  sources:
  services: dhcpv6-client ssh
  ports:
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:
# firewall-cmd --permanent --zone=public --get-target
default
```

Going line by line through the output:

- `public (default, active)` indicates that the public zone is the default zone (interfaces default to it when they come up), and it is active because it has at least one interface or source associated with it.
- `interfaces: eno1 eno2` lists the interfaces associated with the zone.
- `sources:` lists the sources for the zone. There aren't any now, but if

there were, they would be of the form xxx.xxx.xxx.xxx/xx.

- `services: dhcpv6-client ssh` lists the services allowed through the firewall. You can get an exhaustive list of firewalld's defined services by executing `firewall-cmd --get-services`.
- `ports:` lists port destinations allowed through the firewall. This is useful if you need to allow a service that isn't defined in firewalld.
- `masquerade: no` indicates that IP masquerading is disabled for this zone. If enabled, this would allow IP forwarding, with your computer acting as a router.
- `forward-ports:` lists ports that are forwarded.
- `icmp-blocks:` a blacklist of blocked icmp traffic.
- `rich rules:` advanced configurations, processed first in a zone.
- `default` is the target of the zone, which determines the action taken on a packet that matches the zone yet isn't explicitly handled by one of the above settings.

A Simple Single-Zoned Example

Say you just want to lock down your firewall. Simply remove the services currently allowed by the public zone and reload:

```
# firewall-cmd --permanent --zone=public --remove-service=dhcpv6-client
# firewall-cmd --permanent --zone=public --remove-service=ssh
# firewall-cmd --reload
```

These commands result in the following firewall:

```
# firewall-cmd --zone=public --list-all
public (default, active)
  interfaces: eno1 eno2
```

```
  sources:
  services:
  ports:
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:
# firewall-cmd --permanent --zone=public --get-target
default
```

In the spirit of keeping security as tight as possible, if a situation arises where you need to open a temporary hole in your firewall (perhaps for ssh), you can add the service to just the current session (omit `--permanent`) and instruct firewalld to revert the modification after a specified amount of time:

```
# firewall-cmd --zone=public --add-service=ssh --timeout=5m
```

The timeout option takes time values in seconds (s), minutes (m) or hours (h).

Targets

When a zone processes a packet due to its source or interface, but there is no rule that explicitly handles the packet, the target of the zone determines the behavior:

- `ACCEPT`: accept the packet.
- `%%REJECT%%`: reject the packet, returning a reject reply.
- `DROP`: drop the packet, returning no reply.
- `default`: don't do anything. The zone washes its hands of the problem, and kicks it "upstairs".

There was a bug present in firewalld 0.3.9 (fixed in 0.3.10) for source zones with targets other than `default` in which the target was applied

Therefore, the general design pattern for multi-zoned firewalld configurations is to create a privileged source zone to allow specific IP's elevated access to system services and a restrictive interface zone to limit the access of everyone else.

regardless of allowed services. For example, a source zone with the target `DROP` would drop all packets, even if they were whitelisted. Unfortunately, this version of firewalld was packaged for RHEL7 and its derivatives, causing it to be a fairly common bug. The examples in this article avoid situations that would manifest this behavior.

Precedence

Active zones fulfill two different roles. Zones with associated interface(s) act as interface zones, and zones with associated source(s) act as source zones (a zone could fulfill both roles). Firewalld handles a packet in the following order:

1. The corresponding source zone. Zero or one such zones may exist. If the source zone deals with the packet because the packet satisfies a rich rule, the service is whitelisted, or the target is not default, we end here. Otherwise, we pass the packet on.

2. The corresponding interface zone. Exactly one such zone will always exist. If the interface zone deals with the packet, we end here. Otherwise, we pass the packet on.

3. The firewalld default action. Accept icmp packets and reject everything else.

The take-away message is that source zones have precedence over interface zones. Therefore, the general design pattern for multi-zoned firewalld configurations is to create a privileged source zone to allow

specific IPs elevated access to system services and a restrictive interface zone to limit the access of everyone else.

A Simple Multi-Zoned Example

To demonstrate precedence, let's swap ssh for http in the public zone and set up the default internal zone for our favorite IP address, 1.1.1.1. The following commands accomplish this task:

```
# firewall-cmd --permanent --zone=public --remove-service=ssh
# firewall-cmd --permanent --zone=public --add-service=http
# firewall-cmd --permanent --zone=internal --add-source=1.1.1.1
# firewall-cmd --reload
```

which results in the following configuration:

```
# firewall-cmd --zone=public --list-all
public (default, active)
  interfaces: eno1 eno2
  sources:
  services: dhcpv6-client http
  ports:
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:
# firewall-cmd --permanent --zone=public --get-target
default
# firewall-cmd --zone=internal --list-all
internal (active)
  interfaces:
  sources: 1.1.1.1
  services: dhcpv6-client mdns samba-client ssh
  ports:
  masquerade: no
  forward-ports:
  icmp-blocks:
```

