

Sidekiq | IPv6 | Compliance | Oregano | RPi Clusters

LINUX JOURNAL

Since 1994: The Original Magazine of the Linux World

MAY 2013 | ISSUE 229 | www.linuxjournal.com

WHAT YOU NEED
TO KNOW ABOUT
DEPLOYING
OPEN SOURCE
IN THE CLOUD

RASPBERRY PI!



Build a Home Backup,
Multimedia and
Print Server with RPi

Use an RPi as
a Low-Cost IPv6
Router and
Tunnel Endpoint

How-to: Make
a Redundant
RPi Cluster

Create a Light
Controller with an RPi

SPEED UP YOUR
WEB APPLICATIONS
WITH SIDEKIQ

PREPARE FOR THE
GOOGLE READER
SHUTDOWN

DESIGN YOUR
OWN CIRCUITRY
WITH OREGANO

O'REILLY®

Velocity

Web Performance
and Operations

CONFERENCE

Building a Faster
and Stronger Web

June 18–20, 2013

Santa Clara, CA



Velocity is much more than a conference; it's become the essential training event for web operations and development professionals from companies of all sizes.

Experience Velocity

- Three days of in-depth workshops, sessions, and plenaries
- Access to the foremost industry leaders and more than 2000 professionals building a faster, stronger web
- An Exhibit Hall featuring dozens of the latest tools and products
- Fun evening events, Birds of a Feather sessions, and plenty of networking opportunities

Conference Tracks:

- Web Performance
- Operations
- Mobile Performance
- Velocity Culture

June 18–20, 2013 | Santa Clara, CA

velocityconf.com/sc

“Velocity is the conference where people talk about how to get things done in the real world — if you want to know how the best in the world handle their operations and site performance, **Velocity is the place to learn.**”

Register today to
reserve your seat.

SAVE 20%
WITH CODE LNXJ20

zStax

ZFS Unified Storage

Father and son take their need for speed from the track to the data center.

Is your current storage solution slowing down your Tier 1 applications?

Steve and Tommy Scherer, two Silicon Mechanics experts, know that storage performance is crucial for today's datacenter deployments. The zStax StorCore 104 unified storage appliance exceeds the performance requirements of today's business without restricting customers to legacy proprietary hardware.

The zStax StorCore 104 is specifically tailored to meet the storage requirements for datacenter technologies like virtualization, cloud computing, VDI, and software-defined storage architectures at a fraction of the cost of legacy storage vendors.

Take a ride on the zStax StorCore 104. Best-in-class storage, full of win.



Steve Scherer

Tommy Scherer

RASPBERRY PI

FEATURES

66 **Raspberry Pi: the Perfect Home Server**

The low-cost, energy-efficient Raspberry Pi is an excellent choice for a home backup, multimedia and print server.

Brian Trapp

78 **Autoconfiguring an IPv6 Access Point with SixXS and a Raspberry Pi**

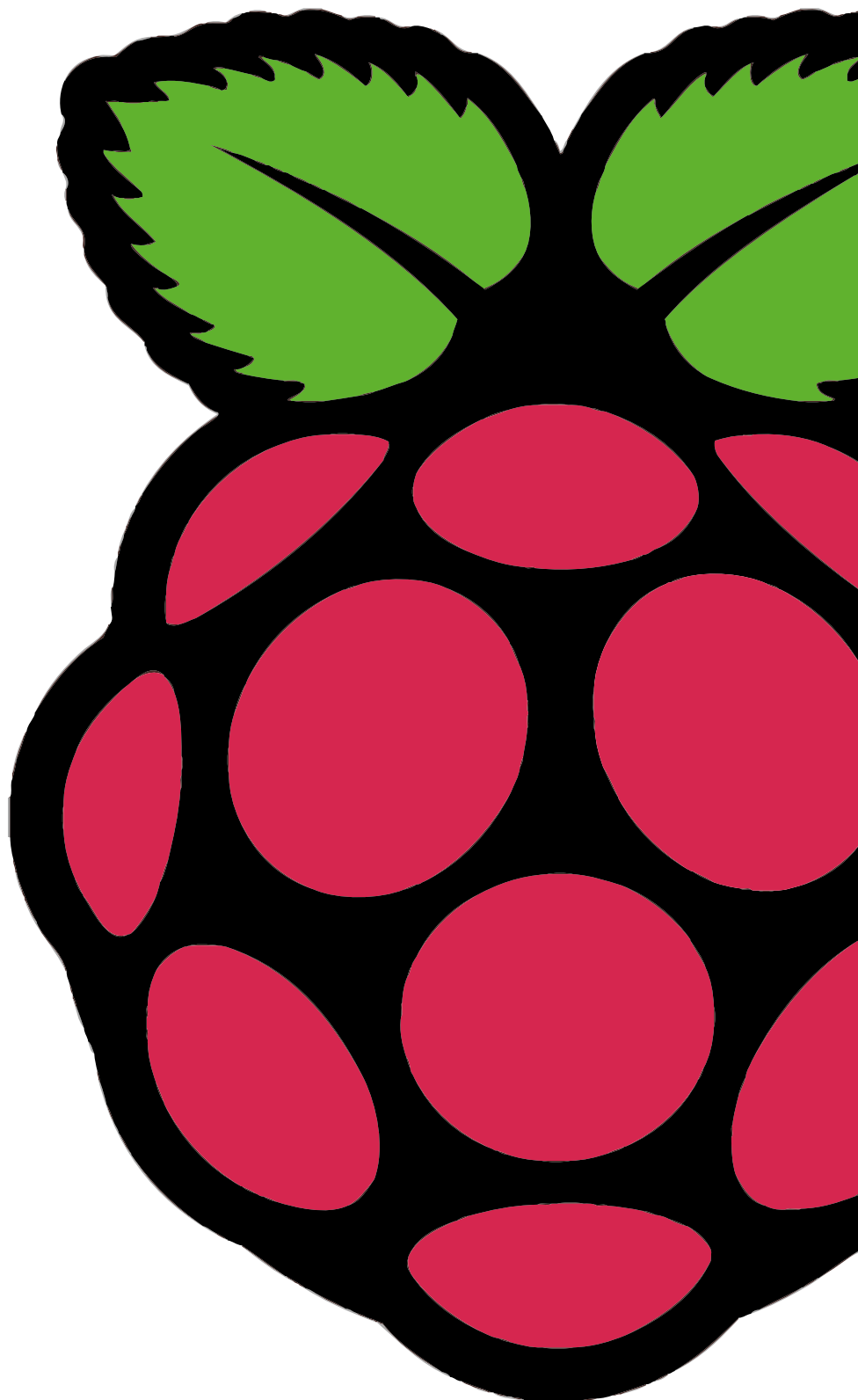
Set up a Raspberry Pi as an IPv6 router and tunnel endpoint that you can use to provide IPv6 access to your home or office LAN.

Igor Partola

84 **Control the Limelight with a Raspberry Pi**

Take the stage like a pro with a Raspberry Pi light controller.

Jonathan Brogdon



INDEPTH

98 Effects of Cloud Computing on Open-Source Compliance

Deploying open source to the cloud? This is what you need to know.

Diana Marina Cooper

COLUMNS

34 Reuven M. Lerner's At the Forge

Sidekiq

42 Dave Taylor's Work the Shell

Summing Up Points

46 Kyle Rankin's Hack and /

Two Pi R

54 Shawn Powers' The Open-Source Classroom

The Google Giveth

106 Doc Searls' EOF

One Hand Slapping

IN EVERY ISSUE

8 [Current_Issue.tar.gz](#)

12 [Letters](#)

18 [UPFRONT](#)

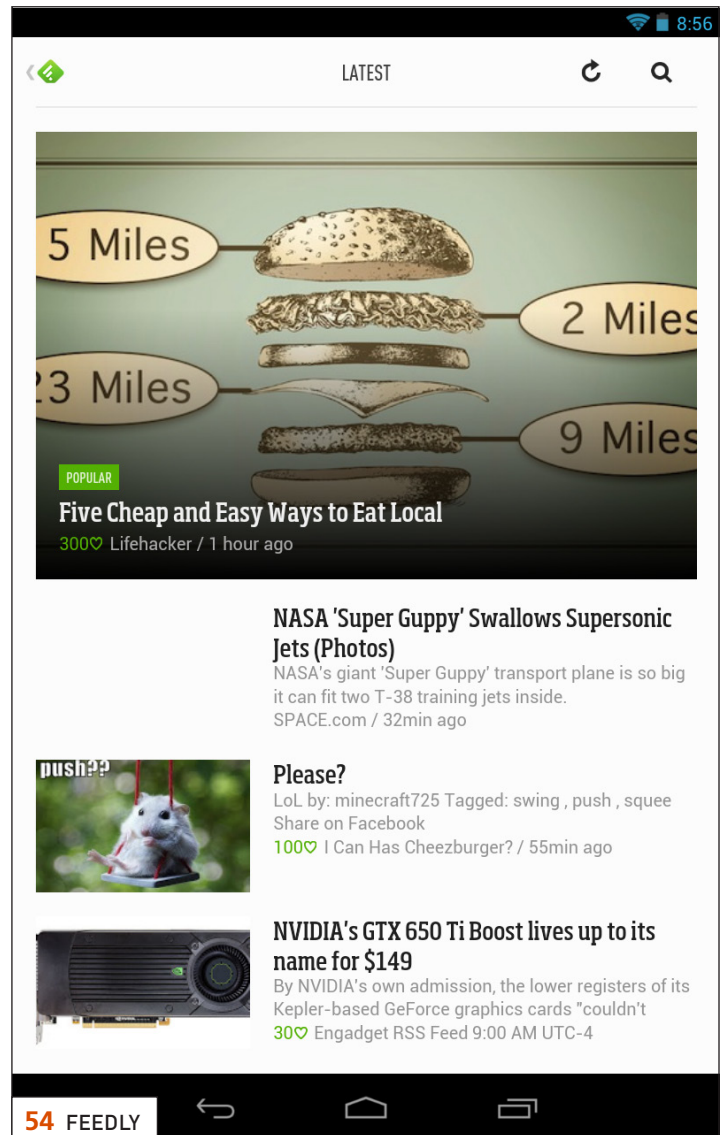
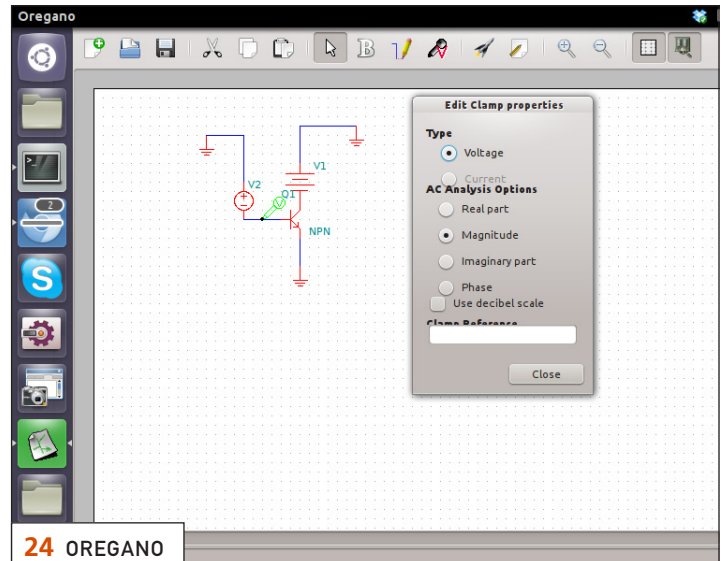
32 [Editors' Choice](#)

62 [New Products](#)

111 [Advertisers Index](#)

ON THE COVER

- [What You Need to Know about Deploying Open Source in the Cloud](#), p. 98
- [Build a Home Backup, Multimedia and Print Server with RPi](#), p. 66
- [Use an RPi as a Low-Cost IPv6 Router and Tunnel Endpoint](#), p. 78
- [How-to: Make a Redundant RPi Cluster](#), p. 46
- [Create a Light Controller with an RPi](#), p. 84
- [Speed Up Your Web Applications with Sidekiq](#), p. 34
- [Prepare for the Google Reader Shutdown](#), p. 54
- [Design Your Own Circuitry with Oregano](#), p. 24



LINUX JOURNAL™

Subscribe to
Linux Journal
Digital Edition
for only
\$2.45 an issue.



ENJOY:

- Timely delivery
- Off-line reading
- Easy navigation
- Phrase search and highlighting
- Ability to save, clip and share articles
- Embedded videos
- Android & iOS apps, desktop and e-Reader versions

SUBSCRIBE TODAY!

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

Publisher Carlie Fairchild
publisher@linuxjournal.com

Director of Sales John Grogan
john@linuxjournal.com

Associate Publisher Mark Irgang
mark@linuxjournal.com

Webmistress Katherine Druckman
webmistress@linuxjournal.com

Accountant Candy Beauchamp
acct@linuxjournal.com

**Linux Journal is published by, and is a registered trade name of,
Belltown Media, Inc.**

PO Box 980985, Houston, TX 77098 USA

Editorial Advisory Panel

Brad Abram Baillio • Nick Baronian • Hari Boukis • Steve Case
Kalyana Krishna Chadalavada • Brian Conner • Caleb S. Cullen • Keir Davis
Michael Eager • Nick Faltys • Dennis Franklin Frey • Alicia Gibb
Victor Gregorio • Philip Jacob • Jay Kruiuzenga • David A. Lane
Steve Marquez • Dave McAllister • Carson McDonald • Craig Oda
Jeffrey D. Parent • Charnell Pugsley • Thomas Quinlan • Mike Roberts
Kristin Shoemaker • Chris D. Stark • Patrick Swartz • James Walker

Advertising

E-MAIL: ads@linuxjournal.com
URL: www.linuxjournal.com/advertising
PHONE: +1 713-344-1956 ext. 2

Subscriptions

E-MAIL: subs@linuxjournal.com
URL: www.linuxjournal.com/subscribe
MAIL: PO Box 980985, Houston, TX 77098 USA

LINUX is a registered trademark of Linus Torvalds.

High Performance, High Density Servers for Data Center, Virtualization, & HPC

On-board 10 Gigabit Ethernet and Infiniband for greater throughput in less rack space

The Intel® Xeon® Processor E5-2600 family powers the highest-density servers iXsystems has to offer. The iXR-1204 +10G features dual onboard 10GigE + dual onboard 1GigE network controllers, up to 768GB of RAM and dual Intel® Xeon® E5-2600 family processors, freeing up critical expansion card space for application-specific hardware. The uncompromised performance and flexibility of the iXR-1204 +10G makes it suitable for clustering, high-traffic webservers, virtualization, and cloud computing applications - anywhere you need the most resources available.

For even greater performance density, the iXR-22X4IB squeezes four server nodes into two units of rack space, each with dual Intel® Xeon® E5-2600 Family Processors, up to 256GB of RAM, and an on-board Mellanox® ConnectX QDR 40Gbp/s Infiniband w/QSFP Connector. The iXR-22X4IB is perfect for high-powered computing, virtualization, or business intelligence applications that require the computing power of the Intel® Xeon® Processor E5-2600 family and the high throughput of Infiniband.

iXR-1204 +10G

- Dual Intel® Xeon® Processors E5-2600 Family
- Intel® X540 Dual-Port 10 Gigabit Ethernet Controllers
- Up to 16 Cores and 32 process threads
- Up to 768GB Main Memory
- 700W Redundant high-efficiency power supply

iXR-22X4IB

- Dual Intel® Xeon® Processors E5-2600 Family per node
- Mellanox® ConnectX QDR 40Gbp/s Infiniband w/QSFP Connector per node
- Four server nodes in 2U of rack space
- Up to 256GB Main Memory per server node
- Shared 1620W Redundant high-efficiency Platinum level (91%+) power supply



Intel, the Intel logo, and Xeon Inside are trademarks or registered trademarks of Intel Corporation in the U.S. and other countries.

E5-2600

HIGH &
Throughput
INCREDIBLE
Performance Density

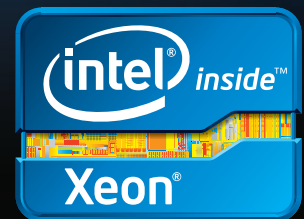


iXR-1204+10G: 10GbE On-Board

4 Server
Nodes
in 2U



iXR-22X4IB



Call iXsystems toll free or visit our website today! **1-855-GREP-4-IX** | www.iXsystems.com



SHAWN POWERS

A Pint-Size Powerhouse with a Tasty-Sounding Name

It's hard not to love the Raspberry Pi. For less money than it takes to stop at a drive-through, it's possible to buy the top-end RPi model. (Granted, I have three teenagers, so the drive-through is expensive.) The Raspberry Pi is as open as the manufacturer can make it, has huge community support, and its software isn't backed by a huge corporation with mysterious motivations (*cough*, Android, *cough*). It's just a cool little ARM device that runs Linux. This month, we focus on the Raspberry Pi. The drive-through is optional.

Dave Taylor doesn't work directly with a Raspberry Pi this month, but his continuing series on scripting a *Cribbage* game certainly will work on an RPi. Even if you're not a *Cribbage* fan, Dave's scripting lessons are

infinitely useful for learning. Reuven M. Lerner teaches a thing or two this month as well—specifically in regard to running background tasks in your Web applications. Running things in the background can make a monstrous Web application perform much more quickly from the end user's perspective. Reuven shows how with Sidekiq.

Kyle Rankin, who first interested me in Raspberry Pi devices, talks about redundancy this month. What's better than a Raspberry Pi? Redundant pies—or Pis, rather. If you want to bolster your RPi reliability or just want an inexpensive platform to learn clustering, Kyle walks through the process. I've been writing about my Raspberry Pi adventures off and on for the past six months, so this month, I decided to focus on an issue that is near

and dear to my heart: RSS. If the Google Reader shutdown in July has you worried about how you'll browse the Web, perhaps my column will help. I'm past panic mode, and I've been able to wean myself off Google Reader altogether.

I'm probably not the only *Linux Journal* reader who has a full virtualization system with iSCSI SAN/NAS storage in my basement. Unfortunately, along with that nerdy power comes a big electricity bill. Brian Trapp describes the other end of that spectrum with his article on creating the perfect home server—with a Raspberry Pi. There are plenty of reasons powering your Linux server infrastructure with a cell-phone charger is awesome, and Brian explains how. If you want to add IPv6 to the mix, Igor Partola follows up with an article on creating an IPv6 router on your network, even if your ISP doesn't support it!

I realize all my "servers in the basement" stuff is almost cliché when talking about Linux users. To be fair, I didn't say it was my mom's basement. Still, Jonathan Brogdon's article on controlling stage lighting with a Raspberry Pi is a great way to avoid our basement-dwelling stereotype. Jonathan literally puts RPi in the spotlight with his hardware/software combination for controlling external lighting. It's a real-world solution and really cool to read about.

The coming shutdown of Google Reader has made all of us think a little harder about the dangers of cloud computing. At the very least, it's forced us to think about trusting cloud-based services we don't directly control. Diana Marina Cooper talks about the other end of the cloud—specifically as it relates to open-source compliance. How does the GPL relate to a world of "Software as a Service"? If the cloud obfuscates the software and the code, what does that mean with regards to FOSS? Diana takes a serious look at a problem not many of us consider.

Do you have a Raspberry Pi, and are you looking for something to do with it? Do you already have a rack of clustered-RPi devices in your basement, and are you looking for tips on optimization? Regardless of your immersion level into the Pi, this issue should prove useful and entertaining. Even if you have no interest in the Raspberry Pi, this issue is full of the same tech tips and Linux news you're used to seeing. This issue was incredibly fun to put together, and we hope you enjoy it as much as we did. ■

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



Our Certifications

vmware[®]
vCLOUD DATACENTER

vmware[®]
PARTNER
ENTERPRISE
SERVICE PROVIDER

Our Awards

vmware[®]
PARTNER NETWORK AWARD
2013 GLOBAL WINNER

vmware[®]
PARTNER NETWORK AWARD
2013 EMEA WINNER

Discover **Dedicated Cloud**

Designed for your Ambition

VMware vSphere & vCloud

Immediate accessibility

Dedicated resources provisioned

within 5 minutes

Scalable infrastructure

High availability

Hourly or monthly billing

1 month free trial

1-855-684-5463 (toll free)



Private Cloud



Dedicated Servers



Dedicated Cloud

For more details:



OVH.COM

Dedicated Infrastructures For Your Business

or contact us: **1-855-684-5463** (toll free)

letters



Raspberry Pi

We recently asked readers to tweet or comment on our Web site about what they've done with their Pis. Here are some

of the responses:

- @gholmer: Using my Pi to run a Q-Link Reloaded server:
<http://www.lyonlabs.org/commodore/qlink/index.html>
- @packetgeek: One of my #Rpi boards runs Softsqueeze, as a \$35 replacement for a \$300 Squeezebox.
- @retux: Using Raspi with cloud storage davfs and encryption with encfs. A way to extend SD memory lifetime.
- @strawberrybrick: I use my Raspberry Pi as Squeezebox while at work:
<http://my.opera.com/djfake/blog/2012/12/12/squeezeslave-and-the-raspberry-pi>.

- @kitestramrt: My Rasp is #3, housed by @fusabe. Hosts <http://drift.to/pap>, #tt-rss, #ownCloud and drift.to (Figure 1).

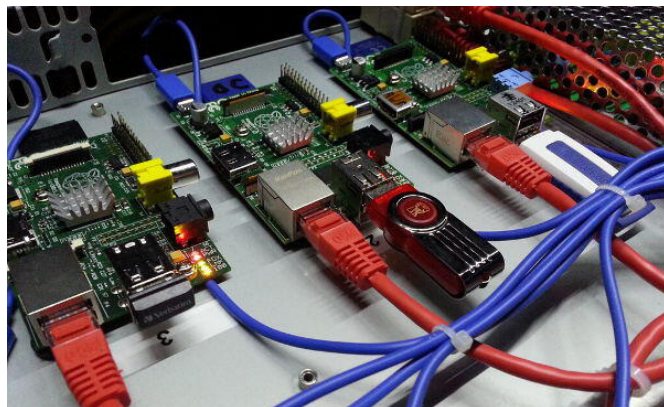


Figure 1. @kitestramrt's RPI

- @anlarye: I use my pi as an xbmc media center.
- @Bonbuzaru: #Rpi is our computer for a self-built RV/camper: logging GPS & environment data, media centre & soon leisure battery monitoring.



Figure 2. @_hamzux_'s RPI runs #owncloud and a #subversion server.

- @_hamzux_: my Raspberry Pi running #owncloud and a #subversion server #RPi (Figure 2).
- jerpi_bilbo: #RPi I am a Raspberry Pi Model B w/512MB. I like hiking, long walks along the beach, and wait...I'm a Linux/ARM target board.
- @josephbottinger: Raspberry Pi running a midi foot controller AKA Alcyone Beta (Figure 3).



Figure 3. @josephbottinger's RPI runs a midi foot controller AKA Alcyone Beta

- @shawnp0wers: My Raspberry Pi dangles naked behind my television, running RaspBMC!
- Submitted by bolt on LJ.com: My Raspberry Pi sits in the middle of the

living room, on a shelf, outputting sound to my stereo while looking like an out-of-place ancient SCSI CD-ROM. Controlled remotely with SSH from an Android phone (<http://blog.dhampir.no/content/cd-rom-raspberry-pi-case>).

Thanks for all the feedback folks! It's great to see how everyone uses their Raspberry Pis.—Ed.

SIGALRM Timers and Stdin Analysis

Regarding Dave Taylor's article "SIGALRM Timers and Stdin Analysis" in the November 2012 issue: here's a method that's a little tidier. I commented out the "tracing" prints. With more thought, it could be made to handle concurrent timed tasks:

```
#!/bin/bash

function allow_time
{
    (
        #echo timer allowing $1 seconds for execution
        sleep $1
        kill -ALRM $$
    ) &
    alarmPID=$!
    taskTimedOut=0
}
```

```
}

function timeout_handler
{
    #echo allowable time for execution exceeded.
    taskTimedOut=1
}

function timeTask
{
    trap timeout_handler SIGALRM
    allow_time $2
    #echo "alarmPID=$alarmPID"
    $1
    if [ $taskTimedOut -eq 0 ]; then
        kill -9 $alarmPID
        wait >/dev/null 2>&1
        true
        return
    else
        false
        return
    fi
}

echo "Finishes soon enough"
timeTask "sleep 2" 4 || (echo "task 1 took too long"; exit 1;)

echo "Takes too long"
timeTask "sleep 5" 4 || (echo "task 2 took too long"; exit 1;)
```

—Neal Murphy

Dave Taylor replies: Nice solution, Neal!

Linux in Education

To Shawn Powers: I happened upon your blog following a link in the last issue of *Linux Journal* (to which I subscribe)—the Gc script. I will give it a try, and let you know how it worked for me.

As a fellow Director of Technology of a school district, I imagine that you must be facing similar challenges when it comes to implementing and adopting Linux in your environment. I am a Microsoft-certified professional, a Mac lover and a Linux enthusiast (although I still consider myself a newbie), and as such, I want to give our students and faculty more choices when it comes to computer platforms. Last year, I converted some older PCs from Windows XP to Linux Mint, but the experience wasn't a totally positive one. Because of that, I am somewhat reluctant to try again or to expand Linux to more of our schools. The main issue was an inability to make the Linux experience an enjoyable one to the users and to make Linux "play nice" in an Active Directory environment (especially when it came to accessing and saving files to network shares). I wrote asking for help and advice from

the *Linux Journal* folks, but no one even acknowledged my messages.

If you are currently using Linux in your school district, I'd be happy to know which distribution works best for your students, and I'd appreciate any recommendations, tips or advice you might have for me. Also, perhaps you can convince your colleagues at *Linux Journal* to do a story about the use of Linux in K–12 education. I'm sure that it would inspire other IT professionals who might be struggling with the same issues as I am.

Thank you and looking forward to your insight.

—Lucian Micu

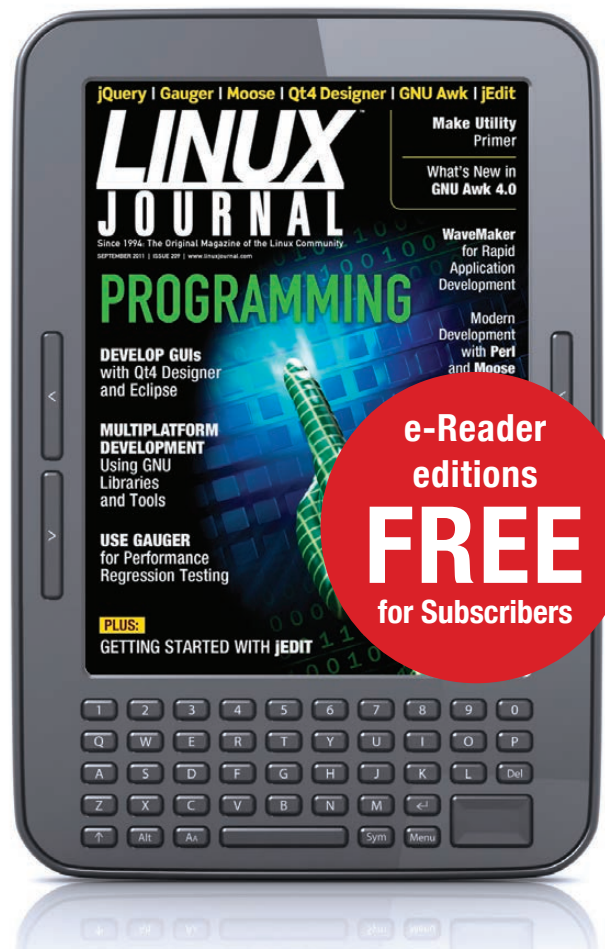
Indeed, it's difficult to bolster the usage of Linux in an environment bent on proprietary solutions. Through the years, I've used several distributions in our school, but for the past few years, I've had the most luck with Ubuntu or a variant. I didn't have an Active Directory environment, but I found that hosting files on a Linux server worked best for me. I could share via NFS with my Linux

LINUX JOURNAL

on your
e-Reader

Customized
Kindle and Nook
editions
now available

[LEARN MORE](#)



[LETTERS]

thin clients, and SMB with Windows and OS X. I even used netatalk to share with OS X sometimes. Because it was from a central Linux file server, I could share the same files over a variety of protocols. You may not have that option, which means lots more work and testing, unfortunately. Good luck, and keep fighting the good fight!—Shawn

Photo of the Month

It's always good to see Tux on my flights!

—Oscar Javier Bello Pérez



Tux on a Flight

WRITE LJ A LETTER We love hearing from our readers. Please send us your comments and feedback via <http://www.linuxjournal.com/contact>.

PHOTO OF THE MONTH

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

LINUX JOURNAL

At Your Service

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

ACCESSING THE DIGITAL ARCHIVE:

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

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

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

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

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

1&1 DYNAMIC CLOUD SERVER

PRICE CONTROL



AS LOW AS

\$0.06
PER HOUR*

✓ COMPLETE COST CONTROL

- **NEW: No setup fee!**
- **NEW: No base price!**
- **NEW: No minimum contract period!**
- **Full transparency** with accurate hourly billing
- **Parallels® Plesk Panel 11 included,** with unlimited domains

✓ FULL ROOT ACCESS

- The complete functionality of a root server with dedicated resources

✓ MAXIMUM FLEXIBILITY

- Configure the Processor Cores, RAM and Hard Disk Space
- Add up to 99 virtual machines

✓ FAIL-SAFE SECURITY

- Redundant storage and mirrored processing units reliably protect your server



DOMAINS | E-MAIL | WEB HOSTING | eCOMMERCE | SERVERS

Call **1 (877) 461-2631** or buy online

1and1.com

*Other terms and conditions may apply. Visit www.1and1.com for full promotional offer details. Customer is billed monthly for minimum configuration (\$0.06/ hour * 720 hours = \$43.20/ month minimum). Program and pricing specifications and availability subject to change without notice. 1&1 and the 1&1 logo are trademarks of 1&1 Internet, all other trademarks are the property of their respective owners. © 2013 1&1 Internet. All rights reserved.

diff -u

WHAT'S NEW IN KERNEL DEVELOPMENT

Remember the **SCO lawsuit**, claiming Linux used code from **UNIX System V**? Disproving that claim required a tremendous amount of work, partly because Linux had no revision control system to help track down the original authors of any given patch.

In response to that whole mess, **Linus Torvalds** implemented the **Developer's Certificate of Origin** (DCO) in 2004, which required people submitting a kernel patch to include a "Signed-off-by" header, identifying themselves and certifying that the code was legal to go into the kernel.

Luis R. Rodriguez recently said the DCO was so great, that a lot of other projects had started using it, but that in some cases, they weren't using it in a way that actually would help them track the origins of each patch. He wanted to extract the text of the DCO into its own git repository, so that other free software projects could access the explanatory text more easily and use it better.

This initially met with some resistance from people like **Alan**

Cox and **Jiry Slaby** who felt that any project wishing to use the DCO could extract it from the kernel sources without much of a problem. They saw no reason for a standalone git repository.

W. Trevor King disagreed. He felt a standalone git tree for the DCO would be a fine idea. But (ironically), he had some copyright questions about the DCO itself. He said that in the initial patch to add the text of the DCO to the kernel sources, Linus hadn't used a "Signed-off-by" header, so he couldn't be sure the DCO's explanatory text even was legal to release under the GPL version 2. That might seem like nitpicking, except that Trevor also pointed out that the **Open Source Development Labs** (OSDL) had put out a press release announcing the existence of the DCO and had specifically licensed the DCO under the Creative Commons Attribution-ShareAlike 2.5 License.

This could have turned out to be a thorny issue—except it didn't. No one spoke up to clarify the licensing

question on the linux-kernel mailing list. It went completely unanswered.

Trevor, however, didn't wait. He created a new git repository at <https://github.com/wking/signed-off-by>, exactly along the lines of what Luis originally had requested. It even preserved the commit history of the DCO, through to its current version of 1.1.

To address the licensing question himself, he said that the project was derived from the GPLv2 Linux kernel, so it also was released under that same license. But, he also said, "Because many projects that are not GPLv2 may still want to use the DCO/s-o-b approach, I've included an example CONTRIBUTING file (and CONTRIBUTING.md for GitHub) that are licensed under the very permissive Creative Commons CC0 1.0 Universal. Merge the 'contributing' or 'contributing-github' branch into your project and edit as you see fit."

Somehow I don't think anyone will start complaining about any leftover licensing issues that may remain concerning the DCO. Luis certainly won't. He immediately replied to Trevor, saying he'd already started using Trevor's repository in his own

free software projects.

A fight broke out recently on the linux-kernel mailing list between the **kvmtool** developers (primarily **Ingo Molnar** and **Pekka Enberg**) and Linus Torvalds. The issue was whether kvmtool should go into the official kernel tree.

kvmtool is used by the **KVM** (Kernel Virtual Machine) subsystem to boot up virtual computers under a running Linux system. It's awesome. Everyone should use it, whether they have a need for it or not. Create your own cluster of 20 systems without spending a dime.

There was a lot of heated back-and-forth discussion over what to do. Essentially, Ingo and Pekka argued that kvmtool should be included in the official kernel sources, because kvmtool was tightly integrated with kernel development. They relied on the same pool of developers and used the same mailing lists. Ports of the KVM subsystem to other architectures would have a functioning kvmtool immediately—in some cases before the KVM port even had been completed. Ingo and Pekka cited many examples of instances where the existence of kvmtool in the

kernel tree (really in the linux-next tree, waiting to go to Linus) had resulted in better, stronger, faster development overall. They argued that this constituted a real benefit that would be lost if kvmtool had to exist outside the kernel. Without that tight integration, development would be slower, and the kvmtool developers would have to implement all the infrastructure required by a standalone project.

Linus, however, was completely unmoved by any of these arguments. In fact, the whole idea of speeding up development and making things easier for the developers didn't register to him as even remotely relevant to whether a piece of code should go into the kernel.

Instead (which he said over and over again), he needed to have a solid technical reason why the code "belonged" in the kernel rather than staying outside. It had to make sense, not from his personal perspective or from any other developer's perspective, but from the kernel's perspective itself. Did kvmtool really belong in the kernel? And that question, he said, Ingo and Pekka had not even addressed.

Ultimately, he simply refused to merge the code, and the kvmtool developers probably will extract it from the kernel and create a standalone git repository and the rest of the necessary infrastructure. Or, they'll articulate better technical (and less developer-focused) reasons why kvmtool should be more tightly integrated.—**ZACK BROWN**

They Said It

I'd learned enough about circuitry in high school electronics to know how to drive a TV and get it to draw—shapes of characters and things.

—**Steve Wozniak**

I think it's very comforting for people to put me in a box. "Oh, she's a fluffy girlie girl who likes clothes and cupcakes. Oh, but wait, she is spending her weekends doing hardware electronics."

—**Marissa Mayer**

Other kids went out and beat each other up or played baseball, and I built electronics.

—**Robert Moog**

When I was a teenager in the late 1930s and early 1940s, electronics wasn't a word. You were interested in radio if you were interested in electronics.

—**Ken Olsen**

I met Woz when I was 13, at a friend's garage. He was about 18. He was, like, the first person I met who knew more electronics than I did at that point.

—**Steve Jobs**

Non-Linux FOSS: Seashore

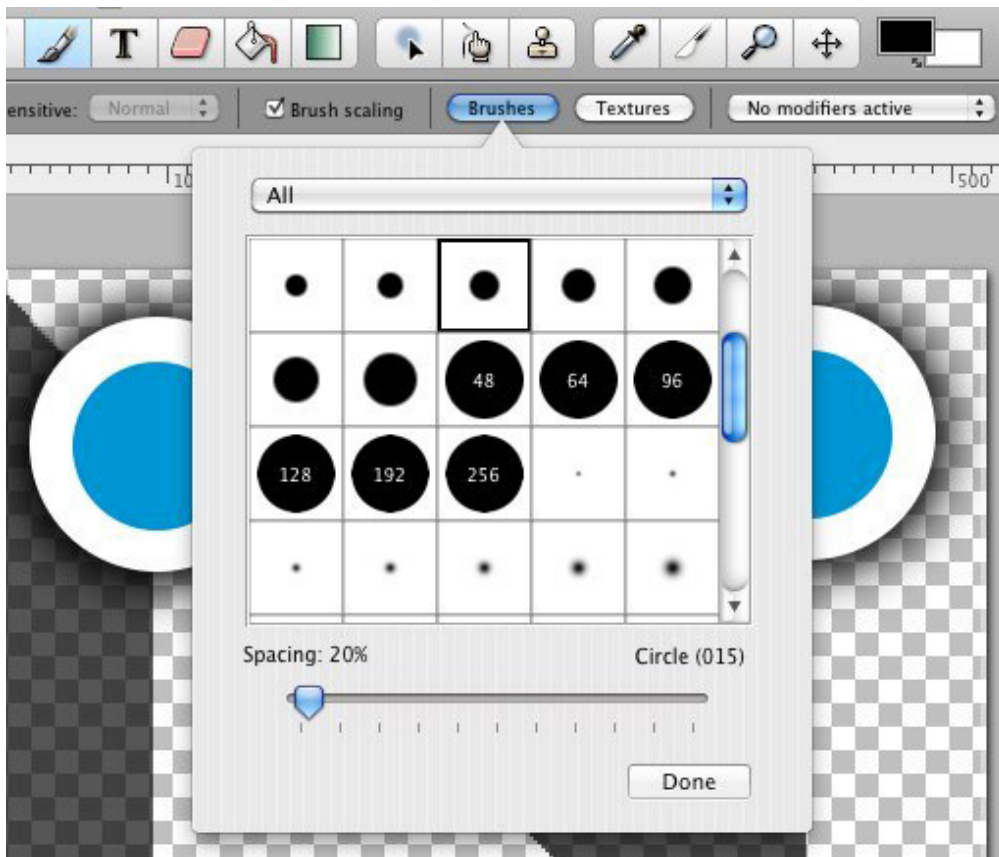


Figure 1. Based on the interface alone, it's obvious Seashore is a native OS X application.

As Linux users, we tend to take programs like GIMP for granted. Thankfully, as of version 2.8.2, GIMP is available as a native application for OS X! Because everyone reading this most likely is familiar with how awesome GIMP is for photo editing, it's worth mentioning there is another

open-source photo-editing application for OS X named Seashore.

Seashore is an application written in the native OS X Cocoa language, so it clearly looks like a native application when running (Figure 1). Although OS X users no longer have to install X11 in order to take advantage of

GIMP, Seashore is a powerful image editor in its own right, and because it's open source, it's well worth the effort to try it out! You can download Seashore at <http://seashore.sourceforge.net>. The native version of GIMP is available at <http://www.gimp.org/downloads>.—**SHAWN POWERS**

Raspberry Pi Poll

We asked LinuxJournal.com readers about their Raspberry Pis, and the results are below. We see that 11% indicated that they have multiple models, and 6% own more than 3 Pis, and at \$35 a piece, we can hardly blame you. Also fun to note is that 77% of respondents bought their Pis with no specific purpose in mind, just to experiment with, which is yet another fabulous side effect of the low price and wee form factor. Only 35% of our readers indicated that the Pi is not fast enough to use as a desktop machine, while the rest found it at least sufficient enough for basic tasks. Command-liners and GUI users are split down the middle with 50% each, which is not surprising considering the multitude of applications for a Pi, each of which lends itself to a different way of interacting. And finally, Raspian is the favorite operating system by far, but there are many to experiment with. So, please continue to enjoy your Pi, and we'll continue writing about new and fun ways to help you with that.

1) Do you own a Raspberry Pi?

- Yes: 83%
- No: 17%

2) If you own a Raspberry Pi, which model do you own?

- Model A: 5%
- Model B: 35%
- Updated Model B with 512MB of RAM: 49%
- Multiple models: 11%

3) What do you keep your Raspberry Pi units in?

- Computer case for RPi: 53%
- Custom-built case (Legos, Pop-Tart box and so on): 16%
- No case, it hangs in the wind: 31%

4) If you own a Raspberry Pi, did you pay extra for the MPEG2 license?

- Yes: 14%
- No: 86%

5) What potential model C feature is most important to you?

- USB boot: 37%
- PXE/Etherboot booting: 17%
- Power Over Ethernet (POE): 32%
- Other: 14%

6) Do you overclock your RPi?

- Yes: 31%
- No: 69%

7) What operating system is on your Raspberry Pi?

- Raspian: 63%
- RaspBMC: 9%
- OpenELEC: 6%
- Fedora Remix: 3%
- PiBang Linux: <1%
- Arch Linux: 9%
- SliTaz: <1%
- Android: 2%
- Other: 7%

8) Did you buy a Raspberry Pi...

- For a specific purpose: 23%
- To experiment with: 77%

9) Did you already own the required peripherals (SD card, Micro USB power adapter, HDMI cable)?

- Yes, I have several of each: 47%
- I had some and had to buy some: 45%
- No, and it more than doubled the cost: 8%

10) Is the Raspberry Pi fast enough for you to use as a desktop machine?

- Yes: 8%
- No: 35%
- For basic things, but not as my main computer: 57%

11) Do you use a GUI on your Raspberry Pi or CLI only?

- CLI only (server, etc.): 50%
- GUI (desktop or video playback): 50%


12) How many Raspberry Pis do you own?


- 1: 67%
- 2-3: 27%
- More than 3: 6%

Ultra Small Panel PC

PPC-E4+

- ARM9 400Mhz Fanless Processor
- Up to 1 GB Flash & 256 MB RAM
- 4.3" WQVGA 480 x 272 TFT LCD
- Analog Resistive Touchscreen
- 10/100 Base-T Ethernet
- 3 RS232 & 1 RS232/422/485 Port
- 1 USB 2.0 (High Speed) Host port
- 1 USB 2.0 (High Speed) OTG port
- 2 Micro SD Flash Card Sockets
- SPI & I2C, 4 ADC, Audio Beeper
- Battery Backed Real Time Clock
- Operating Voltage: 5V DC or 8 to 35V DC
- Optional Power Over Ethernet (POE)
- Optional Audio with Line-in/out
- Pricing starts at \$375 for Qty 1





2.6 KERNEL

The PPC-E4+ is an ultra compact Panel PC that comes ready to run with the Operating System fully configured on the onboard flash. The dimensions of the PPC-E4+ are 4.8" by 3.0", about the same as that of popular touch cell phones. The PPC-E4+ is small enough to fit in a 2U rack enclosure. Apply power and watch either the Linux X Windows or the Windows CE User Interface appear on a vivid 4.3" color LCD. Interact with the PPC-E4+ using the responsive integrated touch-screen. Everything works out of the box, allowing you to concentrate on your application rather than building and configuring device drivers. Just Write-It and Run-It.

www.emacinc.com/panel_pc/ppc_e4+.htm

Since 1985
OVER
28
YEARS OF
SINGLE BOARD
SOLUTIONS

EMAC, inc.

EQUIPMENT MONITOR AND CONTROL

Phone: (618) 529-4525 • Fax: (618) 457-0110 • Web: www.emacinc.com

Designing Electronics with Linux

In many scientific disciplines, the research you may be doing is completely new. It may be so new that there isn't even any instrumentation available to make your experimental measurements. In those cases, you have no choice but to design and build your own measuring devices. Although you could build them using trial and error, having a way to model them first to see how they will behave is a much better choice—in steps oregano

(<https://github.com/marc-lorber/oregano>). With oregano, you can design your circuitry ahead of time and run simulations on it to iron out any problems you may encounter.

The first step, as always, is installing the software. Most distributions should have a package for oregano available. If you want to follow the source version, it is available at GitHub. Oregano also needs another software package to handle the actual simulation. The two

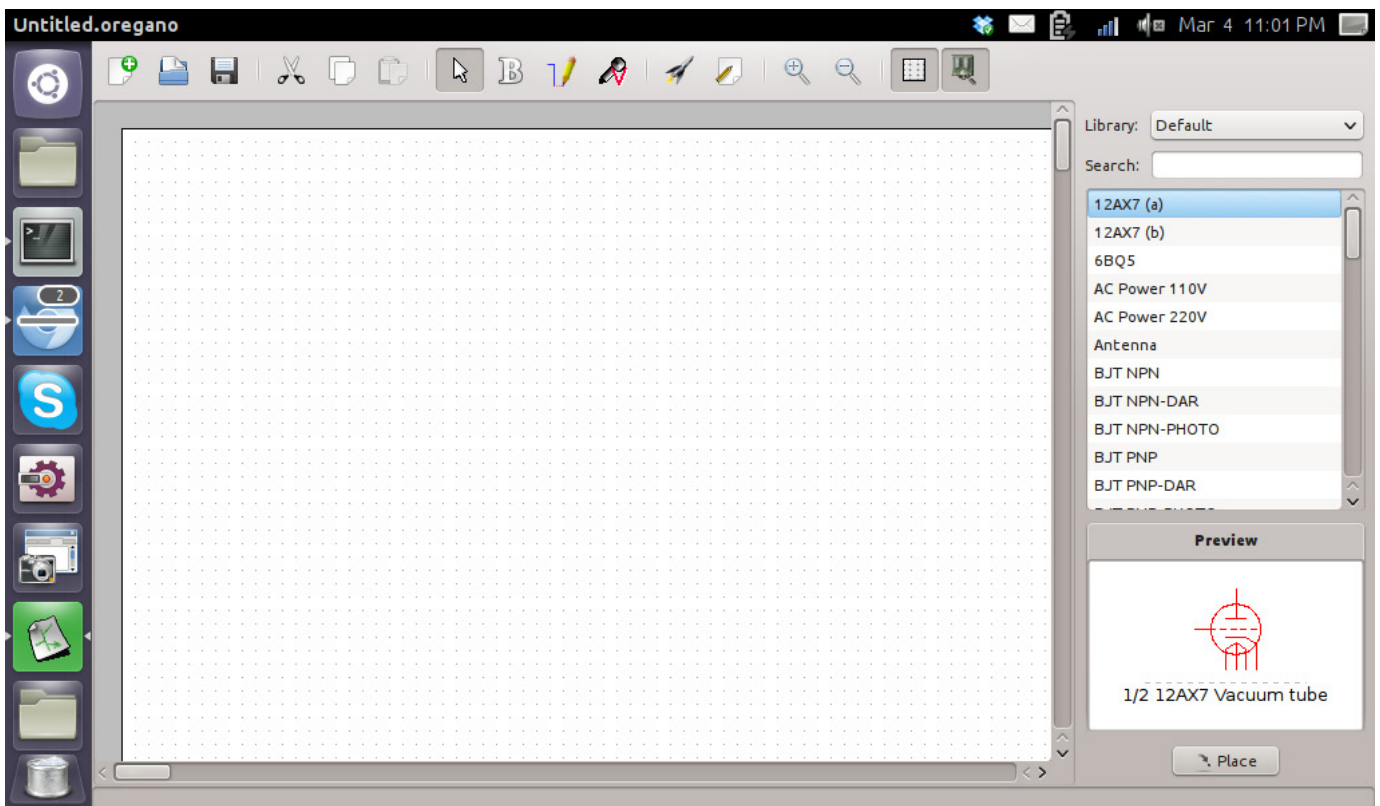


Figure 1. On startup, you get a blank canvas and a parts list.

packages it currently can work with are GnuCap and ngspice. Either of these two packages needs to be installed in order to do the calculations for the simulation. While this is handled automatically by your distribution's package manager, you will need to install this dependency yourself if you are building from source.

Once it's installed, you will get a blank new project when you first start up oregano (Figure 1). On the right-hand side, you should see a list of elements you can use to build your circuits. It starts up with the default library selected. This library provides all the standard electronic components you likely will want to use. But, this isn't the

only library included. You can select from other libraries, such as TTL, Linear, CPU or Power Devices, among others.

Each of these libraries contains a list of associated elements you can use in your circuits. Selecting one of the elements shows a preview of the schematic drawing of that element in the bottom window. You then can drag and drop the element onto your canvas and start building your circuit. Once you have an element on the canvas, you can double-click the element to edit its properties (Figure 2). You need to click on the "Draw wires" icon at the top of the window in order to connect the elements together into a proper circuit.

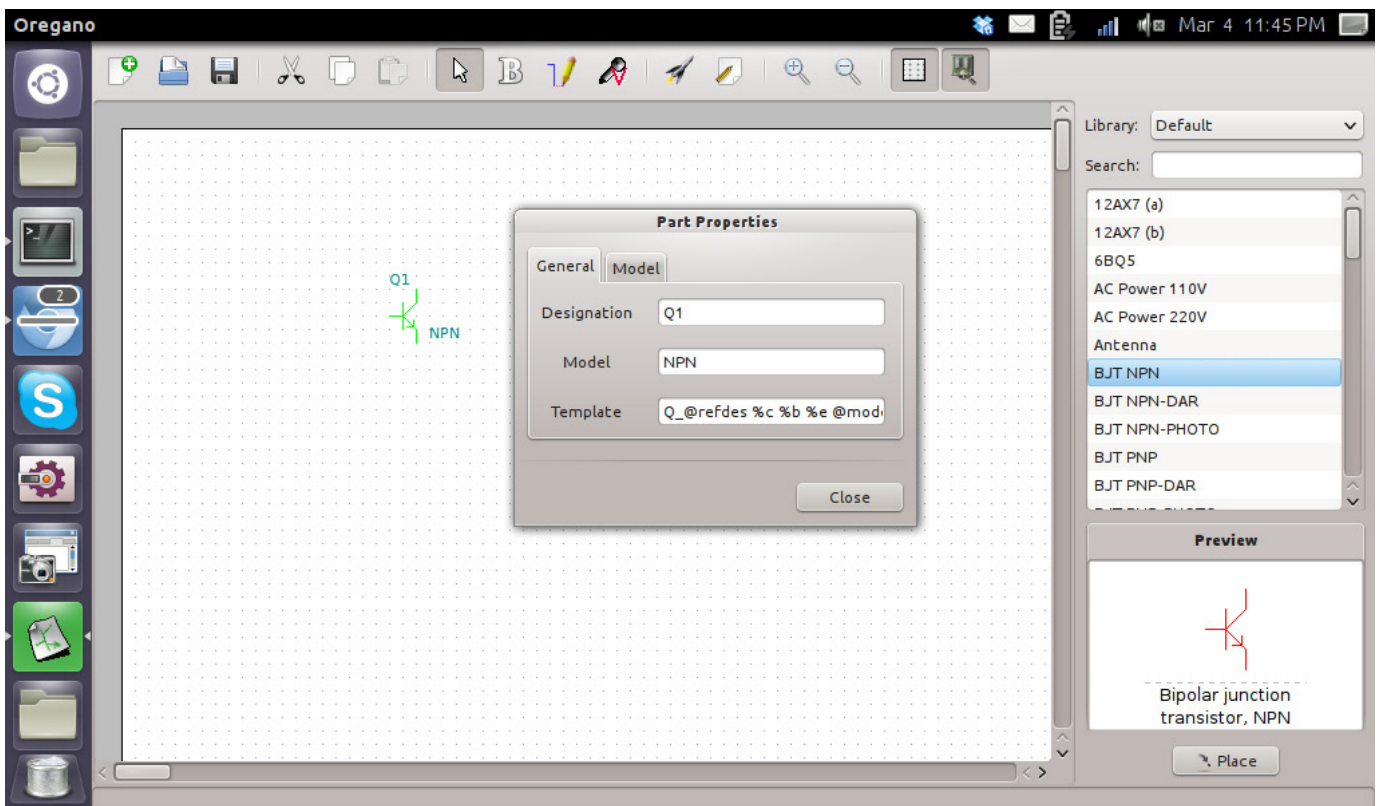


Figure 2. The property window depends on which properties are available for that element.