```
  rich rules:
# firewall-cmd --permanent --zone=internal --get-target
default
```

With the above configuration, if someone attempts to `ssh` in from 1.1.1.1, the request would succeed because the source zone (internal) is applied first, and it allows ssh access.

If someone attempts to `ssh` from somewhere else, say 2.2.2.2, there wouldn't be a source zone, because no zones match that source. Therefore, the request would pass directly to the interface zone (public), which does not explicitly handle ssh. Since public's target is `default`, the request passes to the firewalld default action, which is to reject it.

What if 1.1.1.1 attempts http access? The source zone (internal) doesn't allow it, but the target is `default`, so the request passes to the interface zone (public), which grants access.

Now let's suppose someone from 3.3.3.3 is trolling your website. To restrict access for that IP, simply add it to the preconfigured drop zone, aptly named because it drops all connections:

```
# firewall-cmd --permanent --zone=drop --add-source=3.3.3.3
# firewall-cmd --reload
```

The next time 3.3.3.3 attempts to access your website, firewalld will send the request first to the source zone (drop). Since the target is `DROP`, the request will be denied and won't make it to the interface zone (public) to be accepted.

A Practical Multi-Zoned Example

Suppose you are setting up a firewall for a server at your organization. You want the entire world to have http and https access, your organization (1.1.0.0/16) and workgroup (1.1.1.0/8) to have ssh access, and your workgroup to have samba access. Using zones in firewalld, you can set up this configuration in an intuitive manner.

Given the naming, it seems logical to commandeer the public zone for your world-wide purposes and the internal zone for local use. Start by replacing the dhcpv6-client and ssh services in the public zone

It is more secure to exhibit the behavior of an inactive IP and instead drop the connection.

with http and https:

```
# firewall-cmd --permanent --zone=public --remove-service=dhcpv6-client
# firewall-cmd --permanent --zone=public --remove-service=ssh
# firewall-cmd --permanent --zone=public --add-service=http
# firewall-cmd --permanent --zone=public --add-service=https
```

Then trim mdns, samba-client and dhcpv6-client out of the internal zone (leaving only ssh) and add your organization as the source:

```
# firewall-cmd --permanent --zone=internal --remove-service=mdns
# firewall-cmd --permanent --zone=internal --remove-service=samba-client
# firewall-cmd --permanent --zone=internal --remove-service=dhcpv6-client
# firewall-cmd --permanent --zone=internal --add-source=1.1.0.0/16
```

To accommodate your elevated workgroup samba privileges, add a rich rule:

```
# firewall-cmd --permanent --zone=internal --add-rich-rule='rule
 ➥family=ipv4 source address="1.1.1.0/8" service name="samba"
 ➥accept'
```

Finally, reload, pulling the changes into the active session:

```
# firewall-cmd --reload
```

Only a few more details remain. Attempting to `ssh` in to your server from an IP outside the internal zone results in a reject message, which is the firewalld default. It is more secure to exhibit the behavior of an inactive IP and instead drop the connection. Change the public

zone's target to `DROP` rather than `default` to accomplish this:

```
# firewall-cmd --permanent --zone=public --set-target=DROP
# firewall-cmd --reload
```

But wait, you no longer can ping, even from the internal zone! And icmp (the protocol ping goes over) isn't on the list of services that firewalld can whitelist. That's because icmp is an IP layer 3 protocol and has no concept of a port, unlike services that are tied to ports. Before setting the public zone to `DROP`, pinging could pass through the firewall because both of your `default` targets passed it on to the firewalld default, which allowed it. Now it's dropped.

To restore pinging to the internal network, use a rich rule:

```
# firewall-cmd --permanent --zone=internal --add-rich-rule='rule
 ➥protocol value="icmp" accept'
# firewall-cmd --reload
```

In summary, here's the configuration for the two active zones:

```
# firewall-cmd --zone=public --list-all
public (default, active)
  interfaces: eno1 eno2
  sources:
  services: http https
  ports:
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:
# firewall-cmd --permanent --zone=public --get-target
DROP
# firewall-cmd --zone=internal --list-all
internal (active)
  interfaces:
  sources: 1.1.0.0/16
```

```
  services: ssh
  ports:
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:
        rule family=ipv4 source address="1.1.1.0/8"
         ➥service name="samba" accept
        rule protocol value="icmp" accept
# firewall-cmd --permanent --zone=internal --get-target
default
```

This setup demonstrates a three-layer nested firewall. The outermost layer, public, is an interface zone and spans the entire world. The next layer, internal, is a source zone and spans your organization, which is a subset of public. Finally, a rich rule adds the innermost layer spanning your workgroup, which is a subset of internal.

The take-away message here is that when a scenario can be broken into nested layers, the broadest layer should use an interface zone, the next layer should use a source zone, and additional layers should use rich rules within the source zone.

Debugging

Firewalld employs intuitive paradigms for designing a firewall, yet gives rise to ambiguity much more easily than its predecessor, iptables. Should unexpected behavior occur, or to understand better how firewalld works, it can be useful to obtain an iptables description of how netfilter has been configured to operate. Output for the previous example follows, with forward, output and logging lines trimmed for simplicity:

```
# iptables -S
-P INPUT ACCEPT
... (forward and output lines) ...
-N INPUT_ZONES
-N INPUT_ZONES_SOURCE
-N INPUT_direct
```

```
-N IN_internal
-N IN_internal_allow
-N IN_internal_deny
-N IN_public
-N IN_public_allow
-N IN_public_deny
-A INPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -j INPUT_ZONES_SOURCE
-A INPUT -j INPUT_ZONES
-A INPUT -p icmp -j ACCEPT
-A INPUT -m conntrack --ctstate INVALID -j DROP
-A INPUT -j REJECT --reject-with icmp-host-prohibited
... (forward and output lines) ...
-A INPUT_ZONES -i eno1 -j IN_public
-A INPUT_ZONES -i eno2 -j IN_public
-A INPUT_ZONES -j IN_public
-A INPUT_ZONES_SOURCE -s 1.1.0.0/16 -g IN_internal
-A IN_internal -j IN_internal_deny
-A IN_internal -j IN_internal_allow
-A IN_internal_allow -p tcp -m tcp --dport 22 -m conntrack
 ➥--ctstate NEW -j ACCEPT
-A IN_internal_allow -s 1.1.1.0/8 -p udp -m udp --dport 137
 ➥-m conntrack --ctstate NEW -j ACCEPT
-A IN_internal_allow -s 1.1.1.0/8 -p udp -m udp --dport 138
 ➥-m conntrack --ctstate NEW -j ACCEPT
-A IN_internal_allow -s 1.1.1.0/8 -p tcp -m tcp --dport 139
 ➥-m conntrack --ctstate NEW -j ACCEPT
-A IN_internal_allow -s 1.1.1.0/8 -p tcp -m tcp --dport 445
 ➥-m conntrack --ctstate NEW -j ACCEPT
-A IN_internal_allow -p icmp -m conntrack --ctstate NEW
 ➥-j ACCEPT
-A IN_public -j IN_public_deny
-A IN_public -j IN_public_allow
-A IN_public -j DROP
-A IN_public_allow -p tcp -m tcp --dport 80 -m conntrack
```

```
 ➥--ctstate NEW -j ACCEPT
-A IN_public_allow -p tcp -m tcp --dport 443 -m conntrack
 ➥--ctstate NEW -j ACCEPT
```

In the above iptables output, new chains (lines starting with `-N`) are first declared. The rest are rules appended (starting with `-A`) to iptables. Established connections and local traffic are accepted, and incoming packets go to the `INPUT_ZONES_SOURCE` chain, at which point IPs are sent to the corresponding zone, if one exists. After that, traffic goes to the `INPUT_ZONES` chain, at which point it is routed to an interface zone. If it isn't handled there, icmp is accepted, invalids are dropped, and everything else is rejected.

Conclusion

Firewalld is an under-documented firewall configuration tool with more potential than many people realize. With its innovative paradigm of zones, firewalld allows the system administrator to break up traffic into categories where each receives a unique treatment, simplifying the configuration process. Because of its intuitive design and syntax, it is practical for both simple single-zoned and complex multi-zoned configurations.■

Nathan Vance is a computer science major at Hope College in Holland, Michigan. He installed Linux Mint 12 as a high school junior and now prefers Arch Linux. He drives a home-built electric-powered '95 Ford Probe with a Raspberry Pi car computer.

William Polik is a computational chemistry professor at Hope College in Holland, Michigan. He cut his programming teeth with Turbo Pascal 3 in 1986 and joined the Linux revolution with Red Hat 5 in 1997. He founded two web-based software companies: DiscusWare LLC and WebMO LLC.

HARD DRIVE RESCUE with a Raspberry Pi and Relay

Automate a monotonous routine with a setup powered by Linux to solve a real-world problem.

ANDREW NII ADDO

PREVIOUS
Feature: Understanding Firewalld in Multi-Zone Configurations

NEXT
Doc Searls' EOF

Linux Journal previously has published articles that provide insight on the applications of udev, ddrescue and Raspberry Pi home automation employing the use of relays. This article combines the unique features of each of those tools to solve the issue of failing hard disks.

My Uncle Tee has a knack for relegating most of his computer-related problems to me. During our last visit, I was confronted with a task of transferring the files on an old 750GB USB hard disk to a newer one. My knee-jerk reaction was to plug the device in to my Linux box, mount it and use `cp` or `rsync` to replicate all the files to the new disk.

And, that's exactly what I did. With the old and new disk drives plugged in to USB ports and mounted at /media/usb0 and /media/usb1, respectively, I proceeded to invoke the `rsync` command. (Use your distro's package manager to install rsync if it is not already present.) I must mention, however, that it took a considerable amount of time for the old drive device to register itself, and the mounting time also was noticeably protracted. All of this indicated a failing hard disk drive:

```
$ rsync -av --progress --inplace \
/media/usb0/* /media/usb1/
```

Everything seemed okay, and I continued nursing my warm cup of favorite black tea—at least, until it started to look otherwise. I could not fail to notice the input/output error messages that started to appear in rsync's verbose output. Then, rsync itself exited abruptly with error code 23. This translates to "partial transfer" from the errcode.h header file in rsync's sources. Immediately re-running the same command confirmed the source no longer existed.

The old hard disk device that was mounted at /dev/usb0 was not present in the output of `lsusb` and `blkid`. It was suspended after the first 13GB was transferred, and I had to power-cycle the hard disk for it to show up again in the list of detected devices. rsync exited prematurely again in another attempt, this time after a few megabytes.

In fact, a few more attempts followed, each one adding a decreasing amount to the transferred files. I obviously was making little progress with this laborious process, which also required a lot of attention. That was when it occurred to me as to why my Uncle Tee wanted to relegate this task to me.

Apparently, the hard disk had not registered on his Windows machine for some time now, and with no available backup, he was in need of a miracle.

I needed a solution that would just continue where it left off after a previous unsuccessful attempt. A solution that skipped the mounting step also could improve the overall process in rescuing the contents of the disk. Replicating the files always could be done at a later stage when an image was available. ddrescue came in handy here. It maintains a log file that is used to resume the rescue process. A good review of ddrescue is available at http://www.linuxjournal.com/magazine/hack-and-when-disaster-strikes-hard-drive-crashes. (Again, use your Linux distro's package manager to install ddrescue.) I basically ran the following command with root privileges:

```
# ddrescue -dv /dev/sdb1 ./freeagent.img \
./freeagent.log
```

The /dev/sdb1 device corresponds to the source partition I wanted to recover. I ensured that there was enough space on the destination partition for the freeagent.img image file to be generated. The last argument is a log file maintained by ddrescue to make resumption possible. Although this is optional, it will be very much needed in this case, as the ddrescue process will be resumed a number of times.

It is also possible to run ddrescue with a retry option, in which case it retries bad blocks a specified number of times before proceeding. However, it's not advisable to use this option in this situation, as it could wind the failing hard disk down to a halt at a faster rate. With the log file in place, it is always possible to rerun ddrescue after a first complete scan to retry the bad blocks. The generated image file will be updated accordingly.

Armed with the above knowledge, I planned the recovery process:

1. Run the `ddrescue` command as above.

2. Power-cycle the hard disk when it suspends.

3. Wait until the device is detected by the machine and get the new device name.

4. Go to step 1, updating the source device with the name from step 3.

After looping through the above sequence a few times and also considering the sometimes tiny progress being made on each iteration, one thing was clear: I was going to spend a great part of the rest of my life staring at a console screen waiting for the return of an error message, manually power-cycling the hard disk, restarting the rescue process and start staring again. I needed to upgrade my solution to free me from this torture. This solution should require as little human intervention as possible from the beginning until the full image is generated. This was when I started to explore the possibility of employing a Raspberry Pi for the task. (The reasoning that led to this choice will become clear as the solution takes shape.)

I happened to own a Raspberry Pi 2 model B. I dumped the latest version of the Raspbian image onto a MicroSD card, and this is how the solution started. (Note: you can find a good how-to on how to get this running at https://www.raspberrypi.org/documentation/raspbian.) You can use any of the available Raspberry Pi models, although you might need to adapt the steps that follow slightly (I'll try to point it out if your mileage will differ as a result of using a different model).

Step 1 is the one-liner as already shown above. But, what happens after a power cycle? A udev rule could be added to resume the ddrescue recovery when the device is detected after being powered on. udev is currently the default device manager for the Linux kernel, and it comes pre-installed in almost all modern distributions, including the Raspbian image of the Raspberry Pi.

To create a rule, you need to know the device information of the hard disk. Once you have the device name and the partition to be recovered, you can call up the rest of the information with the following:

```
$ udevadm info --query=property --name /dev/sdb1
```

You also can monitor the flow of information during device detection by running the following command before plugging in the USB hard disk and powering it up:

```
$ udevadm monitor --environment
```

A stripped-down version of the section I am looking for looks like the following:

```
UDEV [6899.460576] add /devices/platform/soc/...
ACTION=add
DEVNAME=/dev/sdb1
DEVTYPE=partition
ID_SERIAL=ST3750640AS_5QD463QL
ID_SERIAL_SHORT=5QD463QL
ID_TYPE=disk
SUBSYSTEM=block
TAGS=:systemd:
```

You will use this information to prepare the udev rule that will be saved in the /etc/udev/rules.d directory. It is a good idea not to add your rule directly to any of the existing default files. Create a new file following the naming conventions for your new rule. I named my file 90-freeagent.rules, and it contains a rule to match the USB hard disk using the information from above:

```
ACTION=="add", KERNEL=="sd?1", \
ENV{SUBSYSTEM}=="block", \
ENV{ID_SERIAL}=="ST3750640AS_5QD463QL", \
ENV{DEVTYPE}=="partition", \
RUN+="/opt/bin/freeagent.sh '%E{DEVNAME}'"
```

With the exception of the last section, notice that the rule basically is made up of a string matching the attributes obtained above. The last section specifies the path to a script to be run when a match is made. Notice also that the syntax allows me to pass the name of the device detected as an argument to the script. Now you safely can check steps 3 and 4 as done.

There is one more fact about udev worth mentioning: `RUN` can be used only for very short-running foreground tasks. Scripts with a protracted duration will be terminated prematurely and unconditionally after the event handling has finished. udev enforces this to prevent

The Raspberry Pi makes interfacing a relay to a PC even easier, which adds more weight to this choice of target device or platform on which to implement the solution.

blocking all further events for the device or a dependent one. There are a number of ways to circumvent this; some obviously are more elegant than others. I relocated the main job to another script and delegated this to the atd dæmon. Install the "at" package if you are following along.

Now you can roll out the contents of the script that is referenced by the `RUN` section of the udev rule:

```
#!/bin/bash
export HDDEVNAME=$1
at -f "/opt/bin/ddfreeagent.sh" now
```

The worker script, named ddfreeagent.sh, for now, will contain the following:

```
#!/bin/bash

IMG=/media/usb0/freeagent/freeagent.img
LOG=/media/usb0/freeagent/freeagent.log

/usr/bin/ddrescue -dv ${HDDEVNAME} ${IMG} ${LOG}
```

Ensure that both scripts have the execute permissions set.

Step 2 currently is the only one outstanding. You need a way to power-cycle the hard disk when it suspends. This sounds like a task for a relay. The Raspberry Pi makes interfacing a relay to a PC even easier, which adds more weight to this choice of target device or platform on which to

implement the solution. It already comes equipped with a set of easy-to-use GPIO pins that could be used to perform the power-cycle procedure through a relay.

The eventual solution employs a solid-state relay whose control signals come from the GPIO pins of a Raspberry Pi unit. The relay controls the power to the hard disk. The rescue process starts by triggering the relay to turn on the hard disk. The Raspberry Pi senses the device, and the udev rule implemented above spawns a script that starts the ddrescue process. This comes to an abrupt end when the hard disk suspends and returns with an error code, which is caught and processed accordingly. The relay then fires a reset sequence to the hard disk power supply in case of an error return from ddrescue, and the whole process starts again. The log file maintained by ddrescue means the rescue operation resumes from where it left off on the previous attempt. A success return from ddrescue ends the rescue process.

As you may have gathered from the above summary, this step involves playing with naked wires that will carry dangerous voltage levels. It is very important to observe safety precautions. Make sure you do all the work when the devices are off and can worked on safely. I recommend that you proceed with the implementation only when you are sure of what you are doing.

I managed to find a solid-state relay at a local store to use to control a load running on AC power. This is essential, as relays that control DC-powered loads also are available, and they are by no means interchangeable. Working with these little boxes is very convenient, as they also respond to voltages as little as 3.3V. As this also happens to be the voltage level of the Raspberry Pi output pins, this means they can be connected directly. Other relay types (mostly coil-based conventional relays) require a separate circuit to bridge them to the Raspberry Pi unit. This is necessary to raise the 3.3V to levels that can trigger the relay. These circuits isolate the load and provide added protection to the unit. Solid-state relays typically come with built-in isolation circuits.

Having all the needed tools at hand, I proceeded to work on step 2. I identified one of the general-purpose pins on the board for use as an output pin. This pin, together with the ground pin, will be connected

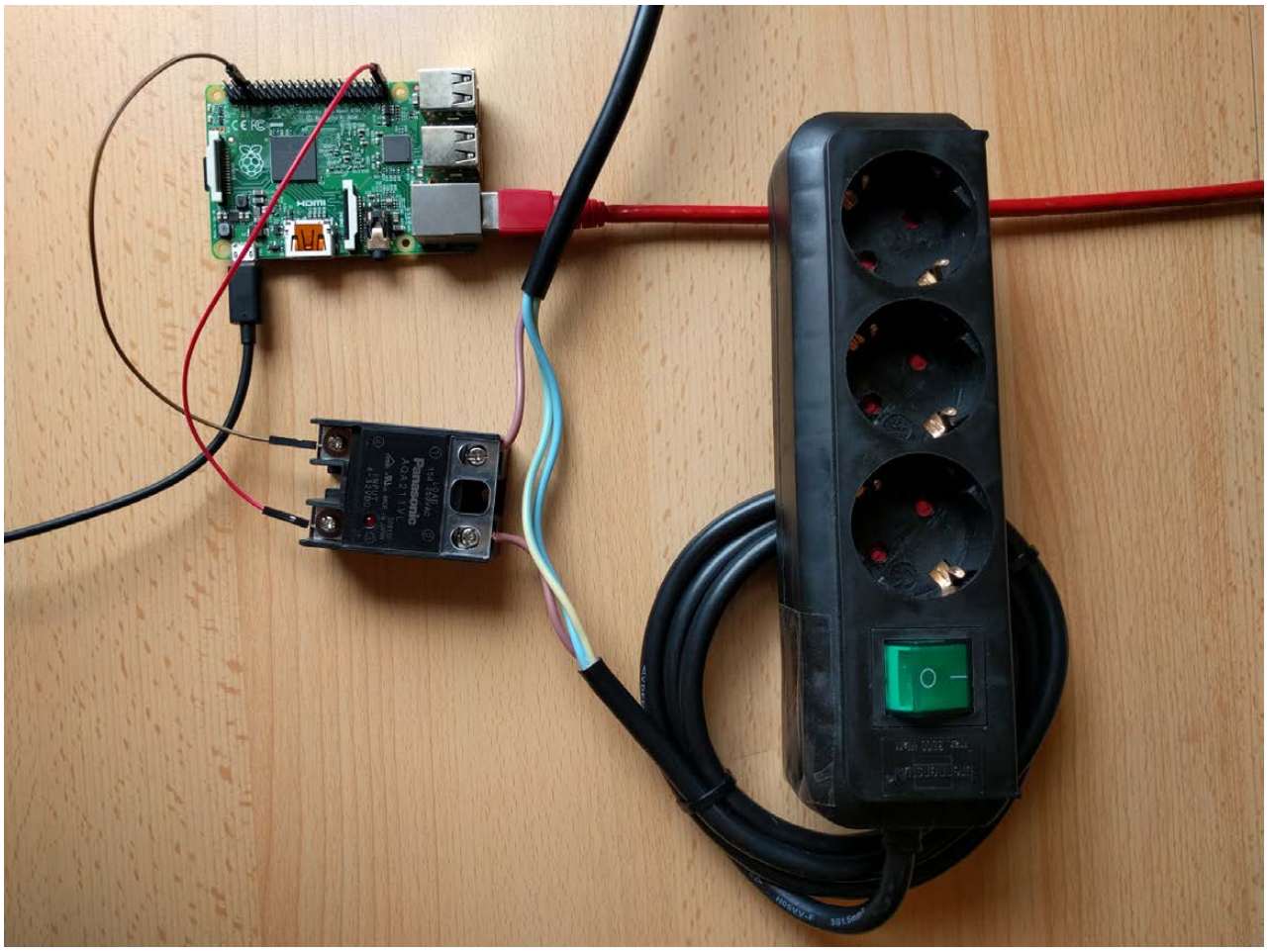


Figure 1. Hardware Setup

to the input side of the relay. The relay is clearly labeled: the ground pin goes to the negative terminal, and the chosen output pin goes to the positive.

Next, I found an old extension cord, striped off the insulation at a section along its length and severed the live cable. Both ends were then fed into the output or load terminals of the relay. This extension cord will be used to power only the failing hard disk. Figure 1 shows a picture of the setup.

Two steps are needed to prepare the selected GPIO pin for output on the console. First, export the pin for the operating system to prepare the direction files. There are two different ways to refer to pins, and this can be a source of great confusion. Physical numbering is the

natural way to refer to the pins, and it counts across and down from pin 1 at the top left (nearest to the SD card). GPIO numbering, on the other hand, refers to how the computer sees the pins and does not follow any particular order. You need to check the particular model of Raspberry Pi you are using and identify the correct way to refer to the pins. I use the GPIO numbers in the script here.

I used pin 40 (GPIO 21) on the Raspberry Pi 2 model B as my output pin. Run the following to export the pin:

```
$ echo "21" > /sys/class/gpio/export
```

Next, specify the direction of use—an output pin in this case:

```
$ echo "out" > /sys/class/gpio/gpio21/direction
```

With this done, and the above setup in place, the following command switches on the hard disk:

```
$ echo "1" > /sys/class/gpio/gpio21/value
```

Switch it off with this:

```
$ echo "0" > /sys/class/gpio/gpio21/value
```

At the end of everything, you need to clean up by un-exporting the pin:

```
$ echo "21" > /sys/class/gpio/unexport
```

Putting it all together, now you can update the contents of the ddfreeagent.sh worker script (Listing 1).

I also have included some logs to give me an idea of how long the whole process took and how many times I otherwise would have had to power-cycle the hard drive manually. Kickstart the chain process by triggering the relay to turn on the hard disk.

After about 27 hours and some 130 hard disk power cycles, ddrescue finally exited with success. (Your mileage will, of course, vary depending

Listing 1. ddfreeagent.sh

```
#!/bin/bash

IMGFILE=/media/usb0/freeagent/freeagent.img
LOGFILE=/media/usb0/freeagent/freeagent.log
RUNLOG=/media/usb0/freeagent/ddrun.log
PIN=21

do_log ()
{
  echo -n $(date) >> ${RUNLOG}
  echo -n "," >> ${RUNLOG}
  echo -n $(cat /proc/uptime | cut -d' ' -f1) >> ${RUNLOG}
  echo -n "," >> ${RUNLOG}
  echo $1 >> ${RUNLOG}
}

#Initialize log file
if ! [ -f ${RUNLOG} ]; then
  do_log START
fi

/usr/bin/ddrescue -dv ${HDDEVNAME} ${IMGFILE} ${LOGFILE}
if [ $? -eq 0 ]
then
  #Yay! ddrescue completed successfully.
  do_log PASS

  # Clean up now and exit
  echo "0" > /sys/class/gpio/gpio${PIN}/value
  echo ${PIN} > /sys/class/gpio/unexport

  exit
else
  #Oops; another ddrescue error.
  do_log FAIL

  #Power-cycle the hard disk.
  echo "0" > /sys/class/gpio/gpio${PIN}/value
  sleep 10
  echo "1" > /sys/class/gpio/gpio${PIN}/value
fi
```

on the size and state of the hard disk and also on the Raspberry Pi model you're using.) This meant a mountable image of the failing hard disk was ready. I then mounted the image with a simple mount command:

```
$ mount /media/usb0/freeagent/freeagent.img /media/usb3/
```

And finally, I copied them to a backup directory:

```
$ rsync -av --progress --inplace /media/usb3/* /media/usb0/backup
```

Remember to disable the process when you're done to avoid unwanted runs of the cycle. You can remove the execute permissions from the udev target scripts and/or comment out the line containing the udev rule.

And, this concludes the story of how I managed to get Uncle Tee the miracle he so badly needed. All the files he couldn't do without were recovered successfully. I kept smiling as I sat there watching the hard disk being power-cycled, having to do nothing myself. There were no interruptions this time, as I sipped my favorite black tea.■

Andrew Addo works as an engineer with a leading navigation solution provider. He recently added salsa to his list of hobbies, and he welcomes your comments sent to and.addo@gmail.com.

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

RETURN TO CONTENTS

Identity: Our Last Stand

Time to get root for our selves.

DOC SEARLS

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

PREVIOUS
Feature: Hard Drive Rescue with a Raspberry Pi and Relay

Linux has built countless cathedrals, but still no bazaar.

By that I mean every corporate cathedral you can shake a mouse at is full of Linux, yet Linux has not yet enabled a free and open marketplace for every business and every customer. Instead, every human being on the commercial net remains trapped in corporate cathedrals, many of which are ravenous for the blood of personal data, most of which is acquired by surveillance. In fact, nearly our entire existence in the commercial world is inside cathedrals where we have near-zero autonomy and great exposure to whatever those running the cathedrals wish to know about us.

The wide-open bazaar—the open public marketplace—where we can roam free, as anonymous or selectively know-able as we please, still doesn't exist online. And it should, because the

"Give me a place to stand and I can move the world", Archimedes said. Each of us has that place with the internet. What we lack is a fulcrum.

internet protocol was built to support it. Just because it isn't there yet doesn't mean we shouldn't build it. Hell, commercial activity has existed on the internet only for 21 years so far. (Starting on April 30, 1995—that's when the NSFnet, the last of the internet backbones that forbade commercial traffic, stood down.)

I know this isn't what Eric S. Raymond (http://www.catb.org/%7Eesr) was talking about in *The Cathedral and the Bazaar* (his landmark book about software development, published back at the turn of the millennium: http://www.catb.org/%7Eesr/writings/cathedral-bazaar/cathedral-bazaar). Eric was talking about development styles, contrasting closed "cathedral" environments with open "bazaar" ones. Linux was, and remains, the greatest exemplar of bazaar-style development at work: a fact owed in no small measure to Eric's evangelism of Linux and open source, much of it on these very pages (http://www.linuxjournal.com/googlesearch?s=Eric%2520S.%2520Raymond).

I'm borrowing Eric's metaphors here for two reasons. One is that I hope it motivates some readers to admit that Linux has been used at least as much to build corporate (and government) cathedrals as to liberate the geeks who continue to write open-source code that makes building anything possible. The other is that we need another coterie of alpha geeks working today on creating an open marketplace, setting everyone free from the countless closed ones that have become the norm and have made the surveillance economy possible.

"Give me a place to stand and I can move the world", Archimedes said. Each of us has that place with the internet. What we lack is a fulcrum.

That fulcrum isn't a machine. It's identity. We need to have root for our own identities online. We have it in the offline world, but not yet online.

Getting that root is our challenge. With root for our own identities, we will be able to go about our business anonymously by default, and identify ourselves selectively on a need-to-know basis. That includes being able to call ourselves whatever we please when dealing with other entities in the world, and then engaging administrative systems—such as those in the world's many cathedrals—in full control over what we share, what we don't and how we leverage the same data, and attached permissions, across all those systems.

Let's look at the physical world for a moment. By default, we are anonymous to others there—literally, nameless. For example, when we walk down a city street, we do not want or need everybody we pass or encounter to know who we are, or anything about us, other than the fact that we are human and participating in society. When we meet somebody, we may introduce ourselves by our first names or nicknames. Or, we may give somebody a business card. Asked for our name at the counter of a coffee shop, we can tell them anything. I've met more than one guy named Mike who uses a different name—Clive or something—because the name Mike is so common. At a conference, we may wear a name badge, but even in those cases, some people still just use their first names or turn their badges around.

What happens in all these cases is data sharing on a *need-to-know* basis that we control. Being able to do so is a grace of civilization. Not being able to do so is a curse of celebrity, and a useful case in point. Being known by all is a Faustian bargain (http://www.dictionary.com/browse/faustian-bargain). And we are all Fausts online today, whether we like it or not.

Faust was the scholar in German legend who sold his soul to the devil for unlimited knowledge and worldly pleasure (https://en.wikipedia.org/wiki/Faust). The difference with us is that we don't sell personal data about ourselves. We don't even give it away. We just acquiesce to ubiquitous surveillance, through which all kinds of personal data gets snarfed up without our knowing much, if anything, about it.

The bishops in charge of personal data acquisition in today's corporate cathedrals are the Chief Marketing Officers (a title that hardly existed in the pre-internet world) or their equivalents. They and their many agents believe it is both possible and desirable to know

Thanks to growing Big Data budgets and appetites, and absent legal and technical restraints, the market for personal data has become vast and complex beyond any one party's full understanding.

everything about users and customers, either by direct surveillance through browsers and apps or indirectly through access providers and other third parties.

Thanks to growing Big Data budgets and appetites, and absent legal and technical restraints, the market for personal data has become vast and complex beyond any one party's full understanding. It even includes real-time data, harvested from cookies and other tracking files, sold by auction to help guide advertising messages directly toward crosshairs on eyeballs and eardrums.

As if all this were not bad enough, everybody interacting with these cathedrals online has the added burden of needing separate passports—logins and passwords—to clear customs at every entrance.

ADVERTISER INDEX

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

In "Doing for User Space What We Did for Kernel Space" (published in *LJ* two months ago: http://www.linuxjournal.com/content/doing-user-space-what-we-did-kernel-space), I gave the examples of what a few startups are doing to give us identity root. There are, and should be, many more working on the same case. And soon. Because **identity is our last stand**. Making it ours, finally and absolutely, is the only way we secure our independence and liberty online. It is the only way the world's economy becomes a true bazaar.

It's a handy thing that we can get together soon to talk about it and work on code: next month, at the next Internet Identity Workshop (http://www.internetidentityworkshop.com), on October 25–27, 2016. I have co-hosted these with Phil Windley (http://www.windley.com) and Kaliya Hamlin (aka IdentityWoman: http://identitywoman.net) since 2005. IIW, as it is best known, is a three-day unconference (https://en.wikipedia.org/wiki/Unconference) held twice a year at the Computer History Museum (http://www.computerhistory.org) in Silicon Valley. It's cheap as conferences go. The charge just covers our expenses; we don't make money off it. (In fact, if you can send sponsors our way, that'll help too. Sponsors pay for the food, which is always good.) Register at https://www.eventbrite.com/e/internet-identity-workshop-xxiii-23-2016b-tickets-25853411249.

And see you there—as whatever you want to call yourself.■

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

RETURN TO CONTENTS

drupalize.me

Instant Access to Premium Online Drupal Training

- *Instant access to hundreds of hours of Drupal training with new videos added every week!*
- *Learn from industry experts with real world experience building high profile sites*
- *Learn on the go wherever you are with apps for iOS, Android & Roku*
- *We also offer group accounts. Give your whole team access at a discounted rate!*

Learn about our latest video releases and offers first by following us on Facebook and Twitter (@drupalizeme)!

Go to http://drupalize.me and
get Drupalized today!