[UPFRONT]

All circuits have some necessary components to make an actually functioning circuit. The first of these is the ground. This element is labeled GND in the default library. Along with ground, you need some sort of power source. In most cases, you will want some form of DC current. This is provided by the element labeled with VDC in the default library. With those two important elements in your circuit, you can go ahead and wire up the rest of the circuit.

Once you have a circuit made up, you will want to run a simulation to see how it behaves. Because of the nature of electrical circuits, you need to put sensors into the circuit to see

its behavior. You can click on the “Add voltage clamp” icon at the top of the window to select the sensor object. Then, you can click on the areas of your circuit where you want to measure during the simulation. At each point you click, you will see a new icon on your circuit marking a sensor location. Double-clicking on the clamp will pop up a window where you can set the parameters of what is being measured (Figure 3). You need at least one clamp in your circuit before you can run a simulation; otherwise, you won’t have any measurements to study in your simulation.

Once you have all of your clamp points

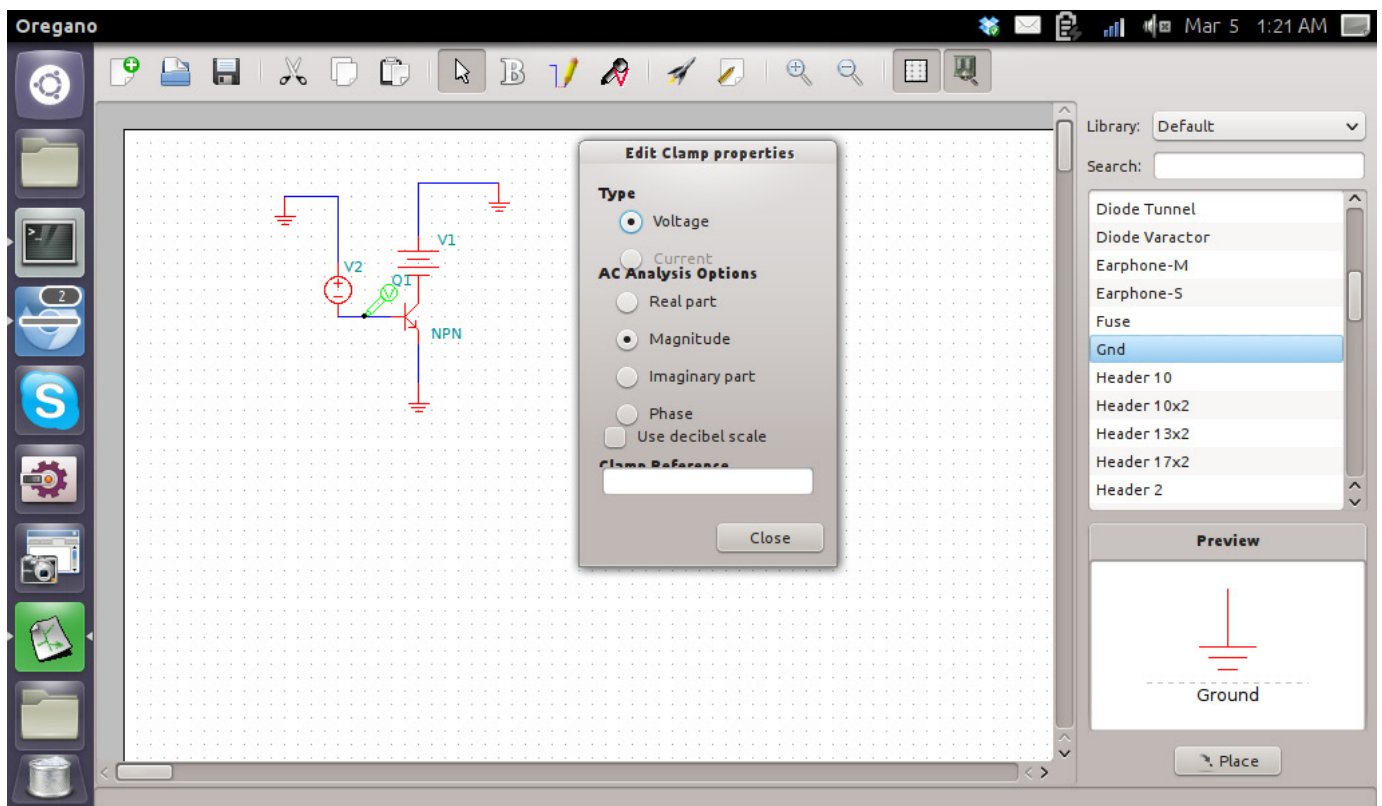


Figure 3. Here you can select the properties for the clamp.

selected, you can run the simulation and see what happens by clicking on the “Run a simulation” icon at the top of the window (Figure 4). When you do so, oregano opens a new window where you can see a plot of the data registered by the clamp (usually voltage or current).

When you do analysis, you have the choice of two different circuit analysis programs: Gnucap and spice. On Ubuntu, the default analysis program that is installed as a dependency is Gnucap. This means you need to install spice explicitly if you want to use it instead.

To select the analysis engine, click on Edit→Preferences. In this dialog, you also can set whether the log window

will open automatically when needed, and you can set the data paths for the models and libraries that will be available for your circuits. In most cases, you will want to leave those as they are.

To help you get started, oregano comes with several examples. Again, on Ubuntu (since that is my current desktop), these examples are located in /usr/share/doc/oregano/examples. You might want to load one of these examples first.

Once you have a completed circuit and want to run a simulation, you will need to set parameters to control this simulation. Click on the menu item Edit→Simulation Settings to bring up

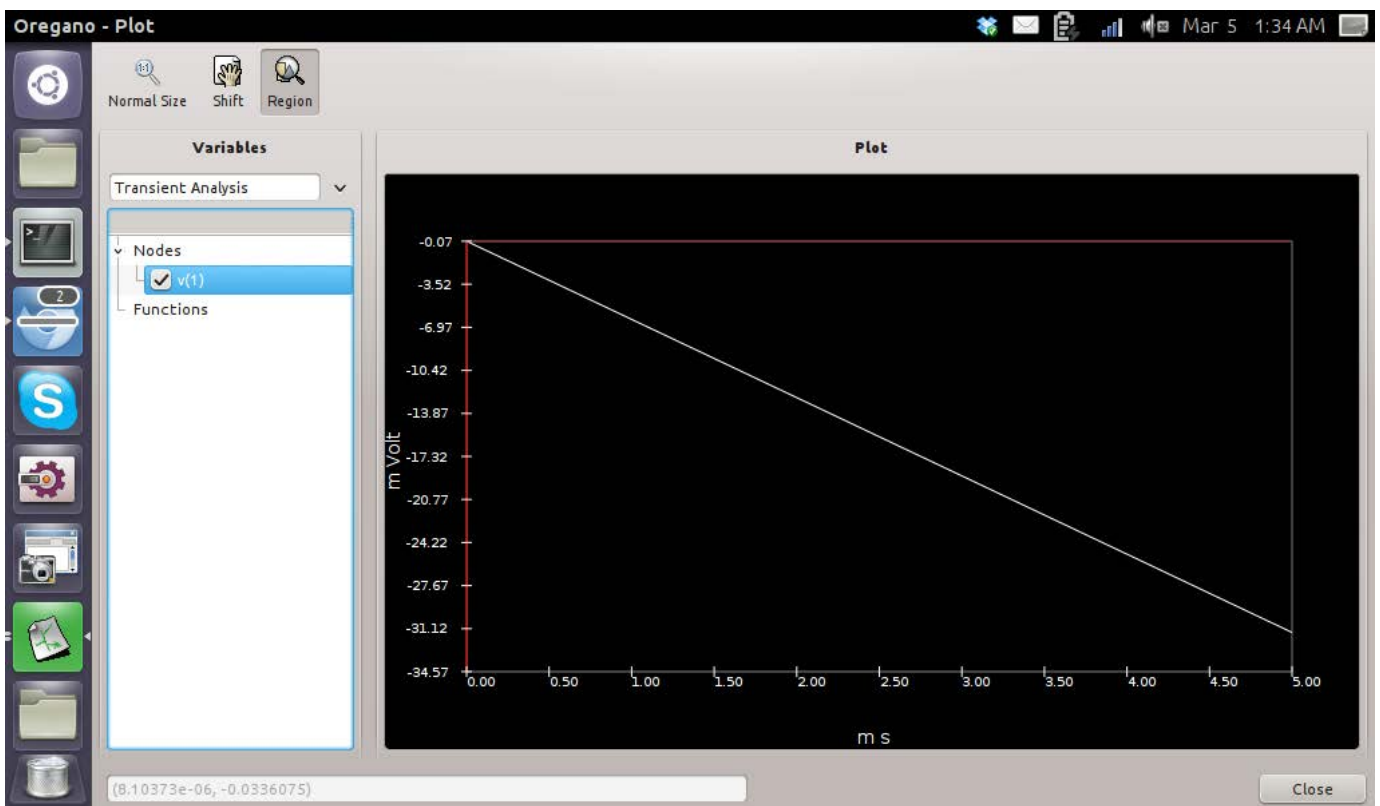


Figure 4. Plotting the results of a circuit clamp.

[UPFRONT]

the dialog window. The first tab lets you see analysis parameters, such as transient options, fourier options, DC sweep options and AC options. Clicking on any of the check boxes will open up a subset of further options for each of those sections. The second tab lets you set a series of analysis options. You also can set parameters that may affect your circuit, such as the ambient temperature.

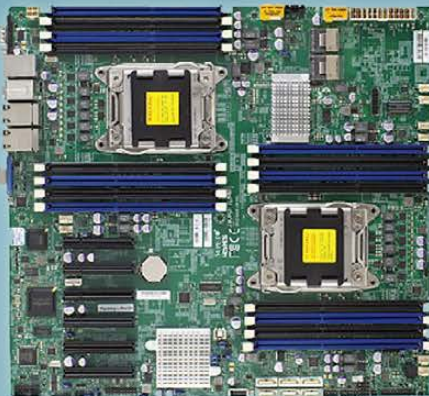
Once you have all of the options and parameters set, you can start the simulation by selecting the menu item Tools→Simulate or by pressing F11. Don't forget to attach some test clamps

first; otherwise, you will get an error. Your simulation will run and pop up a new plot window where you can look at the values generated within your circuit. You can select whether to look at the transient analysis or AC analysis.

On the left, you will see a list of available plotting options. On the right-hand side, you will find the actual plot of your data. Only the items that you select from list will be plotted, so that means when this window first opens, nothing actually will be plotted.

You also can plot functions of the available values. For example, you could

ENTERPRISE Big Data Storage



(Rear View)



plot the difference in voltage between two separate test clamps. These functions will be available in the list on the left side, so you can select them and have them plotted on the graph.

Additionally, you can include even more complicated elements like full CPUs to your circuit. The problem with these is that the way they respond to electrical signals can be very complicated. These elements need a separate model file that describes this response to signals. Unfortunately, the licensing for model files means that many cannot be included with oregano. You can search

the Internet and download the model files for the elements that interest you, or you can create your own model file. In either case, you can place the model files into the directory set in the preferences.

When you actually want to build your circuit, you can export the associated diagram by clicking on the menu item File→Export. You then can export the circuit diagram as either an SVG file, a PDF, a PostScript file or a PNG. Now, you can go ahead and build your new test equipment, secure in the knowledge that you did some initial testing and should get the behavior you need.—**JOEY BERNARD**

KSC

KING STAR COMPUTER
Rackmount Server Specialist

Dual Socket System, Support the latest Single or Dual Intel Xeon Processor E5-2600 product family

Up to 512GB DDR3 1600MHz ECC Registered DIMM; 16x DIMM Sockets

Intel i350 Quad Gigabit Ethernet Ports

72 x 3.5" HDDs, Supports up to 288TB in 4U chassis

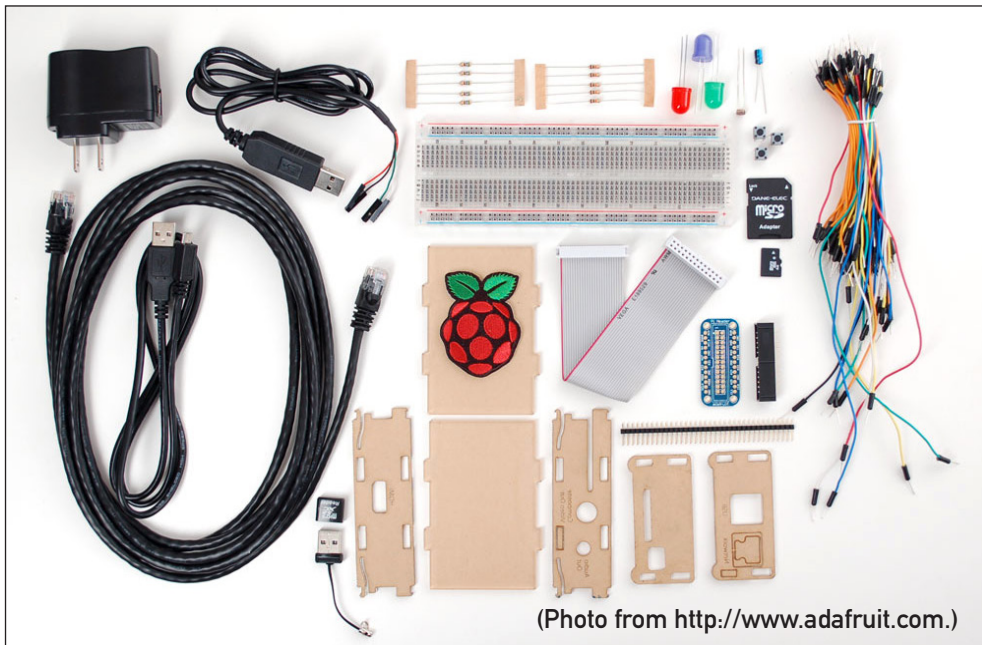
IT Mode or Hardware RAID providing reliable RAID levels 0, 1, 5, 6, 10, 50, 60

Two 2.5" Internal OS Disks

SAS Expander Technology

Redundant Platinum Level (Up to 95%) Digital Power Supplies

Add More Fruit to Your Raspberry Pi!



(Photo from <http://www.adafruit.com>.)

The Raspberry Pi can be a building block to nerdery!

Since this is our Raspberry Pi issue, I did some research on “what folks do with their Raspberry Pi”. I sent out queries via Twitter, Facebook, the *Linux Journal* Web site and even the #linuxjournal IRC room. When it comes to doing extra-geeky projects with the RPi, every person I spoke with mentioned buying parts from Adafruit.

Anybody familiar with Raspberry Pi development most likely has done business with Adafruit. I had the pleasure of speaking with Limor Fried, owner of Adafruit Industries regarding its most popular products. Although I was expecting the most common

Raspberry Pi-related sales to be cases or SD cards, as it happens, the list of most-popular products was much geekier!

While the low-profile microSD card adapter is popular, Limor told me the breakout boards and breadboard PCB kits are the most-popular items. This news both surprised and fascinated me. Although I’ve certainly done some cool things with RPi, the notion that folks have been doing elaborate and complex projects based on the Raspberry Pi is awesome! (A special thanks to Limor at Adafruit for talking to me about what folks are doing with their Raspberry Pi devices.)—**SHAWN POWERS**

AnDevCon

The Android Developer Conference

BOSTON • May 28-31, 2013

The Westin Boston Waterfront

Register Now
and SAVE!

Get the best real-world Android developer training anywhere!

- Choose from more than 75 classes and tutorials
- Network with speakers and other Android developers
- Check out more than 40 exhibiting companies

"AnDevCon is one of the best networking and information hubs available to Android developers."

—Nate Vogt, Android Developer,
Willow Tree Apps



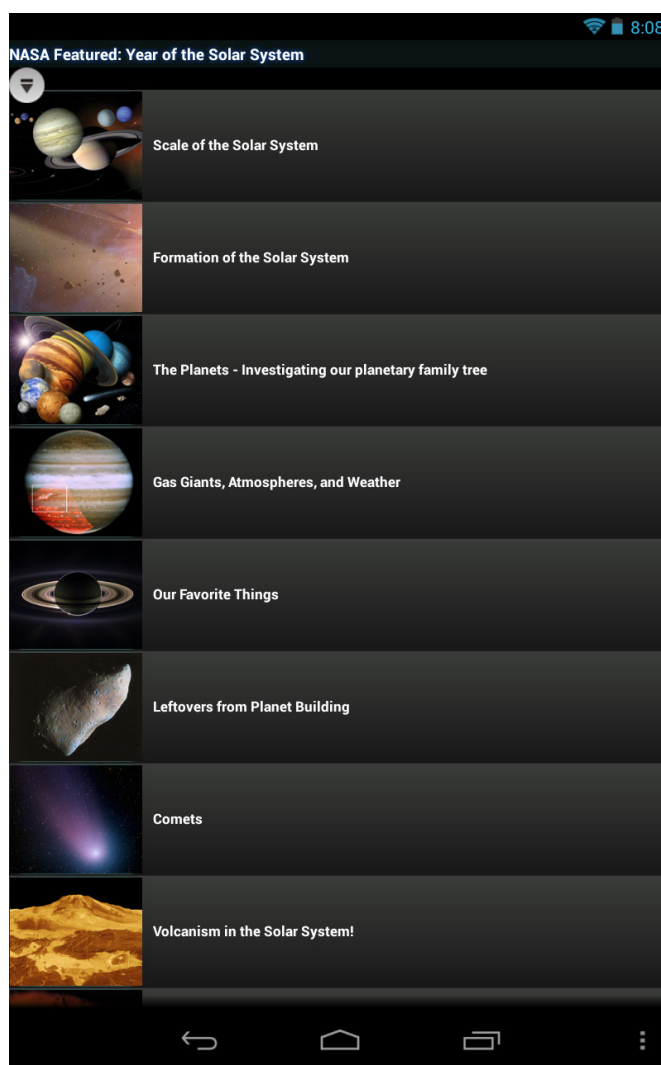
Register NOW at www.AnDevCon.com

Android Candy: Outer Space, in Your Pocket



When the Space Shuttle program shut down, I have to admit, it deflated my excitement about space exploration just a bit. Although it's not fair to pin the future of manned missions to space on a fleet of aging crafts built in the 1980s, the Space Shuttle represented the latest step in a process that would get us to other planets. Because there wasn't anything immediately replacing the Shuttle, it felt like we were giving up on space. (Some might argue we gave up on space after landing on the moon, but that's an entirely different discussion.)

Then the Mars *Curiosity* rover arrived at Mars, and I remembered how much I love science! My daughter stayed up late with me, and we passed a jar of peanuts back and forth while we chewed our nails watching NASA TV. The seven minutes of hell ended, and after a painfully long wait, we heard from *Curiosity*! My excitement was palpable, and that moment sparked a new generation of space geek in my daughter. The first



thing she did was look on her phone for NASA information, and I'm glad she did.

If you haven't looked at the NASA Android app in a while, you owe it to yourself to check it out. The last time

I tried a NASA app, it was a simple streaming app that did a poor job of streaming the NASA TV video. It crashed. It was ugly. It did not inspire awe or excitement. The current NASA app is amazing! Not only does it have high-definition streaming video (which doesn't seem to crash), it also has information on current and upcoming missions, a launch calendar, categorized information on moons and planets, and tons of other space-nerd fodder. It even will change the background photo on your Android device to the Astronomy Pic of the Day.

I think my favorite feature of the NASA app is its ability to grab your GPS location and tell you exactly where and when to spot things like the International Space Station! If you're a space nerd like I am, I urge you to grab your jar of peanuts and let the NASA app touchdown on your Android device (<https://play.google.com/store/apps/details?id=gov.nasa>). It's so much better than the last time I tried it, several years ago, that the NASA app for Android is this month's Editors' Choice.

—SHAWN POWERS

Powerful: Rhino



Rhino M4700/M6700

- Dell Precision M4700/M6700 w/ Core i7 Quad (8 core)
- 15.6"-17.3" FHD LED w/ X@1920x1080
- NVidia Quadro K5000M
- 750 GB - 1 TB hard drive
- Up to 32 GB RAM (1866 MHz)
- DVD±RW or Blu-ray
- 802.11a/b/g/n
- Starts at \$1190
- E6230, E6330, E6430, E6530 also available

- High performance NVidia 3-D on an FHD RGB/LED
- High performance Core i7 Quad CPUs, 32 GB RAM
- Ultimate configurability — choose your laptop's features
- One year Linux tech support — phone and email
- Three year manufacturer's on-site warranty
- Choice of pre-installed Linux distribution:



Tablet: Raven



Raven X230/X230 Tablet

- ThinkPad X230/X230 tablet by Lenovo
- 12.1" HD LED w/ X@1366x768
- 2.6-2.9 GHz Core i7
- Up to 16 GB RAM
- 750 GB hard drive / 180 GB SSD
- Pen/finger input to screen, rotation
- Starts at \$2050
- T430, T530, W530 also available

Rugged: Tarantula



Tarantula CF-31

- Panasonic Toughbook CF-31
- Fully rugged MIL-SPEC-810G tested: drops, dust, moisture & more
- 13.1" XGA TouchScreen
- 2.4-2.53 GHz Core i5
- Up to 8 GB RAM
- 320-750 GB hard drive / 512 GB SSD
- CF-19, CF-52, CF-H2 also available

EmperorLinux

...where Linux & laptops converge

www.EmperorLinux.com

1-888-651-6686





REUVEN M.
LERNER

Sidekiq

Want to speed up your Web application? Putting some tasks into the background is likely to help.

From my perspective, one of the best parts of being a Web developer is the instant gratification. You write some code, and within minutes, it can be used by people around the world, all accessing your server via a Web browser. The rapidity with which you can go from an idea to development to deployment to actual users benefiting from (and reacting to) your work is, in my experience, highly motivating.

Users also enjoy the speed with which new developments are deployed. In the world of Web applications, users no longer need to consider, download or install the “latest version” of a program; when they load a page into their browser, they automatically get the latest version. Indeed, users have come to expect that new features will be rolled out on a regular basis. A Web application that fails to change and improve over time quickly will lose ground in users’ eyes.

Another factor that users consider is the speed with which a Web

application responds to their clicks. We are increasingly spoiled by the likes of Amazon and Google, which not only have many thousands of servers at their disposal, but which also tune their applications and servers for the maximum possible response time. We measure the speed of our Web applications in milliseconds, not in seconds, and in just the past few years, we have reached the point when taking even one second to respond to a user is increasingly unacceptable.

The drive to provide ever-greater speed to users has led to a number of techniques that reduce the delays they encounter. One of the simplest is that of a delayed job. Instead of trying to do everything within the span of a single user request, put some of it aside until later.

For example, let’s say you are working on a Web application that implements an address book and calendar. If a user asks to see all of his or her appointments for the coming week, you almost certainly

could display them right away. But if a user asks to see all appointments during the coming year, it might take some time to retrieve that from the database, format it into HTML and then send it to the user's browser.

One solution is to break the problem into two or more parts. Rather than having the Web application render the entire response together, including the list of appointments during the coming year, you can return an HTML page without any appointment. However, that page can include a snippet of JavaScript that, after the page is loaded, sends a request to the server asking for the list. In this way, you can render the outline of the page, filling it with data as it comes in.

Sometimes, you can't divide jobs in this way. For example, let's say that when you add a new appointment to your calendar, you would like the system to send e-mail to each of the participants, indicating that they should add the meeting to their calendars. Sending e-mail doesn't take a long time, but it does require some effort on the part of the server. If you have to send e-mail to a large number of users, the wait might be intolerably long—or just annoyingly long, depending on your users and the task at hand.

Thus, for several years, developers have taken advantage of various "delayed jobs" mechanisms, making it possible to say, "Yes, I want to execute this functionality, but later on, in a separate thread or process from the handling of an HTTP request." Delaying the job like this may well mean that it'll take longer for the work to be completed. But, no one will mind if the e-mail takes an additional 30 seconds to be sent. Users certainly will mind, by contrast, if it takes an additional 30 seconds to send an HTTP response to the users' browser. And in the world of the Web, users probably will not complain, but rather move on to another site.

This month, I explore the use of delayed jobs, taking a particular look at Sidekiq, a Ruby gem (and accompanying server) written by Mike Perham that provides this functionality using a different approach from some of its predecessors. If you're like me, you'll find that using background jobs is so natural and easy, it quickly becomes part of your day-to-day toolbox for creating Web applications—whether you're sending many e-mail messages, converting files from one format to another or producing large reports that may take time to process, background jobs are a great addition to your arsenal.

Background Queues

Before looking at Sidekiq in particular, let's consider what is necessary for background jobs to work, at least in an object-oriented language like Ruby. The basic idea is that you create a class with a single method, called "perform", that does what you want to execute. For example, you could do something like this:

```
class MailSender
  def perform(user)
    send_mail_to_user(user)
  end
end
```

Assuming that the `send_mail_to_user` method has been defined in your system, you can send e-mail with something like:

```
MailSender.new.perform(user)
```

But here's the thing: you won't ever actually execute that code. Indeed, you won't ever create an instance of `MailSender` directly. Rather, you'll invoke a class method, something like this:

```
MailSender.perform_async(user)
```

Notice the difference. Here, the class method takes the parameter that you eventually want to be passed

to the "perform" method. But the "perform_async" class method instead stores the request on a queue. At some point in the future, a separate process (or thread) will review method calls that have been stored in the queue, executing them one by one, separately and without any connection to the HTTP requests.

Now, the first place you might consider queuing method classes that you'll want to execute would be in the database. Most modern Web applications use a database of some sort, and that would be a natural first thought. And indeed, in the Ruby world, there have been such gems as "delayed job" and "background job" that do indeed use the database as a queue.

The big problem with this technique, however, is that the queue doesn't need all of the functionality that a database can provide. You can get away with something smaller and lighter, without all the transactional and data-safety features. A second reason not to use the database is to split the load. If your Web application is working hard, you'll probably want to let the database be owned and used by the Web application, without distracting it with your queue.

So, it has become popular to use non-relational databases, aka NoSQL solutions, as queues for background

jobs. One particularly popular choice is Redis, the super-fast, packed-with-functionality NoSQL store that works something like a souped-up memcached. The first job queue to use Redis in the Ruby world was Resque, which continues to be popular and effective.

But as applications have grown in size and scope, so too have the requirements for performance. Resque is certainly good enough for most purposes, but Sidekiq attempts to go one better. It also uses Redis as a back-end store, and it even uses the same storage format as Resque, so that you either can share a Redis instance between Resque and Sidekiq or transition from one to the other easily. But, the big difference is that Sidekiq uses threads, whereas Resque uses processes.

Threads? In Ruby?

Threading in Ruby is something of a sore subject. On the one hand, threads in Ruby are super-easy to work with. If you want to execute something in a thread, you just create a new Thread object, handing it a block containing the code you want to execute:

```
Thread.new do
  STDERR.puts "Hello!" # runs in a new thread
end
```

The problem is that people who come from languages like Java often are surprised to hear that although Ruby threads are full-fledged system threads, they also have a global interpreter lock (GIL), which prevents more than one thread from executing at a time. This means that if you spawn 20 threads, you will indeed have 20 threads, but the GIL acts as a big mutex, ensuring that no more than one thread is executing simultaneously. Thread execution typically switches for I/O, and given that nearly every program uses I/O on a regular basis, this almost ensures that each thread will be given a chance to execute.

I should note that Ruby isn't the only language with these issues. Python also has a GIL, and Guido van Rossum, Python's creator, has indicated that although he certainly wants Python to support threading, he personally prefers the ease and security of processes. Because processes don't share state, they are less prone to difficult-to-debug problems, without sacrificing too much in execution speed.

Sidekiq is threaded, but it uses a different model for threads than most Rubyists are used to. It uses Celluloid, an "actor-based" threading system that packages the threads inside objects, avoiding most or all

Moreover, Celluloid expects to run in JRuby or Rubinius, two alternative Ruby implementations, which have true threading and lack the GIL.

of the issues associated with threads. Moreover, Celluloid expects to run in JRuby or Rubinius, two alternative Ruby implementations, which have true threading and lack the GIL.

Celluloid-based applications, such as Sidekiq, will work just fine under the standard Ruby interpreter, known as the MRI, but you won't enjoy all of the speed or threading benefits.

Using Sidekiq

Now, let's see how this overview of a delayed job can be implemented in Sidekiq. First and foremost, you'll need to install the Redis NoSQL store. Redis is available from a variety of sources; I was able to install it on my Ubuntu-based machine with:

```
apt-get install redis # check this
```

Once Redis is installed, you'll want to install the "sidekiq" gem. Again, it'll give you the best functionality if you run it under JRuby or Rubinius, but you can run it under the standard Ruby interpreter as well. Just realize that the threads will give you

non-optimal performance. You can install the gem with:

```
sudo gem install sidekiq -V
```

If you're running the Ruby Version Manager (RVM), as I do, you don't want to install the gem as root. Instead, you should just type:

```
gem install sidekiq -V
```

(I always like to use the -V flag, so I can see the details of the gem as it is installed.)

You can use Sidekiq in any Ruby application. However, most of my work is in Rails, and I imagine you're going to want to use it in Rails, Sinatra or a similar Web application. Thus, let's create a simple Rails application so you can try it:

```
rails new appointments
```

Within the new "appointments" directory, you'll then create an "appointment" resource with scaffolding—a combination of model,

controller and view that can get you going quickly:

```
rails g scaffold appointment name:text
  ↳meeting_at:timestamp notes:text
```

Once that is done, you have to run the migration, creating the appropriate “appointments” table in your database. In this case, because you didn’t specify a database, you’ll use SQLite, which is good enough for this toy example.

Now you can fire up your application (`rails s`) and go to `/appointments`. From that URL, you can create, view, edit and delete appointments. However, the point is not to create appointments, but rather delay the execution of something having to do with them. Let’s do something extremely simple, such as sending e-mail:

```
rails g mailer notifications
```

Inside `app/mailers/notifications.rb`, add the following method:

```
def appointment_info(person, appointment)
  @person = person
  @appointment = appointment
  mail(to:person.email, subject:"Appointment update")
end
end
```

And, inside `app/views/notifications/appointment_info.html.erb`, write the following:

```
<p>Hello! You have an appointment with <%= @person %>
at <%= @appointment.meeting_at %>.</p>
```

Finally, let’s tie it all together, sending your notification from within your `AppointmentWorker` class. There’s no rule for where the file defining such a class needs to be put, but it seems increasingly standard to have it in `app/workers`, in part because files under `app` are all loaded when Rails starts up:

```
class AppointmentWorker
  include Sidekiq::Worker

  def perform(appointment)
    Notifications.deliver_appointment_info(appointment)
  end
end
```

Notice several things here. First, the class doesn’t inherit from anything special. Sidekiq doesn’t use inheritance, but rather has you include a module—a class without instances, in Ruby—whose methods then are available to instances of your class. This is how the `perform_async` method is defined on your class. Through a little bit of

Sidekiq uses Ruby's built-in serialization facility to store nearly any sort of object, not just numeric IDs.

magic, importing a module can define both class and instance methods.

Now all you have to do is change your controller, such that after you create a report, you also send a notification:

```
AppointmentWorker.perform_async(@appointment)
```

Notice that you're not passing the ID of the appointment, but the appointment object itself! Sidekiq uses Ruby's built-in serialization facility to store nearly any sort of object, not just numeric IDs. The object and method call are stored in Redis, until they are retrieved by a Sidekiq process.

Indeed, that's the final part of Sidekiq that you need to get in place: the back-end process that will look through the delayed jobs, running each one in turn as it gets to them. Fortunately, running that is as easy as:

```
bundle exec sidekiq
```

Yup, that's all you need to do.

True, there are some options you can set, but generally speaking, this starts up a Sidekiq server that looks at the current queue (as stored in Redis), grabs a job off the queue and processes it. You can configure Sidekiq to run with a timeout or with a specified number of retries, and you even can say how many concurrent workers (that is, threads) you want to be working simultaneously.

Remember that although these are indeed threads, Sidekiq (via Celluloid) ensures that they don't have any state in common. Of course, you need to be sure that your worker methods are thread-safe, such that even if a worker gets through 90% of its job and is then halted, it'll be able to restart without any penalties or errors. Thus, your processes must be transactional, just as you would expect from a database query.

There are other ways to schedule Sidekiq jobs, besides defining methods within a module, as in the above example. If there's an existing

method that you want to run as a background process, just insert the “delay” method before the actual method call—that is:

```
my_object.delay.do_something_big
```

If you are using Rails and the built-in ActiveSupport module for easy time descriptions, you even can do something like this:

```
my_object.delay_for(5.days).do_something_big
```

Conclusion

Sidekiq has become quite popular in the Ruby community since it was released, in no small part because of its high performance, easy installation and ease of use. It also works with commercial hosting

services, such as Heroku, assuming that you first install a Redis instance.

Working with delayed jobs changes your perspective of the Web somewhat—you realize that not everything needs to take place immediately. Rather, you can delay certain jobs, putting them in the background, allowing your Web server to respond to users faster than otherwise would be the case. And, when speed is such a crucial element of Web application success, prudent use of Sidekiq likely will make a big difference. ■

Web developer, trainer and consultant Reuven M. Lerner is finishing his PhD in Learning Sciences at Northwestern University. He lives in Modi'in, Israel, with his wife and three children. You can read more about him at <http://lerner.co.il>, or contact him at reuven@lerner.co.il.

Resources

The Sidekiq home page is at <http://sidekiq.org>. Although Sidekiq.org does point to a commercial version, the basic version is still free and open source, with the source code available on GitHub at <http://mperham.github.com/sidekiq>, including a Wiki containing a great deal of useful information.

Mike Perham, the author of Sidekiq, describes the actor-based model in a blog post: <http://blog.carbonfive.com/2011/04/19/concurrency-with-actors>.

Finally, given that Sidekiq uses Redis, you likely will want to read more about this high-performance NoSQL database at <http://redis.io>.



DAVE TAYLOR

Summing Up Points

Fifteens. Why are they so important to *Cribbage*, and how do you calculate them? And what about pairs? Read on as Dave continues to step through the construction of a *Cribbage* game written as a shell script.

We're still building out the *Cribbage* game with the six-choose-four challenge that's at the very beginning of the game when the players in a two-player game discard two of their six cards into the "crib", a third hand that alternates between players. Think of it like playing draw poker, except when the players discard their cards, the dealer also could pick them up and play them as a second hand. That'd be weird, but interesting, wouldn't it? Hmmmm....

No, let's stay focused!

In my last article, we left the script at a point where it was able to pull out all two-card, three-card and four-card combinations of cards to ascertain which of them add up to 15 or otherwise offer point opportunities. Now let's jump in and actually calculate values and see what happens.

Specifically, here's where we left off:

```
$ cribbage.sh
Hand: AD, AS, 2D, 3C, 5C, KC.
Subhand 0: AD AS 2D 3C
    total 15-point value of that hand: 0
Subhand 4: AD AS 3C KC
    total 15-point value of that hand: 2
Subhand 14: 2D 3C 5C KC
    total 15-point value of that hand: 4
```

If we were looking only for combinations that add up to 15, the script has identified the best possible combination—that of a 2, 3, 5 and king. The problem is, runs are worth lots of points too, and the run of AD, AS, 2D, 3C is worth $AD+2+3=3$, $AS+2+3=3$, and the pair of aces adds another 2, so that's eight points, far more than the two fifteens. But, we'll get to that later.

For now, let's look at how the

fifteens are calculated by just examining the two-card case:

```
for subhand in {0..5}
do
  sum=0
  for thecard in ${fourtwo[$subhand]}
do
  sum=$(( $sum + ${c15[$thecard]} ))
done
if [ $sum -eq 15 ] ; then
  points=$(( $points + 2 ))
fi
done
```

Remember that the `$fourtwo` array is an enumerated list of all possible two-card combinations out of four (for example, 1+2, 1+3, 1+4, 2+3 and so on). The `$points` variable accumulates how many fifteens are found as the function tests two-card, three-card and, finally, the one possible four-card combination, ending with the `echo` statement:

```
echo " total 15-point value of that hand: $points"
```

We'll tweak that as the function expands in capabilities, but for now, that's useful.

Calculating Pairs

The next step is to calculate pairs. It turns out that we don't need any

additional code to calculate the value of three of a kind or four of a kind (though in years of playing *Cribbage*, I have never had four of a kind in my hand!), because they are unto themselves combinations of pairs. That is, if a player has 3D, 3S and 3H, it's worth six points: two points for 3D+3S, two points for 3D+3H and two points for 3S+3H—handy, really!

Because the `calc15()` function (shown in my last article) already offers a lot of the infrastructure we'll need, we're just going to expand on it, even though it technically won't be calculating only 15-point values any more. That's okay; we'll end up renaming the function, but for now, let's just code.

To extract pairs, the loop is slightly different:

```
twocards=${fourtwo[$subhand]}
card1=${twocards:0:1}
card2=${twocards:2}
```

Placing this within the snippet for `subhand in {0..5}` will let us test all two-card combinations of the four-card hand given to the function.

There's a bit of fancy variable referencing here too. It turns out we can extract substrings at a variable reference by using the following notation:

```
${string:position:length}
```

In the first instance, `card1` will end up being the first value in the `twocardsvariable`, which itself is extracted from the `fourtwo[]` array. Its format is "X Y", so the second reference needs to start at 2. Being lazy, we just grab the rest of the string, which means we can omit the `:length` parameter.

Why have the interim variable `twocards`? Because the shell can figure out only a certain level of complexity, and writing something like:

```
`${fourtwo[$subhand]}:0:1}
```

just gives me a headache.

The next step is simply to compare the two cards and see if they're the same rank:

```
if [ ${c15[$card1]} = ${c15[$card2]} ] ; then
    echo "we've got a pair: ${c15[$card1]} and ${c15[$card2]}"
fi
```

This all looks good, but there's a glaring bug in the code, as is immediately obvious with some debugging info:

```
Subhand 14: 9S 10H JH KD
calc15() given ranks: 9 10 10 10
PAIRS: testing two cards 0 and 1 from 0 1
PAIRS: testing two cards 0 and 2 from 0 2
PAIRS: testing two cards 0 and 3 from 0 3
```

```
PAIRS: testing two cards 1 and 2 from 1 2
we've got a pair: 10 and 10
```

```
PAIRS: testing two cards 1 and 3 from 1 3
we've got a pair: 10 and 10
```

```
PAIRS: testing two cards 2 and 3 from 2 3
we've got a pair: 10 and 10
```

The problem is that `calc15()` is being given the ranks of the cards after they've been scrubbed to just point values, so a 10H, JH and KD all look like they are point value 10. That works great for calculating fifteens, but a 10H+KD is most assuredly not a valid pair.

The fix is easy. We can just have `calc15()` get both the four normalized ranks and the four original ranks as parameters. Recall that in the function `handvalue4()` ranks are normalized through code blocks like this:

```
# now fix rank to normalize for value=10
case $r1 in
    11|12|13) nr1=10 ;;
    *) nr1=$r1 ;;
esac
```

So `$r1` already is the proper rank of the first card (that is, 1–13), and `$nr1` is the normalized rank (where a 10 and a K have value 10). Then, invoking `calc15()` is just a tiny bit more complex:

```
calc15 $nr1 $nr2 $nr3 $nr4 $r1 $r2 $r3 $r4
```

For notational convenience, let's also grab the 5th–8th parameters and reassign them into a local array `cr15[]` like this:

```
cr15[0]=$5; cr15[1]=$6; cr15[2]=$7; cr15[3]=$8
```

Now the fix to calculate proper pairs is quite easy:

```
Subhand 14: 10S JS QC QD
calc15() given ranks: 10 10 10 10
PAIRS: testing two cards 0 and 1 from 0 1
PAIRS: testing two cards 0 and 2 from 0 2
PAIRS: testing two cards 0 and 3 from 0 3
PAIRS: testing two cards 1 and 2 from 1 2
PAIRS: testing two cards 1 and 3 from 1 3
```

PAIRS: testing two cards 2 and 3 from 2 3
we've got a pair: 12 and 12

And, I'm out of space for this article. In my next article, we'll continue expanding on the pair calculations and add the final piece we need before we can actually clean it up and pick the best four out of six cards: testing for a flush, the situation where all four cards are of the same suit. ■

Dave Taylor has been hacking shell scripts for more than 30 years. Really. He's the author of the popular *Wicked Cool Shell Scripts* and can be found on Twitter as @DaveTaylor and more generally at <http://www.DaveTaylorOnline.com>.

LINUX JOURNAL

now available
for the **iPad** and
iPhone at the
App Store.



linuxjournal.com/ios



For more information about advertising opportunities within *Linux Journal* iPhone, iPad and Android apps, contact John Grogan at +1-713-344-1956 x2 or ads@linuxjournal.com.



KYLE RANKIN

Two Pi R

Raspberries are found in clusters, so why not Raspberry Pis? Kyle starts a series on building a redundant Raspberry Pi cluster.

Although many people are excited about the hardware-hacking possibilities with the Raspberry Pi, one of the things that interests me most is the fact that it is essentially a small low-power Linux server I can use to replace other Linux servers I already have around the house. In previous columns, I've talked about using the Raspberry Pi to replace the server that controls my beer fridge and colocating a Raspberry Pi in Austria. After I colocated a Raspberry Pi in Austria, I started thinking about the advantages and disadvantages of using something with so many single points of failure as a server I relied on, so I started thinking about ways to handle that single point of failure. When you see "Two Pi R", you probably think the R stands for the radius for a circle. To me, it stands for redundancy. I came to the conclusion that although one Pi isn't redundant, two Pi are.

So, in this article, I'm building

the foundation for setting up redundant services with a pair of Raspberry Pis. I start with setting up a basic clustered network filesystem using GlusterFS. In later articles, I'll follow up with how to take advantage of this shared storage to set up other redundant services. Of course, although I'm using a Raspberry Pi for this article, these same steps should work with other hardware as well.

Configure the Raspberry Pis

To begin, I got two SD cards and loaded them with the latest version of the default Raspberry Pi distribution from the official Raspberry Pi downloads page, the Debian-based Raspbian. I followed the documentation to set up the image and then booted in to both Raspberry Pis while they were connected to a TV to make sure that the OS booted and that SSH was set to start by default (it should be). You probably also will want to use

GlusterFS is a userspace clustered filesystem that I chose for this project because of how simple it makes configuring shared network filesystems.

the raspi-config tool to expand the root partition to fill the SD card, since you will want all that extra space for your redundant storage. After I confirmed I could access the Raspberry Pis remotely, I moved them away from the TV and over to a switch and rebooted them without a display connected.

By default, Raspbian will get its network information via DHCP; however, if you want to set up redundant services, you will want your Raspberry Pis to keep the same IP every time they boot. In my case, I updated my DHCP server so that it handed out the same IP to my Raspberry Pis every time they booted, but you also could edit the `/etc/network/interfaces` file on your Raspberry Pi and change:

```
iface eth0 inet dhcp
```

to:

```
auto eth0
iface eth0 inet static
    address 192.168.0.121
```

```
netmask 255.255.255.0
gateway 192.168.0.1
```

Of course, modify the networking information to match your personal network, and make sure that each Raspberry Pi uses a different IP. I also changed the hostnames of each Raspberry Pi, so I could tell them apart when I logged in. To do this, just edit `/etc/hostname` as root and change the hostname to what you want. Then, reboot to make sure that each Raspberry Pi comes up with the proper network settings and hostname.

Configure the GlusterFS Server

GlusterFS is a userspace clustered filesystem that I chose for this project because of how simple it makes configuring shared network filesystems. To start, choose a Raspberry Pi that will act as your master. What little initial setup you need to do will be done from the master node, even though once things are set up, nodes should fail over automatically. Here is the

information about my environment:

```
Master hostname: pi1
Master IP: 192.168.0.121
Master brick path: /srv/gv0
Secondary hostname: pi2
Secondary IP: 192.168.0.122
Secondary brick path: /srv/gv0
```

Before you do anything else, log in to each Raspberry Pi, and install the `glusterfs-server` package:

```
$ sudo apt-get install glusterfs-server
```

GlusterFS stores its files in what it calls bricks. A brick is a directory path on the server that you set aside for gluster to use. GlusterFS then combines bricks to create volumes that are accessible to clients. GlusterFS potentially can stripe data for a volume across bricks, so although a brick may look like a standard directory full of files, once you start using it with GlusterFS, you will want to modify it only via clients, not directly on the filesystem itself. In the case of the Raspberry Pi, I decided just to create a new directory called `/srv/gv0` for my first brick on both Raspberry Pis:

```
$ sudo mkdir /srv/gv0
```

In this case, I will be sharing my standard SD card root filesystem, but in your case, you may want more storage. In that situation, connect a USB hard drive to each Raspberry Pi, make sure the disks are formatted, and then mount them under `/srv/gv0`. Just make sure that you update `/etc/fstab` so that it mounts your external drive at boot time. It's not required that the bricks are on the same directory path or have the same name, but the consistency doesn't hurt.

After the brick directory is available on each Raspberry Pi and the `glusterfs-server` package has been installed, make sure both Raspberry Pis are powered on. Then, log in to whatever node you consider the master, and use the `gluster peer probe` command to tell the master to trust the IP or hostname that you pass it as a member of the cluster. In this case, I will use the IP of my secondary node, but if you are fancy and have DNS set up, you also could use its hostname instead:

```
pi@pi1 ~ $ sudo gluster peer probe 192.168.0.122
Probe successful
```

Now that my pi1 server (192.168.0.121) trusts pi2 (192.168.0.122), I can create my first volume, which I will call `gv0`. To do this,

I run the `gluster volume create` command from the master node:

```
pi@pi1 ~ $ sudo gluster volume create gv0 replica 2
➔192.168.0.121:/srv/gv0 192.168.0.122:/srv/gv0
Creation of volume gv0 has been successful. Please start
the volume to access data.
```

Let's break this command down a bit. The first part, `gluster volume create`, tells the `gluster` command I'm going to create a new volume. The next argument, `gv0`, is the name I want to assign the volume. That name is what clients will use to refer to the volume later on. After that, the `replica 2` argument configures this volume to use replication instead of striping data between bricks. In this case, it will make sure any data is replicated across two bricks. Finally, I define the two individual bricks I want to use for this volume: the `/srv/gv0` directory on `192.168.0.121` and the `/srv/gv0` directory on `192.168.0.122`.

Now that the volume has been created, I just need to start it:

```
pi@pi1 ~ $ sudo gluster volume start gv0
Starting volume gv0 has been successful
```

Once the volume has been started, I can use the `volume info` command on either node to see its status:

```
$ sudo gluster volume info
```

```
Volume Name: gv0
Type: Replicate
Status: Started
Number of Bricks: 2
Transport-type: tcp
Bricks:
Brick1: 192.168.0.121:/srv/gv0
Brick2: 192.168.0.122:/srv/gv0
```

Configure the GlusterFS Client

Now that the volume is started, I can mount it as a GlusterFS type filesystem from any client that has GlusterFS support. First though, I will want to mount it from my two Raspberry Pis as I want them to be able to write to the volume themselves. To do this, I will create a new mountpoint on my filesystem on each Raspberry Pi and use the `mount` command to mount the volume on it:

```
$ sudo mkdir -p /mnt/gluster1
$ sudo mount -t glusterfs 192.168.0.121:/gv0 /mnt/gluster1
$ df
```

Filesystem	1K-blocks	Used	Available	Use%	Mounted on
rootfs	1804128	1496464	216016	88%	/
/dev/root	1804128	1496464	216016	88%	/
devtmpfs	86184	0	86184	0%	/dev
tmpfs	18888	216	18672	2%	/run
tmpfs	5120	0	5120	0%	/run/lock
tmpfs	37760	0	37760	0%	/run/shm
/dev/mmcblk0p1	57288	18960	38328	34%	/boot
192.168.0.121:/gv0	1804032	1496448	215936	88%	/mnt/gluster1

Since I want to make sure that I could take down either of the two nodes and still have access to the files, I configured a separate client to mount this GlusterFS volume.

The more pedantic readers among you may be saying to yourselves, “Wait a minute, if I am specifying a specific IP address here, what happens when 192.168.0.121 goes down?” It turns out that this IP address is used only to pull down the complete list of bricks used in the volume, and from that point on, the redundant list of bricks is what will be used when accessing the volume.

Once you mount the filesystem, play around with creating files and then looking into `/srv/gv0`. You should be able to see (but again, don't touch) files that you've created from `/mnt/gluster1` on the `/srv/gv0` bricks on both nodes in your cluster:

```
pi@pi1 ~ $ sudo touch /mnt/gluster1/test1
pi@pi1 ~ $ ls /mnt/gluster1/test1
/mnt/gluster1/test1
pi@pi1 ~ $ ls /srv/gv0
test1
pi@pi2 ~ $ ls /srv/gv0
test1
```

After you are satisfied that you can mount the volume, make it permanent by adding an entry like the following to the `/etc/fstab` file on your Raspberry Pis:

```
192.168.0.121:/gv0 /mnt/gluster1 glusterfs defaults,_netdev 0 0
```

Note that if you also want to access this GlusterFS volume from other clients on your network, just install the GlusterFS client package for your distribution (for Debian-based distributions, it's called `glusterfs-client`), and then create a mountpoint and perform the same mount command as I listed above.

Test Redundancy

Now that I have a redundant filesystem in place, let's test it. Since I want to make sure that I could take down either of the two nodes and still have access to the files, I configured a separate client to mount this GlusterFS volume. Then I created a simple script called

What is a **Superhero** without the right tools?



No, we're not referring to costumed crime fighters who come to the rescue of others in trouble. We're talking about IT operations personnel - people with complex skills and a daunting job, who fight downtime, performance slowdowns, and other evils to keep your apps running 24x7. They have to be ready to take action if there is trouble brewing in the system – and being ready involves having the right tools.

ManageEngine provides the right set of monitoring tools for your IT operations team, enabling them to keep track of the performance of their complex apps from both within and outside their data center.

www.manageengine.com/apm

www.site24x7.com

- ⦿ Application Performance Monitoring
- ⦿ End User Experience Monitoring
- ⦿ Anomaly Detection
- ⦿ Automated Dependency Mapping
- ⦿ Deep Transaction Monitoring
- ⦿ Integrated Management Console



glustertest inside the volume:

```
#!/bin/bash

while [ 1 ]
do
    date > /mnt/gluster1/test1
    cat /mnt/gluster1/test1
    sleep 1
done
```

This script runs in an infinite loop and just copies the current date into a file inside the GlusterFS volume and then cats it back to the screen. Once I make the file executable and run it, I should see a new date pop up about every second:

```
# chmod a+x /mnt/gluster1/
glustertest
root@moses:~# /mnt/gluster1/
glustertest
Sat Mar  9 13:19:02 PST 2013
Sat Mar  9 13:19:04 PST 2013
Sat Mar  9 13:19:05 PST 2013
Sat Mar  9 13:19:06 PST 2013
Sat Mar  9 13:19:07 PST 2013
Sat Mar  9 13:19:08 PST 2013
```

I noticed every now and then that the output would skip a second, but in this case, I think it was just a

function of the date command not being executed exactly one second apart every time, so every now and then that extra sub-second it would take to run a loop would add up.

After I started the script, I then logged in to the first Raspberry Pi and typed `sudo reboot` to reboot it. The script kept on running just fine, and if there were any hiccups along the way, I couldn't tell it apart from the occasional skipping of a second that I saw beforehand. Once the first Raspberry Pi came back up, I repeated the reboot on the second one, just to confirm that I could lose either node and still be fine. This kind of redundancy is not bad considering this took only a couple commands.

There you have it. Now you have the foundation set with a redundant file store across two Raspberry Pis. In my next column, I will build on top of the foundation by adding a new redundant service that takes advantage of the shared storage. ■

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.



CLOUD WORLD FORUM

26th - 27th June 2013 • National Hall Olympia, London

www.cloudwf.com

Harness the Cloud opportunity in your business

FREE EXHIBITION
Register Your Place Now

Over 200 Speakers Including:



Dr Jeff Jaffe
CEO
World Wide Web Consortium



Francisco Garcia Moran
Director General
Informatics
European Commission



Dana Deasy
CIO
BP



Oskar Stål
CTO
Spotify



Eric van Miltenburg
Senior Vice President
YouSendIt



Jim Reavis
Co-Founder and
Executive Director
Cloud Security Alliance



Jeff Barr
Chief Evangelist
Amazon Web Services



Richard Harris
CIO
ARM



Sujay Jaswa
VP of Business
Development
Dropbox



John Lewis

Paul Coby
IT Director
John Lewis



Daniel Marion
Head of IT
UEFA



Tony McAlister
CTO
Betfair

2013 Sponsors Include:

Diamond Sponsor:



Platinum Sponsors:



Gold Sponsors:



Silver Sponsors:



Show Highlights:

1. Over 5,000 senior IT decision makers from around the globe
2. 200 visionary speakers – gain a unique insight from industry heavyweights and hear case study examples
3. 8 theatres with 150 seminars answering all of your cloud computing questions
4. 150 global exhibitors helping you discover the latest and most innovative IT products
5. Co-located with the Big Data World Congress – leading 2 day conference



SHAWN POWERS

The Google Giveth

Be prepared for the Google Reader shutdown!

And the Google taketh away. So it is with Google Reader. A while back, Google discontinued its Google Wave product, because it never gained traction as a social-media platform. This surprised approximately zero people. More recently, Google announced it would be closing Google Reader on July 1, 2013. Far more people were surprised, myself included. In this article, I want to explore some options for those about to be left in the lurch.

Those Clouds Look Ominous

I think even more interesting than Google eliminating Google Reader is the collateral damage it's doing to cloud computing in general. Reader is something I've used for years, depended on in fact, to keep up with the Web sites I find interesting. Google Reader is a program I'd happily pay for, but since it's free, I've always just counted my blessings and moved on. Now that

it's disappearing, my dependence on free and/or cloud-based services is weighing heavily on me. Today it's Google Reader; will tomorrow be the end of Dropbox? Flickr? Google Mail?

Since Google's announcement regarding the demise of Reader, I've visited SourceForge and Github more frequently than I have in years. I don't like Google being able to affect my day-to-day computing so dramatically on a whim, and so I've been working hard to make myself less dependent on services like Google Reader. This is the first in what I expect might be a series of articles on self-sufficiency in this cloudy new world.

Web Shmeb

The simplest way to avoid losing your cloud-based solutions is to avoid Web-based services altogether. Before the original Web applications like Bloglines and Google Reader came about, people were perfectly happy with standalone RSS readers. Many

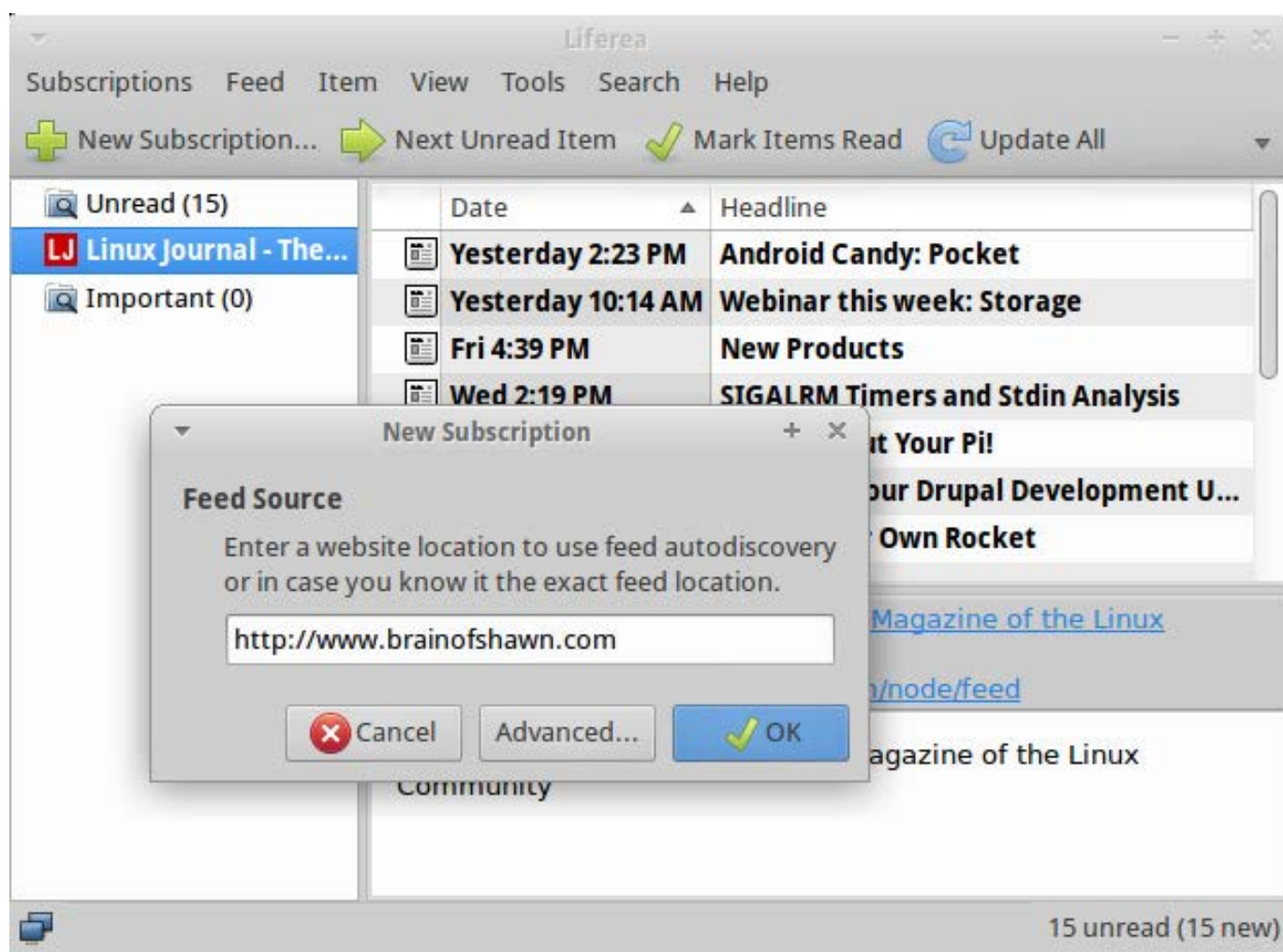


Figure 1. Liferea is simple, but that's not a bad thing.

folks still use a standalone application, and if you tend to browse the Web from the same computer all the time, a standalone application might be the perfect solution.

Liferea is a Linux-native application that does a nice job of managing RSS feeds (<http://lzone.de/liferea>). Like almost every other RSS application, it syncs with Google Reader, but thankfully, it also syncs with Tiny Tiny RSS (more on Tiny Tiny RSS later).

Because it has the ability to sync with a back-end database, Liferea can provide the best of both worlds—namely, a local application for browsing RSS feeds, plus syncing with a common back end for reading on other devices and computers. Liferea has a simple interface, but if you want to burn through your RSS feeds, simple is good (Figure 1).

Tons of other RSS clients work very well under Linux—Akgregator,

Even in light of Google shutting down its much-beloved Reader Application, as long as you keep your eyes wide open, there's nothing wrong with using on-line services—just be ready for them to disappear.

Thunderbird, RSSOwl and many others. I specifically mention Liferea because of its ability to sync with Tiny Tiny RSS, but plenty of perfectly usable RSS readers are available. Check apt-get or your distro's equivalent for "RSS" and you should find several.

Read RSS in Your Browser!

I know it seems like circular logic, but Web browsers can do so much more than browse Web sites. Anyone with a Chromebook can attest to how powerful a browser can be. Firefox has extensions like Sage, Brief or Simple RSS Reader. I could show them all in action, but really, I just recommend going to <http://addons.mozilla.org> and searching for "RSS". See which ones look appealing, and give them a try!

If you fall on the Google Chrome (or Chromium) side of the fence, there are plenty of Chrome extensions for RSS feed-reading as well. Slick RSS, Feed Reader and several others exist. Google also supplies the RSS

Subscription Extension, which allows you to add feeds to your Web-based subscription service directly from the Web site you'd like to add. It recently removed Google Reader as a destination, which makes sense, but other options are available, which leads me to the next possibility.

Third Cloud to the Right

Before I talk about hosting your own solution, I think it's only fair to discuss a few other options available from third parties. Even in light of Google shutting down its much-beloved Reader Application, as long as you keep your eyes wide open, there's nothing wrong with using on-line services—just be ready for them to disappear.

Feedly: Arguably the most popular "alternative" to Google Reader is Feedly (<http://www.feedly.com>). In a slightly ironic twist, Feedly uses Google Reader as its back-end API for keeping your RSS feeds in sync across devices. Feedly was designed as a

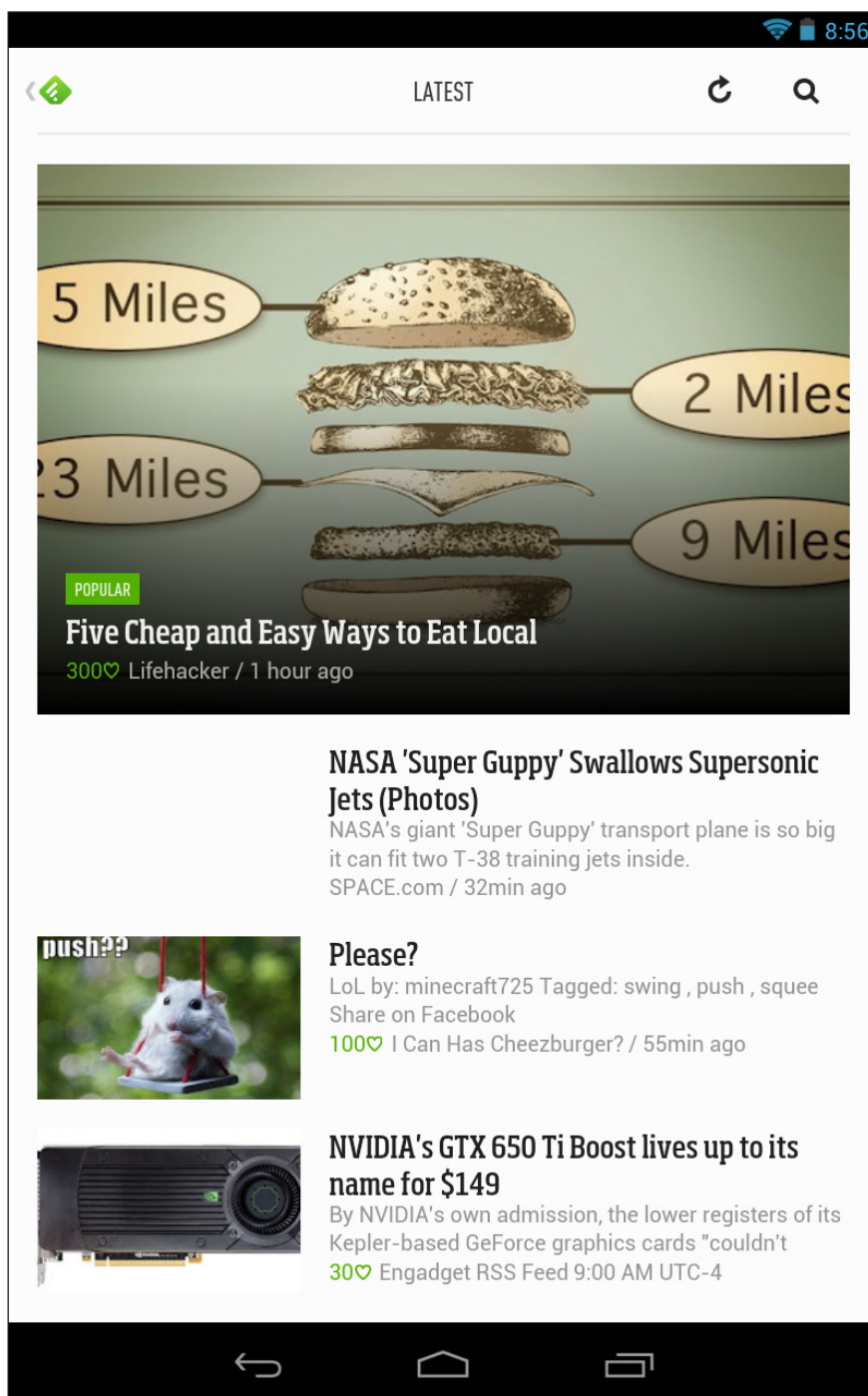


Figure 2. Feedly has many cross-platform clients, plus it works on the Web.

front end to Google Reader, but from a user's perspective, it's an alternative. The Feedly folks have announced

they will be transitioning users seamlessly from the Google Reader back end to whatever their new platform will be. It's still a testament to how much people (even companies) depend on free services to remain available.

Feedly takes a far more visual approach to RSS feeds, and by default, it presents a magazine-like view of Web stories. (See the Feedly Android client in Figure 2.) Some people really like this, and it seems to be a trend for RSS readers of late. Personally, I find it annoying, but I can see the appeal. Feedly is free, but it soon will be offering a "pro" version that supports off-line browsing.

The Old Reader: If the new flipboard/magazine look offends your very being, another on-line alternative to Reader is The Old Reader

(<http://theoldreader.com>). I suspect it's not a coincidence that the interface to The Old Reader looks almost

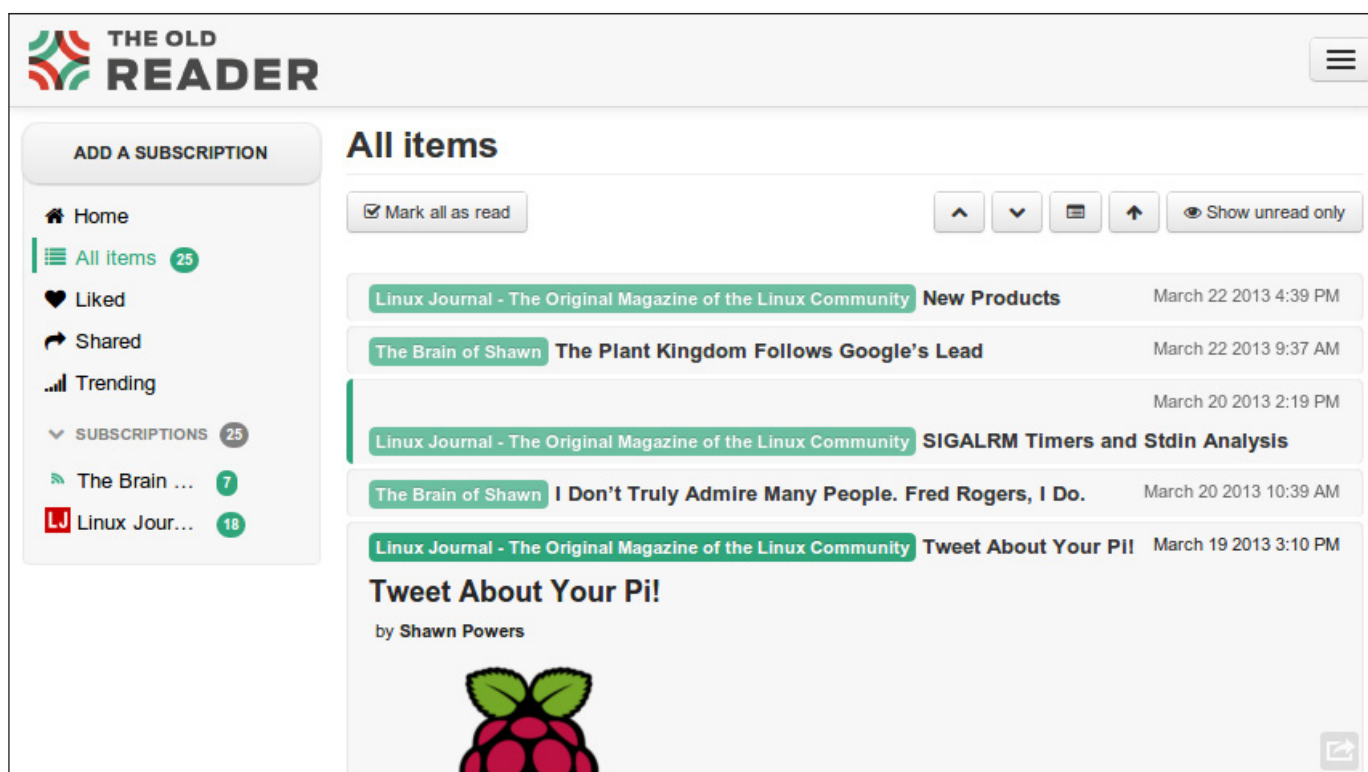


Figure 3. The Old Reader looks like, well, The Old Google Reader!

identical to how Google Reader looked in the old days (Figure 3). It appears to be completely free, which I'd normally consider a good thing, but thanks to Google, I'm gun-shy about such things now. Nonetheless, The Old Reader can import your Google Reader feeds, and it functions almost exactly like the Google Reader of old (with at least one major exception, which I discuss later).

The Old Reader doesn't import Google Reader subscriptions automatically, but it does allow the import of a subscription file in the format Google provides. Because The Old Reader doesn't use Google Reader as its back end,

the demise of the latter shouldn't affect the former. It is possible that the mass exodus of Google Reader users will have an adverse affect on performance, but hopefully that can be overcome.

NewsBlur: NewsBlur is an interesting contender for "Google Reader Replacement". It is a fully open-source program, but the service provided from <http://www.newsblur.com> offers a very restricted "free" offering. I want to like NewsBlur, especially based on its open nature, but the free offering is so limited (it limits the number of feeds you can add), it's hard

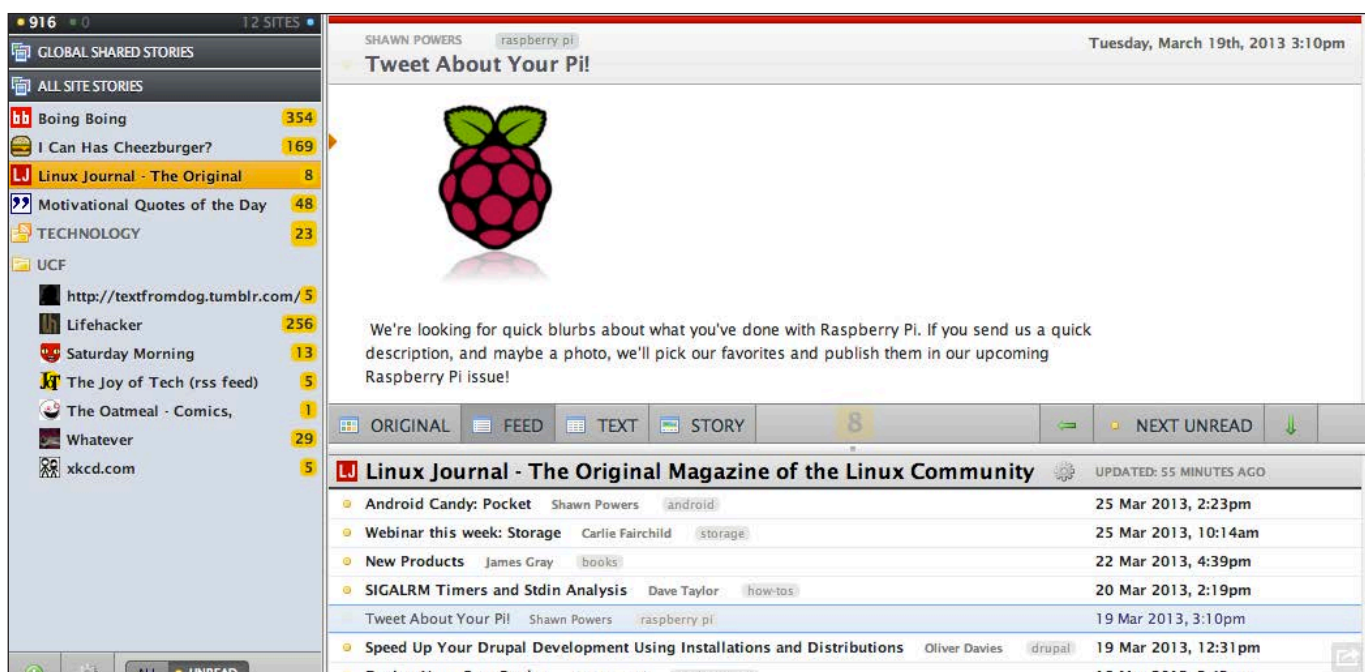


Figure 4. NewsBlur has a great interface, there's no denying it.

to test it long enough to justify the subscription fee. Like Feedly, NewsBlur offers a more “exciting” interface for browsing your feed (Figure 4). If you prefer that sort of look, NewsBlur is worth checking out. The open-source reality of NewsBlur brings me to the next and final section.

Make Your Own Cloud

Although I'm not quite ready to abandon Gmail and host my own e-mail again, I have to admit I've been researching my options ever since the announcement of Google Reader's demise. For my RSS needs, however, I've decided to host my own Web-based RSS reader. Google

Reader going away really disrupts my lifestyle, and I want to make sure I'm not setting myself up for failure by choosing another third-party service.

My first attempt at replacing Google Reader was to install my own copy of NewsBlur. It's a little more glitzy than I like, but it's open source. I fired up my Web-hosting service and created a new site for hosting NewsBlur—and then spent hours beating my head against the wall.

Don't get me wrong, NewsBlur is indeed open source. The code is freely available from GitHub. There are installation instructions, but it's still fairly difficult to install. I

Tiny Tiny RSS reminds me more of a standalone RSS reader like Liferea than a Web-based program, but when you start exploring its plugins and addons, you might wonder why you've been using Google Reader all this time!

understand developers not devoting a ton of time holding people's virtual hands for an end result that would cut directly into their bottom line (NewsBlur is a commercial service after all). Still, if you're thinking you just need a simple LAMP stack, you'll be very surprised.

NewsBlur depends on Django, Celery, RabbitMQ, MongoDB, Pymongo, Fabric, jQuery, PostgreSQL or MySQL, and tons of configuration to get it running. I'm not saying the program is poorly designed. I'm saying that I'm lazy, and installations like WordPress have spoiled me. If you're adventurous enough, installing your own instance of NewsBlur may be very rewarding. I prefer something simpler if I need to maintain it, however. Enter: Tiny Tiny RSS (<http://tt-rss.org/redmine>).

Tiny Tiny RSS: Like the name implies, Tiny Tiny RSS is small. It's a PHP application that requires a back-end MySQL database and nothing else. It literally took less than five minutes

for me to install and configure Tiny Tiny RSS on my Web server (Figure 5).

Tiny Tiny RSS reminds me more of a standalone RSS reader like Liferea than a Web-based program, but when you start exploring its plugins and addons, you might wonder why you've been using Google Reader all this time! If you recall, at the beginning of this article, I mentioned that Liferea would sync with Tiny Tiny RSS. When you add the fact that it can act as a back end to standalone clients, the availability of an Android application and the countless plugins available, it's easy to fall in love with Tiny Tiny RSS. Even if you end up going with a more glitzy alternative, you owe it to yourself to give Tiny Tiny RSS a try.

Sadly, Nobody Surfs Like Me

I'm doing my best to focus on the positive side effects of Google's decision to close down Google Reader. It's forced me (and many

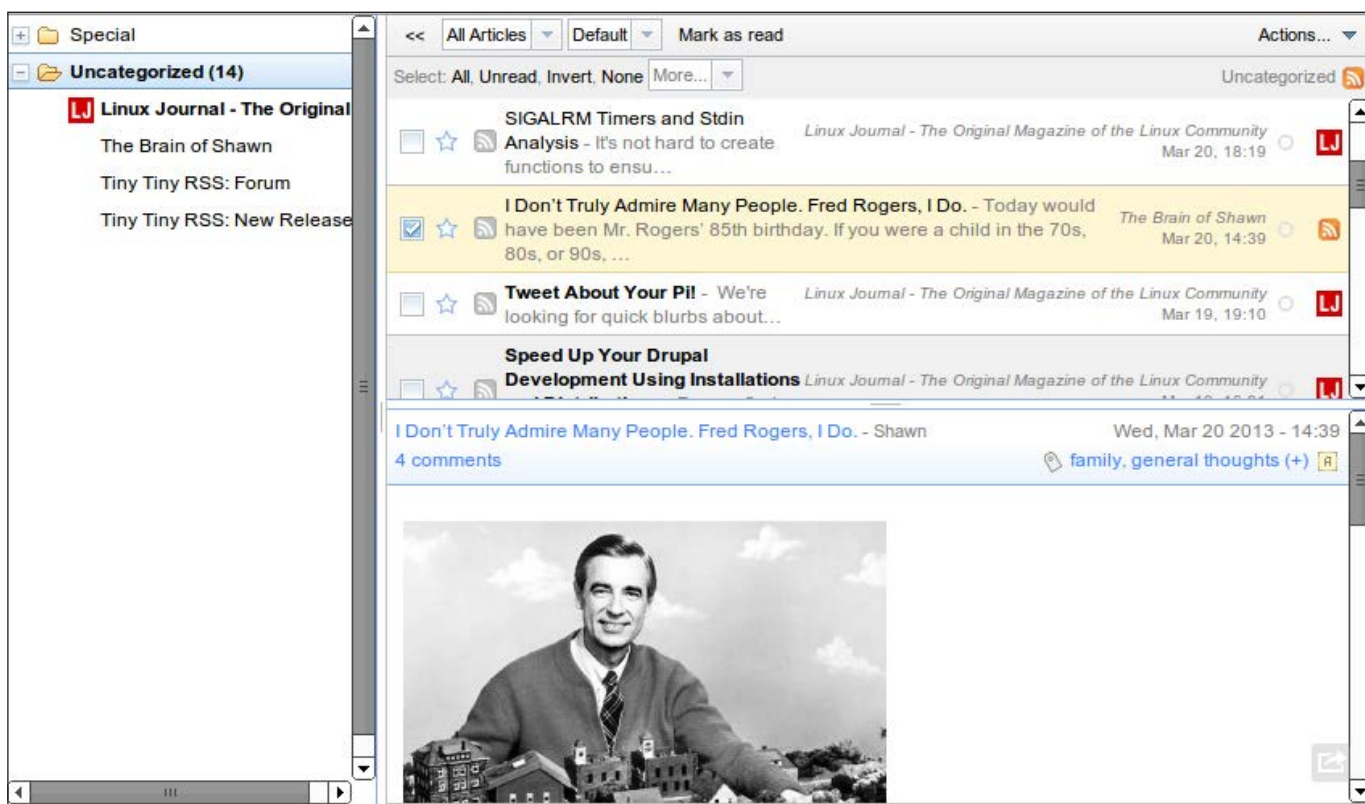


Figure 5. Tiny Tiny RSS is tiny, and it interfaces with plugins and clients alike.

others) to take a serious look at where I'm putting my data, plus it's forced me to think outside my little box. In all my research, however, I still haven't found a way to replicate the obscure Google Reader feature that has been my sole way to browse the Internet for a half decade—the "next unread" bookmarklet. I demonstrated the feature in a *Linux Journal* Tech Tip years ago: <http://www.youtube.com/watch?v=ILGqEsVDPqQ>.

Maybe someone will create a Tiny Tiny RSS plugin that does this for me. Maybe it will be the

reason I finally learn to program on my own. Nevertheless, this seemingly simple feature is one I can't find anywhere else. If anyone has recommendations on how to replicate that feature, or if there are any Tiny Tiny RSS programmers out there looking for a weekend project, I'd love to hear about it! ■

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

NeuroSky's Brainwave Starter Kit

Another inch toward a *Star Trek* reality comes in the form of NeuroSky's Brainwave Starter Kit, a brain-computer interface package that lets users analyze and visualize their own brain waves on Android and iOS. Users simply slip on the MindWave Mobile EEG headset and view their own brainwaves displayed on their handset in the colorful Brainwave Visualizer.

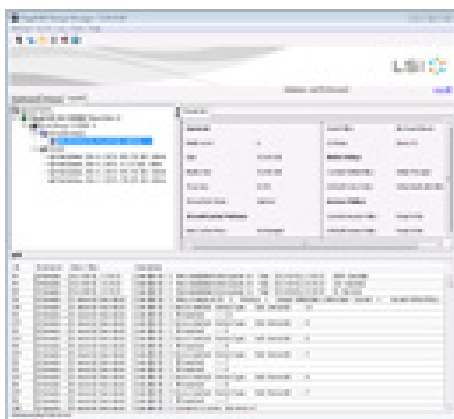
The brain waves visibly change in real time, as one chills out to relaxing music and then does some intensive programming or imagines something Marvin Gaye sings about. The MindWave Mobile safely measures and outputs the EEG power spectra (alpha waves, beta waves and so on), NeuroSky eSense meters (attention and meditation) and eye blinks. The device consists of a headset with an ear clip and a sensor arm. The headset's reference and ground electrodes are on the ear clip, and the EEG electrode is on the sensor arm, resting on the forehead above the eye. Next up: when will someone finally make a transporter?

<http://www.neurosky.com>

BRAINWAVE STARTER KIT

Now Available for

\$99.99



Open-E Data Storage Software V7

The model for the Open-E Data Storage Software is an all-in-one universal storage system that allows businesses of all sizes to leverage off-the-shelf servers to build and operate virtualized storage infrastructure. The Linux-based Open-E DSS V7 data storage software is used for building and managing centralized data storage servers—NAS and SAN—and now includes Hyper-V Cluster Support, as well as the new product Open-E Multiple Storage Server Manager for simplified central management of all storage resources. Built-in enterprise-class features include active-active failover, active-passive failover, volume replication, snapshots and continuous data protection, among others.

<http://www.open-e.com>



AdaCore GNAT Pro Safety-Critical for ARM Processors

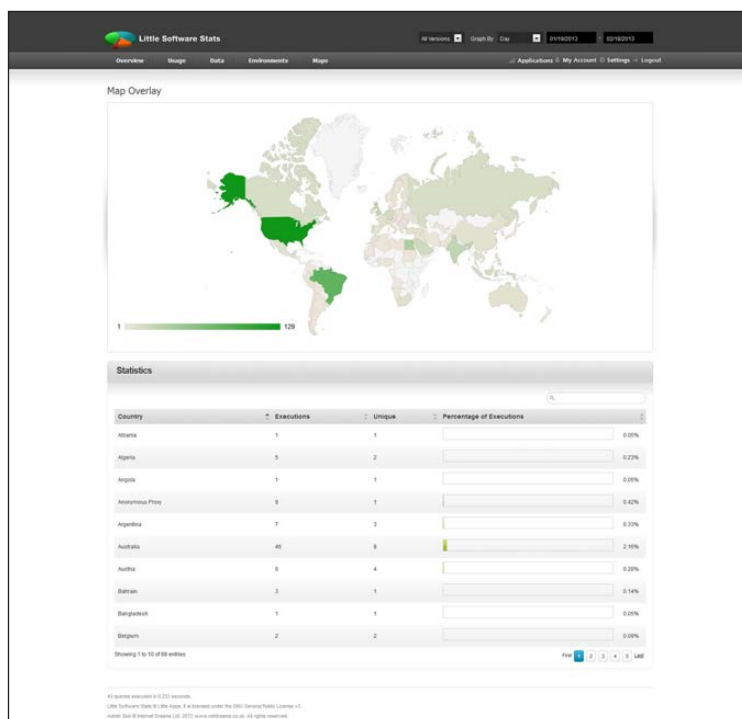
In response to the increasing prevalence of low-cost, low-power ARM processors in the aerospace, defense and transportation industries, AdaCore has developed the GNAT Pro Safety-Critical product for ARM Cortex micro-controllers. This bareboard GNAT Pro Safety-Critical product provides a complete Ada development environment (or Eclipse plugin) oriented toward systems that are safety-critical or have stringent memory constraints. Developers of such systems, says AdaCore, now can exploit the software engineering benefits of the Ada language, including reliability, maintainability and portability. The ARM platform adds to the GNAT Pro Safety-Critical product offering, which already is available for PowerPC and LEON boards, allowing easy portability among all three platforms. The technology does not require any underlying operating system, so it can be deployed on very small memory boards.

<http://www.adacore.com>

Little Apps' Little Software Stats

The big news from Little Apps is Little Software Stats, a new open-source software package that enables software developers to track how users are using their software. Little Apps says that Little Software Stats is the first program for obtaining runtime intelligence that is both open source and free. Because the program is designed and developed using MySQL and PHP, most Web servers can run it. Information that can be gathered includes executions, installations, exceptions and geographical location.

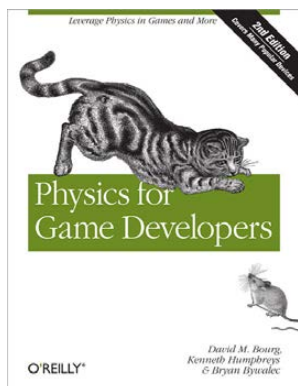
<http://little-apps.org>



Rakhitha Nimesh Ratnayake's *Building Impressive Presentations with Impress.js* (Packt Publishing)

Impress.js is a classic open-source project. The closed-source world creates a slick app—in this case, the Prezi presentation tool—and charges much more than free. Next, creative (and cheapskate) open-source developer says “I want that, and I want it free”—in this case, Impress.js. Impress.js, not to be confused with OpenOffice.org Impress, is a free and open-source JavaScript library inspired by Prezi that utilizes the CSS3 transitions found in modern Web browsers. If this sounds interesting, the path to mastery can be found with Rakhitha Nimesh Ratnayake's new book *Building Impressive Presentations with Impress.js*. Ratnayake explores the features of Impress.js, which allows one to create presentations inside the infinite canvas of modern Web browsers that work anywhere, any time and on any device. Readers will learn how to build dynamic presentations with rotation, scaling, transforms and 3-D effects. Advanced users will find out how to extract the power of Impress.js core API events and configurations to modify existing functionalities, as well as extend the core library to create custom functionalities for different types of applications, such as sliders, portfolios and galleries.

<http://www.packtpub.com>



David M. Bourg, Kenneth Humphreys and Bryan Bywalec's *Physics for Game Developers, 2nd edition* (O'Reilly Media)

Whether or not game development is your vocation, you have to admit the material in *Physics for Game Developers, 2nd edition*, is inherently super interesting. In this book, the authors explore the key knowledge behind bread-and-butter game physics that go into modern games for Nintendo Wii, PlayStation Move, Microsoft's Kinect and various mobile devices. Readers also learn how to leverage exciting interaction gadgets, such as accelerometers, touch screens, GPS receivers, pressure sensors and optical tracking devices. The updated 2nd edition includes new chapters on deformable and soft bodies, fluids and the physics of sound for incorporating realistic effects, including 3-D sound. Major topics include digital physics, the physics of sound, rigid body mechanics, fluid dynamics and the modeling of specific systems based on real-world examples.

<http://oreilly.com>



VanDyke Software SecureFX

VanDyke Software says that many of its customers run Windows just so they don't have to give up the wide range of functionality found on its SecureFX secure file transfer client. Now Linux and Mac OS X users can enjoy that same functionality natively with the new SecureFX 7.1, which delivers features like multiple file transfer protocols, site synchronization and easy recovery of interrupted transfers. Secure file transfers can be performed with SFTP, FTP over SSL and SCP; FTP is provided for use on legacy systems. Aesthetics also are important for VanDyke, which offers a tabbed visual user interface in SecureFX 7.1 allowing for easy learning and organizing for optimum productivity. Also new in SecureFX 7.1 is a dependent session option that can be used to link a session to an SSH2 session that it depends on, which allows connection to a jump host before connecting to other sessions.

<http://www.vandyke.com>

Copernica Marketing Software's MailerQ

MailerQ is a high-performance mail transfer agent designed to do one thing: deliver e-mail at blazing speed. Leveraging RabbitMQ to send up to 10,000 e-mail messages per minute, MailerQ is targeted at users who want the functionality of programs like Port25's PowerMTA but can't stomach its high price. With MailerQ, users are limited only by the ability to process incoming e-mail at the receiving end. The application puts e-mail messages in a queue and enables its users to manage these messages themselves and change parameters, such as send rate per receiving domain or IP or the number of delivery attempts. To be as efficient as possible, MailerQ stores messages in Couchbase NoSQL, which allows the retrieval of content only when an SMTP connection has been established, thus reducing the amount of data being passed back and forth in RabbitMQ message queues.

<http://www.mailerq.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.

RASPBERRY PI: THE PERFECT HOME SERVER



Exploit the Raspberry Pi's unique low-power and energy-efficient architecture to build a full-featured home backup, multimedia and print server.

BRIAN TRAPP

Ever since the announcement of the Raspberry Pi, sites all across the Internet have offered lots of interesting and challenging uses for this exciting device. Although all of those ideas are great, the most obvious and perhaps least glamorous use for the Raspberry Pi (RPI) is creating your perfect home server.

If you've got several different computers in need of a consistent and automated backup strategy, the RPi can do that. If you have music and video you'd like to be able to access from almost any screen in the house, the RPi can make that happen too. Maybe you have a printer or two you'd like to share with everyone easily? The Raspberry Pi can fill all those needs with a minimal investment in hardware and time.



Raspberry Pi Benefits

Low cost: for \$35, the RPi model B is nearly a complete computer with 512MB of RAM, 100Mb Ethernet, an SD card slot, two USB ports, audio out and HDMI or RCA video out. I've seen HDMI cables that cost more than that.

Energy efficient: hardware costs are only one component of a server's expense, because you also need to consider the energy cost to keep the device running constantly. The services needed for home use aren't going to tax the CPU much, and most of the time it will just be idling, waiting for something to do. The RPi's ultra-low power components are a perfect fit for this workload, which helps keep your power bill down. My model B unit plus external hard drive consume only 8 watts total, while the old Athlon-based box it replaced drew 54 watts at idle. Assuming 10 cents per kilowatt hour, that puts the yearly power bill for an RPi at \$7 vs. \$47 for an Athlon-based machine. The RPi basically pays for itself in less than a year!

Low noise: because the RPi doesn't have fans or moving parts, the only component in your final configuration that generates noise or any appreciable heat will be the hard



Figure 1. A Compact, but Highly Capable Home Server

disk. If you're concerned about noise, enthusiast sites like Silent PC Review (<http://www.silentpcreview.com>) often include noise benchmarks in their storage reviews. My experience is that any modern drive is quiet enough to avoid detection anywhere there's something else already running (such as a media center, gaming console or other computer). If your home doesn't provide a lot of flexibility for wiring options, the RPi's small size, minimal thermal

output and low-noise footprint may make it possible to sneak in a server where it was difficult to justify one in the past.

New opportunities: a less tangible benefit is the simple joy of trying something new! For me, this was my first time really working on a Debian-based distribution, and it's probably the first time many Linux enthusiasts will have a chance to try an ARM-based architecture.

Arranging the Hardware

For a home server, you'll need a medium-size SD Flash card for local storage. It's possible to use a USB thumbdrive for booting, but that would use up one of the two precious USB slots. The Flash storage card doesn't need to be large, but the faster the better. I chose a name-brand SD card with an 8GB capacity and class 10 speed rating. For backups and multimedia files, a large hard drive with a USB dock is a must. I chose a 1.5TB hard drive and a Calvary EN-CAHDD-D 2-bay USB 2.0 hard drive dock. This dock has a feature to run two drives in RAID-0 mode, which could be useful someday. Finally, the RPi doesn't come with a power supply, but most smartphone chargers supply the required 5v-over-micro USB. To see if the RPi

was fussy about the power source, I swapped through three different micro-USB cell-phone chargers for power supplies. I tried each one for about a week, with no issues on any of the units.

Installing the Operating System

Installing the RPi operating system is covered in extensive detail elsewhere, but here are a few home-server-specific tips, roughly in the order needed.

1) Get the Raspbian "Wheezy" install image directly from <http://www.raspberrypi.org/downloads>, and copy it onto the SD card, using the steps listed on the site.

2) When booting the RPi for the first time, attach a keyboard, mouse and monitor. Don't forget to turn on the monitor before booting the RPi, so that it can detect the correct HDMI or composite output port.

3) The RPi has a nice "raspi-config" screen that you'll see on first boot. For a home server, the following selections will be useful:

- `expand_rootfs`: resizes the default 2GB OS image to fill the rest of the Flash card.
- `change_pass`: the default password is "raspberrypi", but something more

secure than that would be better.

- Set your locale and timezone.
- `memory_split`: assign the minimum amount possible (16) to the GPU to leave as much room as possible for services.
- SSH: don't forget to enable the SSH server.
- `boot_behaviour`: turn off boot to desktop (again, to save memory for your services).

When finished, you'll be at the `pi@raspberrypi` prompt. The setup script can be re-run at any time via `sudo raspi-config`.

There are just a few more configuration items, and then the operating system is ready to go.

1) A static IP makes everything easier, so switch the network settings for `eth0`:

```
>> sudo nano -w /etc/network/interfaces
```

Change the `eth0` line `iface eth0 inet dhcp` to the following (modify to meet your home network setup):

```
=====/etc/network/interfaces====  
...
```

```
iface eth0 inet static  
address 192.168.1.10  
netmask 255.255.255.0  
gateway 192.168.1.1  
...  
=====/etc/network/interfaces====
```

2) Create a local user, and put it in the users and sudo group:

```
>> sudo adduser YOURUSERIDHERE  
>> sudo usermod -a -G users YOURUSERIDHERE  
>> sudo usermod -a -G sudo YOURUSERIDHERE
```

3) Update the system to ensure that it has the latest and greatest copies of all the libraries:

```
>> sudo apt-get update; sudo apt-get upgrade
```

4) At this point, you're ready to go headless! Shut down the PI:

```
>> sudo /sbin/shutdown -h now
```

Once it's down (monitor the green status LEDs on the RPi circuit board to know when it has finished shutting down), unplug the monitor, keyboard, mouse and power cord. Attach the USB storage, then restart the RPi by plugging the power back in.

5) Once the RPi starts up (again, those green LEDs are the clue to its state), you can `ssh` in to the RPi from

any other machine on the network and finish all the configuration remotely from here on out (modify the following for your static IP):

```
>> ssh YOURUSERIDHERE@192.168.1.10
```

Congratulations, you have a working Raspberry Pi!

Peripherals

The first order of business is to get the external storage device mounted. Use `dmesg` to look for where the storage device was found—it almost certainly will be `/dev/sda`. I like using automounter to handle mounting removable storage devices, as it is more flexible about handling devices that may not be present or ready at boot time:

```
>> sudo apt-get install autofs
>> sudo nano -w /etc/auto.master
===== /etc/auto.master =====
...
/misc /etc/auto.misc
...
===== /etc/auto.master =====

>> sudo nano -w /etc/auto.misc
```

Note, my external storage device is formatted with `ext4`—modify this for your needs if required:

```
===== /etc/auto.misc =====
...
storage -fstype=ext4 :/dev/sda1
...
===== /etc/auto.misc =====
>> sudo /etc/init.d/autofs restart
>> ls -lat /misc/storage
```

Optionally, create a symlink to shorten the path a smidgen:

```
>> ln -s /misc/storage /storage
```

Backup Repository

At the top of any home server feature list is providing rock-solid backups. With the RPi, this is pretty simple, due to the wide range of network-sharing options in Linux: Samba/CIFS for Windows machines, NFS for UNIX-based devices and even SFTP for more advanced backup clients like `deja-dup`. Because the RPi has only 100Mb Ethernet, and the storage device is on USB, it's not going to have super-fast transfer speeds. On the other hand, good backup clients run automatically and in the background, so it's unlikely that you'll notice the slightly slower transfer speeds.

My home network includes one Windows 7 machine. For it, I exported a backup directory on the RPi's external USB storage device via Samba. Because the backup utility in the

basic version of Windows 7 doesn't support network drives as a backup destination, I used SyncBack Free (<http://www.2brightsparks.com/freeware/freeware-hub.html>) to set up automated, daily backups.

Configuring Samba is simple.

1) Install the samba and common-bin library (which has the smbpasswd utility):

```
>> sudo apt-get install samba samba-common-bin
```

2) Use smbpasswd to let your local ID have access:

```
>> sudo smbpasswd -a YOURUSERIDHERE
```

3) Edit the samba configuration file:

```
>> sudo nano -w /etc/samba/smb.conf
```

4) Change the workgroup = WORKGROUP line to match your Windows workgroup name.

5) Comment out or delete the [homes] and [printers] share. (Printer sharing will be done later via direct CUPS access.)

6) Add an entry for the Windows backup paths. Here's my example, which I placed at the bottom of the file:

```
=====/etc/samba/smb.conf=====
```

```
...
```

```
[win7pc]
```

```
comment=Backup for windows PC
```

```
path=/storage/win7pc
```

```
writeable=Yes
```

```
create mask=0777
```

```
directory mask=0777
```

```
browsable=Yes
```

```
public=Yes
```

```
valid users=YOURUSERIDHERE
```

```
...
```

```
=====/etc/samba/smb.conf=====
```

7) Restart Samba to implement your edits:

```
>> sudo /etc/init.d/samba restart
```

8) Test connectivity from the Windows machine by mapping a network drive from the file explorer.

For Linux devices, deja-dup is brilliantly simple to set up and use. It's been installed by default on both my Fedora 18 and Ubuntu 12.10 installs. While the package name is "deja-dup", the front end is simply called "Backup". Although the RPi easily could support NFS export, I've found that using deja-dup's SSH option is easier and more portable, and it eliminates the need for an additional service on the RPi. Specifying a deja-dup encryption password is probably a good idea, unless you like the idea of

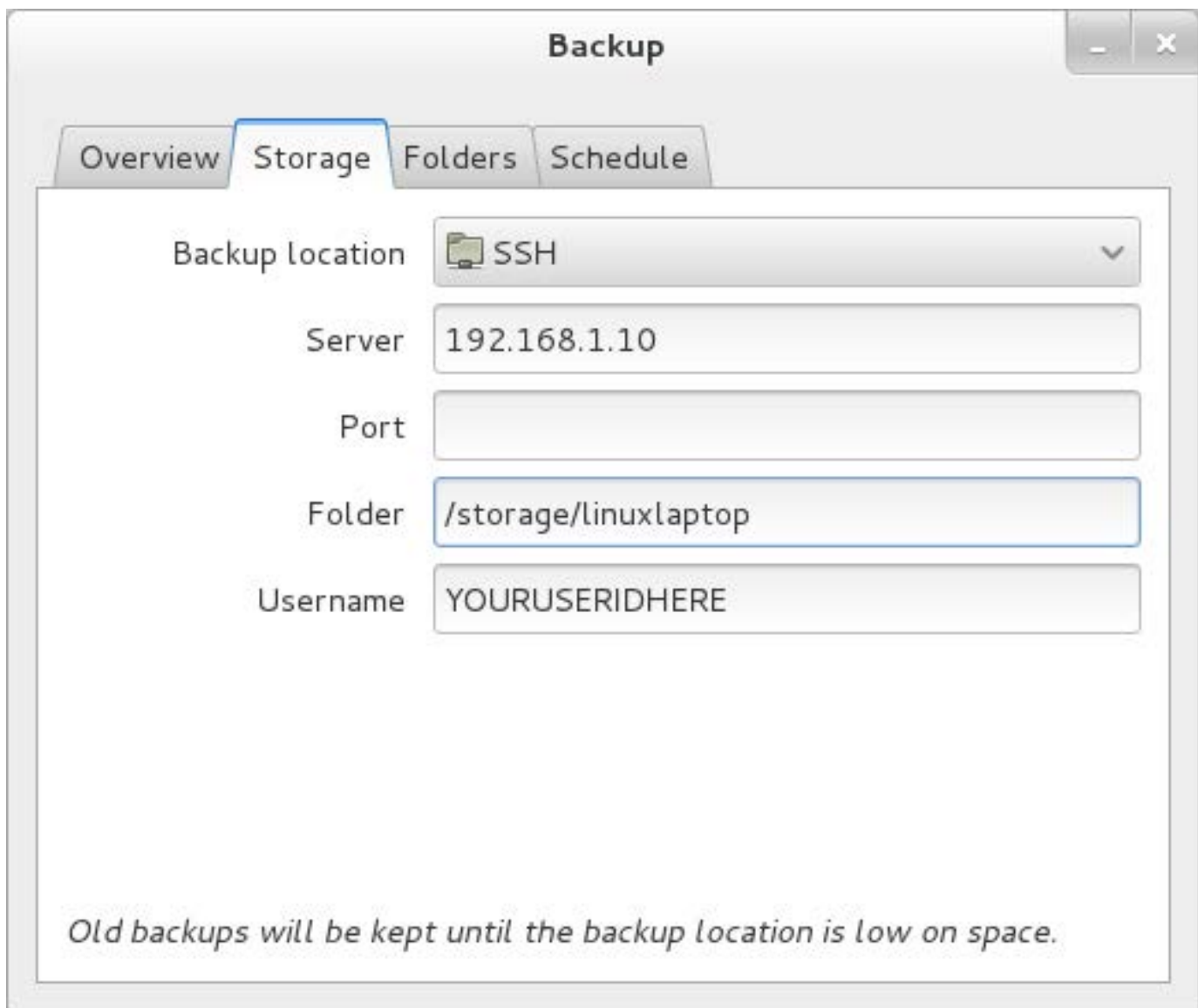


Figure 2. Deja-dup Client Setup

all your files walking off if someone pockets the storage drive:

```
>> sudo mkdir /storage/linuxlaptop  
>> sudo chown -R YOURUSERIDHERE:YOURUSERIDHERE /storage/linuxlaptop
```

From the client Linux machine, launch the backup utility, choose "SSH" as the backup location, and

enter the RPi's IP address and the storage location you just created. The first backup will be slow, but future runs will be sending only incremental changes, which is significantly faster.

Multimedia Server: DLNA

Now that everyone's files are backed up safely, let's move on to

some fun! A DLNA server will give you a central place to store your movies, music and pictures. From this central repository, DLNA clients from every screen in the house can play back this content with ease.

At least, that's the promise. The reality is that the DNLA specs don't quite nail down many important things like which formats or encodings are supported. Each client typically has a slightly different idea of what formats and server features it would like to support. A much higher-power server might be able to transcode local content to device-supported formats on the fly, but that's not possible on the RPi, and the on-the-fly transcoding often messes up other features like pause, fast-forward and rewind. In general, higher-powered devices like the PS3, Xbox and WD TV devices can handle most formats without any transcoding. Lower-end devices like smart TVs or Blu-ray players support a much more limited list of codecs.

For the RPi, your best bet is simply to encode to the standards your primary DLNA device supports and then test your other DLNA clients. If they won't play nicely, the tips in the next section may help. In my case, my PlayStation

3 acts as the DLNA client, which plays nicely with the compact .m4v files generated by Handbrake.

Minidlna is a great choice for the RPi DLNA server. It's already in the Raspbian distribution, is quite simple to set up and uses minimal server resources while running:

```
>> sudo apt-get install minidlna
>> sudo nano -w /etc/minidlna.conf
```

Here are the relevant sections of my /etc/minidlna.conf:

```
...
# I found keeping video + audio in different paths helpful
media_dir=V,/storage/dlna/video
media_dir=A,/storage/dlna/music
...
presentation_url=http://192.168.1.10:8200/
...
friendly_name=MyRPi
...
# Since I add new media infrequently, turning off
# inotify keeps minidlna for polling for
# content changes. It's simple enough to run
# sudo /etc/init.d/minidlna force-reload
# when new content is added.
inotify=no
```

Once done editing, tell minidlna to restart and rescan for content:

```
>> sudo /etc/init.d/minidlna force-reload
```

Minidlna has the ability to provide movie-poster thumbnails for your movies for devices that support it (like the PS3). It makes finding a specific movie when scrolling through dozens of movie files much more convenient. I've found that the most compatible file layout is to have one directory per movie, containing just the movie file plus the thumbnail image named "Cover.jpg". Using a format like "MovieName.m4v" and "MovieName.jpg" works fine for the PS3, but it breaks VLC (if you can convince the VLC uPNP plugin to find the server in the first place).

From the PS3, you can test connectivity by going to "Video" on the XMB bar. The "friendly_name" you set previously should be visible when scrolling down in the Video section. If you can't find it, test to ensure that minidlna is up by going to `http://192.168.1.10:8200/` with a Web browser.

Multimedia for Non-DLNA Devices

Once you get DNLA working with some of your devices, you may find devices it doesn't want to work with, so a multimedia plan B is a good idea. The nginx Web server has an MP4 plugin that tries to improve streaming over plain-old HTTP, but browser playback performance varied

widely, and fast-forwarding within a movie didn't work consistently either. It seems like the lowest common denominator for multimedia sharing across fussy or non-DLNA devices is a good-old-fashioned Samba share with guest read-only access.

Here's an sample section from `/etc/samba/smb.conf`:

```
[dlna]
path=/storage/dlna
read only=yes
browsable=yes
public=yes
```

After defining the share and restarting Samba (`sudo /etc/init.d/samba restart`), you can start to test out your clients.

I tested the following clients with a mix of videos encoded with Handbrake as m4v files:

- Android 4.0.4 phone: "ES File Explorer" with "ES Media Player" (player comes with install).
- Android 4.1.2 tablet: "ES File Explorer" with "ES Media Player" (player comes with install).
- Linux devices: automount `://192.168.1.10/dlna`, then use VLC or MPlayer.

- Windows: mount //192.168.1.10:/dlna, then use VLC.

All devices were able to start playing almost instantly and fast-forward with no delays.

Print Server

The RPi runs CUPS quite well, so it's easy to share an older printer that doesn't have native networking features.

Install CUPS and any packages needed by your printer. I needed hplip-cups since I have an HP inkjet printer:

```
>> sudo apt-get install cups hplip-cups
```

Update the "Listen" line and add the Allow @LOCAL block to the Location directives as shown below (so you can use other machines on your LAN to administer CUPS):

```
=====/etc/cups/cupsd.conf====  
#Listen localhost:631 #Comment this out  
Listen 192.168.1.10:631 #Add this line  
...  
<Location />  
    Order allow,deny  
    Allow @LOCAL  
</Location>  
  
# Restrict access to the admin pages...
```

```
<Location /admin>  
    Order allow,deny  
    Allow @LOCAL  
</Location>  
  
# Restrict access to configuration files...  
<Location /admin/conf>  
    AuthType Default  
    Require user @SYSTEM  
    Order allow,deny  
    Allow @LOCAL  
</Location>  
=====/etc/cups/cupsd.conf====
```

Add your local ID to the lpadmin group so you can administer CUPS:

```
>> sudo usermod -a -G lpadmin YOURUSERIDHERE
```

Restart CUPS:

```
>> sudo /etc/init.d/cups restart
```

Then, go to <http://192.168.1.10:631/> and click "Adding Printers and Classes" to set up your printer. My printer was auto-discovered on the USB, so all I had to do was click "share". Also access <https://192.168.1.10:631/admin>, and make sure to check "Share printers connected to this system".

Once you're done, you can set up your clients the usual way. My Linux clients auto-discovered

the printer and picked the right printer drivers once I entered the hostname. On my Windows 7 machine, once I selected “Network Printer”, I had to click “The printer that I want isn’t listed”, select “Select a shared printer by name” and then enter the URL from the CUPS Web interface: http://192.168.1.10:631/printers/HP_J4500.

Conclusion

With a minimal amount of additional hardware and configuration, the

Raspberry Pi can be a highly capable, compact home server. It can bring the wide range of enterprise services offered by Linux into a home environment with minimal hardware expense. ■

Brian Trapp serves up a spicy gumbo of Web-based yield reporting and analysis tools for hungry semiconductor engineers at one of the leading semiconductor research and development consortiums. His signature dish has a Java base with a dash of JavaScript, Perl, Bash and R, and his kitchen has been powered by Linux ever since 1998. He works from home in Buffalo, New York, which is a shame only because that doesn't really fit the whole chef metaphor.

LINUX JOURNAL

on your
Android device

Download app now in
the **Android Marketplace**



www.linuxjournal.com/android

For more information about advertising opportunities within *Linux Journal* iPhone, iPad and Android apps, contact John Grogan at +1-713-344-1956 x2 or ads@linuxjournal.com.

AUTOCONFIGURING AN IPV6 ACCESS POINT with SixXS and a Raspberry Pi

I use my Raspberry Pi as a low-cost IPv6 router and tunnel endpoint to provide IPv6 connectivity to my entire home LAN. Using SixXS as my tunnel provider, I configured my Raspberry Pi to be completely LAN-agnostic, which means I don't have to worry about my ISP changing my public IPv4 address. I even can take my Raspberry Pi IPv6 router to friends' houses to make their networks IPv6-capable as well.

IGOR PARTOLA

IPv6 is the next-generation IP protocol designed to deal with the address exhaustion we are facing with IPv4. IPv6 addresses are 128 bits long, which is enough to address 340,282,366,920,938,000,000,000,000,000,000,000 individual devices.

You can do some unique things with IPv6 today. Here's a short list:

- Access Google, Facebook and Wikipedia over IPv6.
- Avoid using dynamic DNS to access your home/office computers when on the go.
- Get IPv6 certification from Hurricane Electric (<http://ipv6.he.net/certification>).
- Use IPv6-capable equipment for anything from teleconferencing to sensor networks (<https://www.sixxs.net/misc/toys>).
- Watch *Star Wars* in ASCII over IPv6: Telnet to towel.blinkenlights.nl:23 (technically you can do this over IPv4 as well).

For a more comprehensive list of what you can do, see <https://www.sixxs.net/misc/coolstuff>.

There are many ways to get IPv6

connectivity, including getting native support from your ISP or setting up a tunnel using an IPv6 tunnel broker. Because most ISPs are slow to provide native IPv6 support, let's set up a tunnel using SixXS (pronounced "six access") and a Raspberry Pi. SixXS is a major IPv6 tunnel provider that supports a number of tunneling protocols.

A Raspberry Pi is a low-cost single-board computer that you can obtain for \$25 or \$35. It runs Linux on its ARM-based processor and is powerful enough to act as an IPv6 router without breaking a sweat. There are many uses for a Raspberry Pi—from a Web server to an XBMC-based media center or a *Minecraft* gaming rig.

For the purposes of this article, I assume that you have a LAN with a single dynamic IPv4 address from your ISP and that you already have a Raspberry Pi and are running Raspbian, the de facto standard Linux distribution for it. You can find documentation for how to get started with the Raspberry Pi elsewhere (<http://elinux.org/RaspberryPiBoard>). I further assume that your Raspberry Pi is a model B, which has a built-in Ethernet port. You certainly can use a Model A, but you will need to add a USB Ethernet or a Wi-Fi USB adapter to it first.

As I mentioned, SixXS supports a few different kinds of tunnel protocols, but we are interested only in one: Anything-In-Anything (AYIYA). This protocol is essentially IPv6 packets wrapped in IPv4/UDP datagrams. It has the nice property that it traverses most NAT setups without any special configuration required on the router. Additionally, when you use AYIYA with SixXS's autoconfiguration client (AICCU), you avoid using any LAN-specific configuration. This means once you have everything set up, you can take your Raspberry Pi IPv6 router anywhere. Just power it up on your friend's LAN, and it instantly becomes IPv6-connected! Let's get started.

you can request a tunnel and a subnet. You will need to provide a reason explaining why you want to use IPv6 through SixXS. Provide as detailed a reason as you can, but it might just boil down to "I want to get my local network IPv6-enabled to familiarize myself with how IPv6 works." Make sure to select AYIYA for the tunnel type.

2) Enable IPv6 routing on your Raspberry Pi:

```
$ echo "net.ipv6.conf.all.forwarding=1" |  
➔sudo tee -a /etc/sysctl.conf  
$ sudo sysctl -p
```

3) Once your request is approved, you will have addresses for a tunnel

This means once you have everything set up, you can take your Raspberry Pi IPv6 router anywhere. Just power it up on your friend's LAN, and it instantly becomes IPv6-connected!

1) Register for an account with SixXS (<https://www.sixxs.net/signup/create>). This is a multistep process (<https://www.sixxs.net/faq/account/?faq=10steps>) where some steps will require manual approval from one of the SixXS members, but it normally goes pretty quickly. Once you have registered for an account,

and a subnet. The tunnel address is what your Raspberry Pi will use to communicate with SixXS. You will use the subnet to hand out addresses to the rest of the devices on your LAN. Keep the Tunnel Information page open while you are proceeding with the setup. Be patient. Typically, SixXS reviews tunnel requests within a few

hours to a few days.

4) Because every IPv6-capable device on your network immediately will receive a globally accessible IPv6 address, you need to set up a firewall to protect them. Indeed, not all devices (such as printers) come with a firewall, so it is a good idea to limit connectivity to them from the outside world. This is done using a few `iptables` commands:

```
$ sudo iptables -A INPUT -i lo -j ACCEPT
$ sudo iptables -A INPUT -m conntrack
  --ctstate RELATED,ESTABLISHED -j ACCEPT
$ sudo iptables -A INPUT -p tcp -m tcp --dport 22 -j ACCEPT
$ sudo iptables -A INPUT -s 2001:4830:xxxx:xxx::/64 -j ACCEPT
$ sudo iptables -A INPUT -s 2001:4830:xxxx:Yxxx::/64 -j ACCEPT
$ sudo iptables -A INPUT -p ipv6-icmp -j ACCEPT
$ sudo iptables -A INPUT -j DROP

$ sudo iptables -A FORWARD -m conntrack
  --ctstate RELATED,ESTABLISHED -j ACCEPT
$ sudo iptables -A FORWARD -p tcp -m tcp --dport 22 -j ACCEPT
$ sudo iptables -A FORWARD -s 2001:4830:xxxx:Yxxx::/64 -j ACCEPT
$ sudo iptables -A FORWARD -p ipv6-icmp -j ACCEPT
$ sudo iptables -A FORWARD -j DROP
```

Note that you are letting two IPv6 subnets through: `2001:4830:xxxx:xxx::/64` and `2001:4830:xxxx:Yxxx::/64`. The one with the `Yxxx` is going to be the routed subnet. That's the one the rest of the devices on your network will

use. The one with just the `xxx` will have only two addresses on it: `::1` (the remote end of your tunnel) and `::2` (your Raspberry Pi).

5) Make sure your firewall is enabled at boot time, which is easy. Put the following into `/etc/network/if-pre-up.d/iptables-load`, and make it executable (`$ sudo chmod 755 /etc/network/if-pre-up.d/iptables-load`):

```
#!/bin/sh
iptables-restore < /etc/iptables.rules
exit 0
```

Now, put the following into `/etc/network/if-post-down.d/iptables-save`, and make it executable (`$ sudo chmod 755 /etc/network/if-post-down.d/iptables-save`):

```
#!/bin/sh
iptables-save -c > /etc/iptables.rules

if [ -f /etc/iptables.downrules ]; then
    iptables-restore < /etc/iptables.downrules
fi
exit 0
```

For good measure, execute:

```
$ sudo /etc/network/if-post-down.d/iptables-save
```

6) Now that your firewall is up and

running, let's bring up the IPv6 tunnel. All you need to do is install the SixXS client called from the standard Raspbian. The package is called `aiccu`.

During the installation process, you will be asked a few questions, such as your tunnel ID (for example, T1110xx), your SixXS user name and password and so on. Note that you can set up a tunnel-specific password to avoid having your general SixXS password in plain text on your Raspberry Pi. After the installation is complete, the configuration file will be generated at `/etc/aiccu.conf`.

After AICCU is installed and running, you will have two new network interfaces available: `sit0` and `sixxs`. `sit0` is the IPv6-in-IPv4 tunnel, and `sixxs` is a virtual Ethernet interface that has the IPv6 tunnel address. You can check for these by running `$ ip addr`.

At this point, your Raspberry Pi has IPv6 connectivity, which you can check by pinging an IPv6-capable host:

```
$ ping6 google.com
```

7) The next step is to provide IPv6 addresses to the rest of your network using the routed subnet. First, edit your `/etc/aiccu.conf` and enable the "setupscript" option:

```
-#setupscript /usr/local/etc/aiccu-subnets.sh
+setupscript /usr/local/etc/aiccu-subnets.sh
```

Next, create the script at `/usr/local/etc/aiccu-subnets.sh` with the following content:

```
#!/bin/sh
ip addr add 2001:4830:xxxx:Yxxx::1/64 dev eth0
```

where `2001:4830:xxxx:Yxxx::1` is your routed subnet address. Next, restart AICCU. Check the output of `$ ip addr`, and make sure that `2001:4830:xxxx:Yxxx::1` is an address associated with your `eth0` interface.

8) The last step is to install `radvd`, the router advertising daemon, and configure it to advertise to your LAN. After the installation is complete, make sure that `/etc/radvd.conf` has the following content:

```
interface eth0 {
    AdvSendAdvert on;
    MinRtrAdvInterval 3;
    MaxRtrAdvInterval 10;

    prefix 2001:4830:xxxx:Yxxx::/64 {
        AdvOnLink on;
        AdvAutonomous on;
        AdvRouterAddr on;

        AdvValidLifetime 30;
        AdvPreferredLifetime 20;
    };
};
```

Restart radvd, and check that IPv6 is available on other devices on your network. Most modern computers will pick up the change immediately.

Going forward, AICCU will detect any changes to your LAN automatically, such as an updated public IPv4

of tunnels, such as static tunnels or heartbeat tunnels from SixXS. Or, you could take a look at tunnel providers other than SixXS, such as Hurricane Electric's Tunnel Broker (<https://tunnelbroker.net>) or Freenet6 (<http://www.gogo6.com/freenet6>).

Once you get this basic setup up and running, you can build on top of it. For example, you can provide your network with a local caching DNS server that can resolve hostnames over IPv6 by simply installing bind9.

address from your ISP, and re-adjust the tunnel settings. Your Raspberry Pi is now accessible from anywhere in the world using its IPv6 address at 2001:4830:xxxx:Yxxx::1. Your other devices will get their own public static IPv6 addresses. You can use an IPv6 connectivity test at <http://test-ipv6.com> to make sure that everything works.

As an experiment, you can bring your Raspberry Pi to a friend's house to see if you can make the network IPv6-capable there.

Once you get this basic setup up and running, you can build on top of it. For example, you can provide your network with a local caching DNS server that can resolve hostnames over IPv6 by simply installing bind9.

You also can look into other types

You can use your Raspberry Pi as a Web server, serving content over IPv6. Apache2 and nginx both support IPv6 out of the box, and many guides are available for turning your Raspberry Pi into a Web server.

Another thing to try is setting up VoIPv6 on your Raspberry Pi with VOVIDA (<http://www.linuxjournal.com/article/7047>).

In conclusion, using IPv6 opens up a world of new possibilities and lets you do things you could not do before. Give it a try and have fun. ■

Igor Partola is an independent software developer specializing in scalable, distributed Internet applications. He has a particular interest in free and open-source software, as well as networking. He often is described by his friends as an "IPv6 nut", because he is constantly advocating that people get IPv6 access.



CONTROL THE LIMELIGHT WITH A RASPBERRY PI

**ROCK YOUR STAGE LIGHTING WITH
A RASPBERRY PI LIGHT CONTROLLER.**

JONATHAN BROGDON

I am the proud parent of twin 13-year-old rock-star daughters. They are seriously good musicians, and of course, I taught them everything they know. The girls typically perform indoors “coffee-house” style or outdoors in the daylight. But this fall, the girls booked a number of outdoor gigs in the evening. This meant we had to have stage lighting. After a quick trip to the local music instrument supercenter—where they know us by name—we had all the lights, stands and trusses we needed for a good night-time show. The light system we purchased had many useful built-in options for creating light shows: various color chase sequences, variable strobe speeds and sound-activated or fixed-time sequencing. These are great options. But, if you are looking for maximum wow-factor from a performance, you want to have direct control over the light system.

Lighting control hardware is available to control your stage lighting. In addition, several commercial and open-source software lighting control packages work with many off-the-shelf lighting devices. Some of these, such as QLC and QLC+, are available for Linux. Many of the software light controller packages are based on the Open Lighting

Architecture (<http://www.opendmx.net/index.php/OLA>). If you are looking to set up a professional light show and need to control several complex devices, such as movable lights, one of these options is probably the way to go.

I am the sound guy/roadie/driver/financier for this band. So I figured, what could it hurt to add lighting control guy to my résumé? Given my other duties during the show, I wanted something pretty simple to operate, with capability to add custom features as needed. Commercial lighting control hardware seemed like overkill for my needs, and I wasn't real keen on dragging a laptop to a show just to control the lighting. This sounded like a good Raspberry Pi project.

When I started investigating lighting control options, I quickly found the language of light control: DMX512. The United States Institute for Theatre Technology created DMX512 in 1986 as a standard to control dimmers on stage lights. The standard has had a few updates since 1986 and is now an ANSI standard. DMX512 uses a multi-drop RS-485 bus for the physical layer signaling. There is a single master device (the controller) and up to 512 slave devices (the lights). The RS-485



Figure 1. Raspberry Pi DMX512 Light Controller (photo by Jonathan Brogdon)

differential signaling provides for good noise immunity over the long cable runs needed to control lights that are spread around a stage or arena. Each slave DMX512 device has IN and OUT XLR connections, and devices are daisy-chained together. The last device in the chain should include an XLR stub connector with a 120-Ohm termination resistor. For the girls' stage lighting, we used two Chauvet COLORstrip Mini LED wash lights.

The DMX512 protocol is very basic. Remember, it originally was designed for light dimmers. The only protocol message, sent by the master, is a block of unsigned byte values (0–255)—one for each “channel”. Each slave device has its own definition of how a “channel” is interpreted. For example, channel one may set the general light mode (fixed color, fixed sequence, random sequence and so on). The second channel may indicate the sequence transition speed (fixed value, sound activated and so on). On the other hand, a very simple device might define the channel values as dimmer values for each light on the device. The manufacturer's documentation will spell out how each channel should be used. The DMX512 protocol runs

at a fixed 250k baud rate.

One interesting quirk of the protocol is that messages must be sent continuously. If the master stops sending messages to a device, the lights go out. This also means that once a device receives a message with channel values specifying a particular light sequence, it will run the sequence as long as it keeps receiving messages with the same channel values. For example, suppose the value for channel 1 is 80, which indicates that the color sequence is red, green, blue, yellow, magenta, cyan, white. The value for channel 2 is 10, which indicates that the color should change every ten seconds. If you want to keep this sequence running, the master must keep sending messages with channel 1=80 and channel 2=10.

I found a simple USB-RS-485 converter based on the ubiquitous FT232R USB-Serial chip from Future Technologies Devices International. You can find these for around \$20 or less from several sources. I got mine on eBay. I attached a three-pin female XLR connector to the RS-485 I/O pins, and voilà, I had my DMX512 physical interface. The Entec Open DMX USB is another popular option (http://www.enttec.com/open_dmx_usb). This product also is

based on the FT232R chip and comes in a nice sturdy steel enclosure, all for \$70. The code in this article works with either option. Note that the Open DMX USB box has a five-pin XLR connector. If you plan on using the less-expensive three-conductor DMX512 cables, you will need to add an XLR five-pin-to-three-pin adapter.

Now, I needed a software interface for sending DMX512 protocol messages. The main task here is configuring the FT232R chip for DMX512 communication at 250 kBaud with eight data bits, no parity and two stop bits. Most Linux distributions have a driver that supports the FT232 family of devices. However, if you are thinking about using the serial device interface presented by the driver (for example, `/dev/ttyUSB0`), you probably won't get far. Standard utilities like `stty` will not support setting the 250-kBaud rate. Instead, you need to talk to the FT232R as the USB device that it is.

My language of choice for this project was Python. So, I looked around for Python packages that could address the FT232R chip as a USB device. I found `PyFtdi` (<https://github.com/eblot/pyftdi.git>). `PyFtdi` provides a user-space driver for talking to FTDI devices. The `PyFtdi` package depends on `PyUSB`

(<https://github.com/walac/pyusb.git>), which depends on `LibUSB` (<http://www.libusb.org>). I used the standard Raspian "Wheezy" image for this project. So, I was able to install `libusb` via the package manager. I installed `PyUSB` and `PyFtdi` via the setup scripts included with each package. The source for the DMX512/USB interface is shown in Listing 1. Note that a DMX512 protocol message must be terminated with two break characters.

The DMX512 controller needed a user interface. My first thought was for a grand Web interface over an ad hoc Wi-Fi network. The plan was to run the UI from a browser on my phone or tablet. But, that would mean more gear to be charged up and dragged to the show. Plus, I've learned (the hard way) to pick reasonably economical solutions for band support gear so I can afford a backup. I decided to use a much simpler UI device: the Adafruit Pi Plate LCD+Keypad. This is a two-line x 16-character LCD with five key buttons: "up", "down", "left", "right" and "enter". This device is made specifically for use with the Raspberry Pi, and it plugs in directly to the GPIO header. The Pi Plate uses the Hitachi HD44780 LCD controller. The LCD interface and key buttons are accessed via an i2c I/O expander on the Pi Plate.

Listing 1. DMX512/USB Interface Module

```
from pyftdi.pyftdi import ftdi

#FTDI device info
vendor=0x0403
product=0x6001

#####
# DMX USB controller
#####
class OpenDmxUsb():
    def __init__(self):
        self.baud_rate = 250000
        self.data_bits = 8
        self.stop_bits = 2
        self.parity = 'N'
        self.flow_ctrl = ''
        self.rts_state = 0
        self._init_dmx()

    #Initialize the controller
    def _init_dmx(self):
        self.ftdi=ftdi.Ftdi()
        self.ftdi.open(vendor,product,0)
        self.ftdi.set_baudrate(self.baud_rate)
        self.ftdi.set_line_property(self.data_bits,
                                   self.stop_bits,
                                   self.parity,
                                   break_=0)

        self.ftdi.set_flowctrl(self.flow_ctrl)
        self.ftdi.purge_rx_buffer()
        self.ftdi.purge_tx_buffer()
        self.ftdi.set_rts(self.rts_state)

    #Send DMX data
    def send_dmx(self,channelVals):
        self.ftdi.write_data(channelVals)
        # Need to generate two bits for break
        self.ftdi.set_line_property(self.data_bits,
                                   self.stop_bits,
                                   self.parity,
                                   break_=1)
        self.ftdi.set_line_property(self.data_bits,
                                   self.stop_bits,
                                   self.parity,
                                   break_=1)
        self.ftdi.set_line_property(self.data_bits,
                                   self.stop_bits,
                                   self.parity,
                                   break_=0)
```

Adafruit provides some example Python code for the Pi Plate (<https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code.git>), demonstrating how to write to the LCD and read from the key buttons. This code is great for a single application that controls all the LCD resources. I needed the UI to run automatically when the system boots. I could have adapted the Adafruit Pi Plate example code to run as a *dæmon*. However, I

considered LCDproc to be a better choice (<http://lcdproc.org>). LCDproc is the de facto UI framework for embedded Linux systems that support LCDs—with or without keys.

The LCDproc framework is composed of a server *dæmon* with “drivers” for specific LCD families and physical connection options (that is, parallel, serial and so on). Note that these are userspace drivers that use device interfaces to access the LCD hardware. An LCDproc client

application contacts the server over a socket to output to the LCD or read key input. A client can allocate an area of the LCD for its output or map individual keys for its input. For example, you might have a client that displays the system uptime on the left side of the first line, another client that displays the current system time on the right side of the first line, and a third client that displays the IP address on the second line. I liked the flexibility of having multiple LCD clients for future applications. The LCDproc communication protocol uses text strings over a TCP socket connection. This makes for easy protocol debugging.

LCDproc provides a driver for the HD44780 LCD, with several connection options. Unfortunately, LCDproc did not have driver support for the Pi Plate's LCD connection via the i2c I/O expander. So, I decided to add it. See <https://github.com/jlbrogdon/lcdproc-piplate.git> for the Pi Plate driver patch.

LCDd is the LCDproc server daemon. The LCDd configuration file contains several server and driver options. By default, the path to this file is /etc/LCDd.conf. For the Pi Plate LCD, the following config options should be used:

```
[server]

# Where can we find the driver modules?
# IMPORTANT: Make sure to change this setting to reflect your
#           specific setup! Otherwise LCDd won't be able to find
#           the driver modules and will thus not be able to
#           function properly.
# NOTE: Always place a slash as the last character!
DriverPath=/usr/local/lib/lcdproc/

Driver=hd44780

[menu]

# You can configure what keys the menu should use.
# Note that the MenuKey will be reserved exclusively;
# the others work in shared mode.

# Up to six keys are supported. The MenuKey (to enter
# and exit the menu), the EnterKey (to select values)
# and at least one movement key are required.
# These are the default key assignments:
#MenuKey=Escape
EnterKey=Enter
UpKey=Up
DownKey=Down
LeftKey=Left
RightKey=Right

## Hitachi HD44780 driver ##

[hd44780]

# Select what type of connection.
# See documentation for types.
ConnectionType=i2c-piplate
```

```

# i2c address for the I/O expander
Port=0x20

# Device of the serial interface [default: /dev/lcd]
Device=/dev/i2c-1

# Btrate of the serial port (0 for interface default)
Speed=0

# If you have a keypad connected.
# You may also need to configure the keypad layout
# further on in this file.
Keypad=yes

# If you have a switchable backlight.
Backlight=yes

# Specifies the size of the LCD.
# In case of multiple combined displays, this should
# be the total size.

```

```
Size=16x2
```

```

KeyDirect_1=Enter
KeyDirect_2=Up
KeyDirect_3=Down
KeyDirect_4=Left
KeyDirect_5=Right

```

A complete config file is available with the patch.

With the LCDproc server up and running, I could focus on the LCD client side of my light controller. I used the Python `lcdproc` package (<https://github.com/jingleman/lcdproc.git>) for the client interface to LCDproc. This allowed the DMX512 light controller client to open a connection to the LCDproc server, set up widgets on the LCD, send output to the widgets and get input from

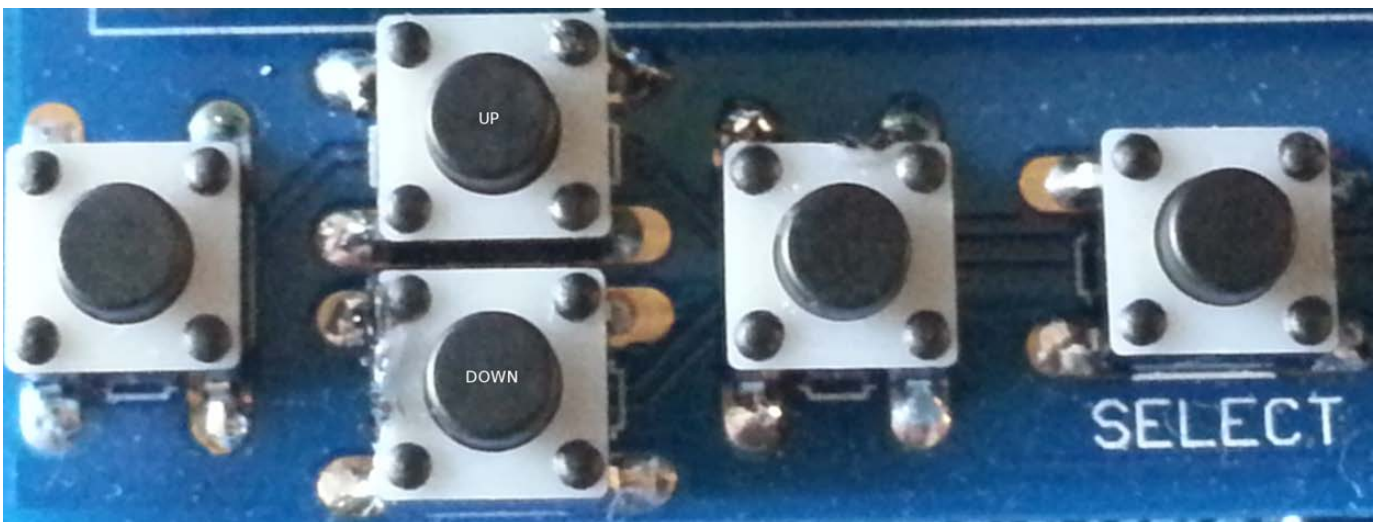


Figure 2. Pi Plate LCD Keys (photo by Jonathan Brogdon)

Listing 2. Light Controller UI Processing Module

```

from lcdproc.server import Server

#####
# LCD UI controller for the DMX controller
#####

class DmxUi():
    def __init__(self,color_list):
        self.color_list=color_list
        self.lcd = Server("127.0.0.1", debug=False)
        self.lcd.start_session()
        self.lcd.add_key("Up",mode="exclusively")
        self.lcd.add_key("Down",mode="exclusively")
        self.lcd.add_key("Enter",mode="exclusively")

        #Allocate the screen
        self.screen = self.lcd.add_screen("DMX")
        self.screen.set_heartbeat("off")

        #Add a widget for the label
        self.label_widget =
        ↪self.screen.add_string_widget("label_widget",
                                     text="",x=1,y=1)

        #Add a widget for the color
        self.color_widget =
        ↪self.screen.add_string_widget("color_widget",
                                     text="",x=7,y=1)

        #Add a widget to display the "selected" status
        self.not_set_widget =
        ↪self.screen.add_string_widget("not_set_widget",
                                     text="",x=16,y=1)

        #Set the label text
        self.label_widget.set_text("Color:")

        self.color_idx=0
        self.current_color_idx=0
        self.color_widget.set_text(self.color_list[self.color_idx][0])
        self.num_colors = len(self.color_list)

        # Get a key from LCDproc
        def get_key(self):
            resp = self.lcd.poll()
            if (resp == None):
                return None

            return resp[4:-1]

        # UI processing
        # -get keyinput
        # -update display
        # -return the current selection
        def ui_process(self):

            key_press = self.get_key()

            if (key_press==None):
                return None

            if (key_press == "Up"):
                self.color_idx -= 1

            if (key_press == "Down"):
                self.color_idx += 1

            self.color_idx %= self.num_colors

            if (key_press == "Enter"):
                self.current_color_idx = self.color_idx

            if (self.color_idx != self.current_color_idx):
                self.not_set_widget.set_text("**")
            else:
                self.not_set_widget.set_text("")

            self.color_widget.set_text(self.color_list[self.color_idx][0])

            return self.current_color_idx

```

the key buttons. For this project, I used the “up” and “down” keys to cycle through light color options and

the “select” key to select the desired color output (Figure 2).

Listing 2 shows the UI processing for

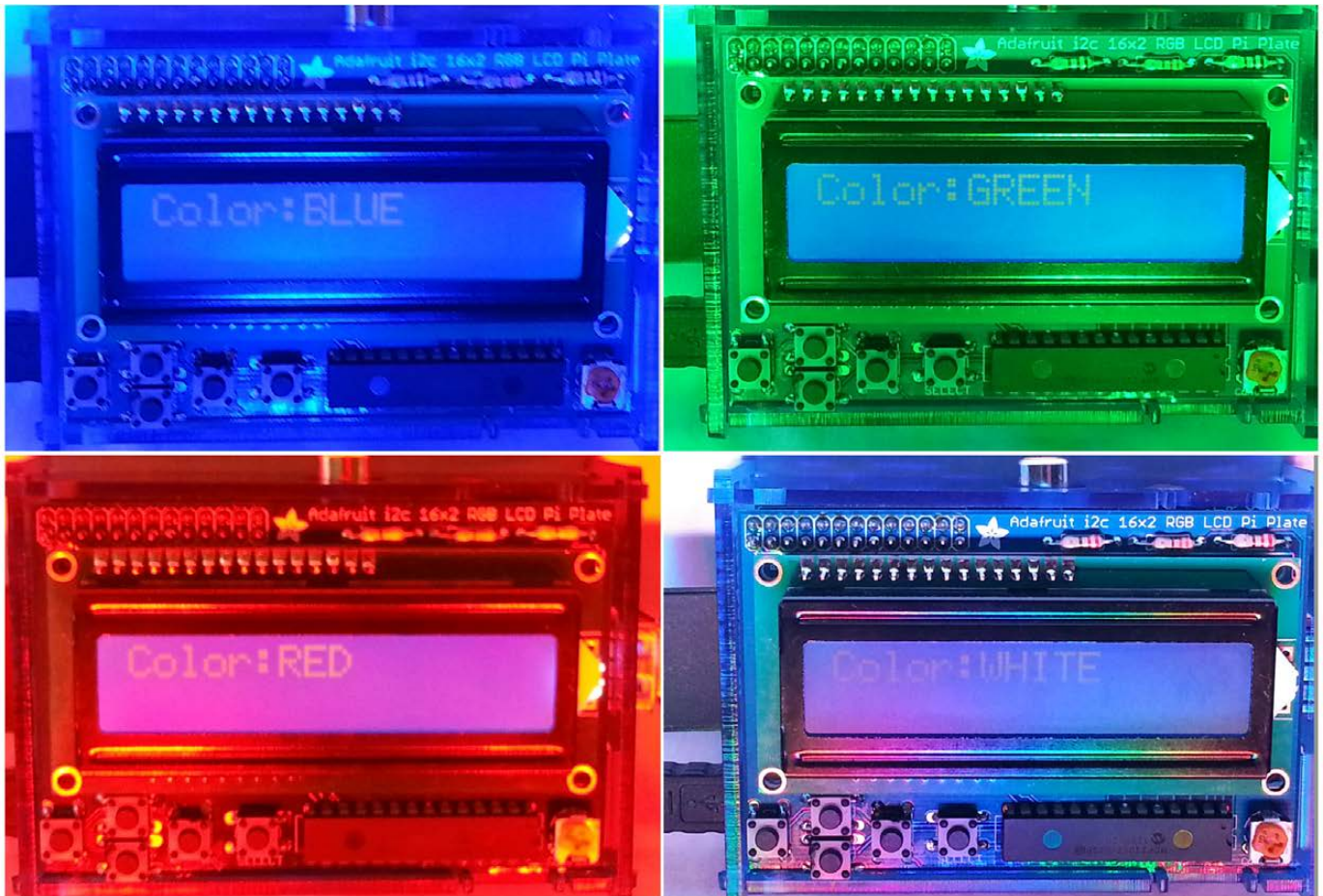


Figure 3. Collage of Color Selections (photo by Jonathan Brogdon)

the light controller. This code opens the LCDproc server connection, defines the UI widgets for the LCD, displays the light color options, processes the key button input and updates the UI widgets based on key inputs.

The light controller application is packaged as a Python script with a start-stop-dæmon script to run the application at system initialization. The Python script is shown in Listing 3 and is available at https://github.com/jlbrogdon/dmx_controller.

The application has two concurrent threads: the UI thread and DMX512 controller thread. The UI thread runs the LCDproc client shown in Listing 2. When a new light color is selected, the UI thread builds a complete DMX512 protocol message, with channel values corresponding to the user-selected light color and queues the message to the DMX512 controller thread.

The DMX512 controller thread initializes the FT232R device for DMX512 communication and

Listing 3. Light Controller Application Module

```

from DmxUi import DmxUi
from OpenDmxUsb import OpenDmxUsb
import Queue
import threading
import signal

# Thread lock serves as a signal to thread to terminate
termLock = threading.Lock()

# Tuple list of colors and corresponding
# Colorstrip channel value
static_color = [('RED' ,10),
                ('GREEN' ,20),
                ('BLUE' ,30),
                ('YELLOW' ,40),
                ('MAGENTA' ,50),
                ('CYAN' ,60),
                ('WHITE' ,70)]

#Asynchronous signal handler. Terminates threads.
def sig_handler(signum,frame):
    termLock.acquire()

#####
# DMX Control thread
#####
class DmxThread(threading.Thread):
    def __init__(self,dmxUsb,queue,tlock=None):
        threading.Thread.__init__(self)
        self.dmxUsb = dmxUsb #DMX/USB controller
        self.queue = queue #DMX data queue
        self.tlock = tlock #thread termination lock

    def run(self):
        channelVals = None
        while (False==self.tlock.locked()):
            #Check for new DMX data
            if (self.queue.empty() == False):
                #New values available
                channelVals = self.queue.get()
                self.queue.task_done()
            if channelVals != None:
                #Send the data to the devices
                self.dmxUsb.send_dmx(channelVals)

#####
# LCD UI thread
#####
class UiThread(threading.Thread):
    def __init__(self,dmx_ui,color_list,queue,tlock=None):
        threading.Thread.__init__(self)
        self.dmx_ui = dmx_ui #DMX UI device
        self.color_list = color_list #Color list
        self.queue = queue #DMX data queue
        self.tlock = tlock

    def run(self):
        channel_vals = bytearray([0]*513)
        channel_vals[0]=0 #dummy channel

        #Set Initial value
        channel_vals[1]=self.color_list[0][1]

        #Send the DMX data
        self.queue.put(channel_vals)

        while (False==self.tlock.locked()):
            #Check for UI input
            color_idx=self.dmx_ui.ui_process()
            if (color_idx != None):
                #New color value input
                channel_vals[1]=self.color_list[color_idx][1]
                #Send the new DMX data
                self.queue.put(channel_vals)

if __name__ == "__main__":
    #Install signal handler for SIGTERM
    signal.signal(signal.SIGTERM,sig_handler)

    #Queue for new DMX data
    channelsQueue=Queue.Queue()

    #DMX/USB controller
    dmx_usb = OpenDmxUsb()

    #UI controller
    dmx_ui = DmxUi(static_color)

    #DMX/USB controller thread
    dmx_thread = DmxThread(dmx_usb,
                           channelsQueue,
                           tlock=termLock)

    #UI controller thread
    ui_thread = UiThread(dmx_ui,
                        static_color,
                        channelsQueue,
                        tlock=termLock)

    #Start the DMX/USB controller thread
    dmx_thread.setDaemon(True)
    dmx_thread.start()

    #Start the UI controller thread
    ui_thread.setDaemon(True)
    ui_thread.start()

    #Wait for SIGTERM
    signal.pause()

    #Wait for threads to terminate
    dmx_thread.join()
    ui_thread.join()

```

Register Now!

2013 USENIX Federated Conferences Week

June 24–28, 2013 • San Jose, CA

www.usenix.org/conference/fcw13

USENIX ATC '13

2013 USENIX Annual
Technical Conference
Wednesday–Friday, June 26–28
www.usenix.org/atc13

ICAC '13

10th International Conference on
Autonomic Computing
Wednesday–Friday, June 26–28
www.usenix.org/icac13

HotPar '13

5th USENIX Workshop on
Hot Topics in Parallelism
Monday–Tuesday, June 24–25
www.usenix.org/hotpar13

UCMS '13

2013 USENIX Configuration
Management Summit
Monday, June 24
www.usenix.org/ucms13

Feedback Computing '13

8th International Workshop on
Feedback Computing
Tuesday, June 25
www.usenix.org/feedback13

ESOS '13

2013 Workshop on
Embedded Self-Organizing Systems
Tuesday, June 25
www.usenix.org/esos13

HotCloud '13

5th USENIX Workshop on
Hot Topics in Cloud Computing
Tuesday–Wednesday, June 25–26
www.usenix.org/hotcloud13

WiAC '13

2013 Women in Advanced
Computing Summit
Wednesday–Thursday, June 26–27
www.usenix.org/wiac13

HotStorage '13

5th USENIX Workshop on
Hot Topics in Storage and
File Systems
Thursday–Friday, June 27–28
www.usenix.org/hotstorage13

HotSWUp '13

5th Workshop on Hot Topics
in Software Upgrades
Friday, June 28
www.usenix.org/hotswup13

Registration Discounts Available!

Register by the Early Bird Deadline,
Monday, June 3, and save!

AND MORE!

Stay Connected...



www.twitter.com/usenix



www.usenix.org/youtube



www.usenix.org/facebook



www.usenix.org/linkedin



www.usenix.org/gplus



www.usenix.org/blog



continuously checks for any new protocol messages from the UI thread. Any new protocol message received from the UI thread is sent repeatedly to the DMX512 slave devices. Recall that when the controller stops sending protocol messages, the slave turns off the lights. Figure 3 shows a collage of the various output colors.

Many of the DMX512 light control systems available today provide a lot of configurable options for the user, which also adds a learning curve and potentially many more levers for you to pull during the stage show. However, if you are looking for something simple

to control a few stage lights for your band, small stage production or holiday lighting display, this Raspberry Pi-based light controller has the benefits of simplicity, extensibility, customization and low cost. One of my ultimate goals is to add a song selection menu and have the controller sequence the lights according to values in a corresponding file. Until then, rock on! ■

Jonathan Brogdon is a software team manager at Ixia, working on timing and synchronization products. He lives in Austin, Texas, with his wife and three rock-star kids. His interests include embedded Linux, M2M software, the Internet of Things and just about any development board he can get his hands on.

The White Paper Library on LinuxJournal.com



SOUTH EAST LINUX FEST 2013

LINUX IN THE GNU/SOUTH



BLAKE HOTEL & CONVENTION
CHARLOTTE, NC
JUNE 7-9, 2013

864-832-SELF (7353)

SOUTH EAST LINUX FEST.ORG
INFO@SOUTH EAST LINUX FEST.ORG

Effects of Cloud Computing on Open-Source Compliance

What you should know if you are deploying open source in the cloud. **DIANA MARINA COOPER**

Since the emergence of strong cloud service providers like Amazon Web Services, Google and Rackspace, software development and deployment is increasingly taking place in the cloud. According to Gartner, cloud computing is expected to grow at a rate of 19% this year. Big industry players including Netflix and eBay already have turned to the cloud for significant proportions of their operations and offerings. And in the next few years, we are likely to see more and more innovative startups like Coupa completely suspended in the cloud, relegating on-premise computing to a vestige of

a bygone era.

While enterprises are shifting from legacy solutions toward the cloud, open-source software is gaining significant traction for similar reasons. Gartner projects that 99% of Global 2000 companies will incorporate open source into their operations by 2016. Adopters of both cloud and open-source solutions are drawn toward the increased potential for collaboration and lower total cost of ownership.

The proliferation of open-source cloud projects (think OpenStack, CloudStack, Eucalyptus) and increasing use of open-source

software within the cloud suggests a need for enterprises to understand how the cloud environment impacts open-source license compliance. Before the emergence of the cloud, restrictive open-source licenses maintained software freedom through the regulation of distribution. However, because software is provided as a service in the cloud, licensing obligations that are linked to the act of distribution no longer apply. This has led to the development of newer cloud-driven restrictive open-source licenses, such as the AGPL. The game-changing effect of the cloud on traditional open-source compliance mechanisms and the subsequent development of remedial open-source licenses calls for organizations to audit and update their intellectual property policies to minimize the risk of infringement.

The Traditional Proprietary vs. Open-Source Battle and the Rise of Permissive and Restrictive Licenses

The emergence of cloud computing and its impact on open-source compliance has reignited the historical battle between proprietary and open-source software, and reinforced traditional divisions within the Open Source community.

The genesis of the proprietary vs. open-source debate dates back to the unbundling of IBM in the mid-1970s, after which it was no longer possible for users to access and modify code. Although user freedoms were removed through the process of unbundling, programmers continued to find ways to access, modify and share code, famously prompting Bill Gates to write his “Open Letter to Hobbyists” after Microsoft’s Basic was leaked.

During the late 1970s and early 1980s, the Open Source movement emerged in two distinct factions, the first of which was headed by Richard Stallman, a former programmer at the MIT Artificial Intelligence Lab. Stallman’s belief that the ability to access, modify and redistribute code is a fundamental freedom led to his development of the GNU project, which was licensed under the GPL—a restrictive license specifically designed to ensure that GNU code could not be rendered proprietary when incorporated in derivative works.

Around the same time, the BSD UNIX system was being developed by the Computer Science Research Group at Berkeley. In the late 1990s, the BSD UNIX became available under the BSD license. While Stallman’s GPL was designed

as a restrictive copyleft license aimed at preventing the underlying code from becoming proprietary, the BSD was drafted as a permissive license that would enable users to embed the underlying code into proprietary offerings.

Permissive vs. Restrictive Open-Source Licenses in the Pre-Cloud Environment

Licenses that cover open-source code carry unique terms that have implications on code use, modification and distribution. As previously mentioned, there are two broad categories of open-source licenses—the permissive and restrictive types. Permissive licenses, such as the MIT and BSD licenses, provide minimal obligations on code use, modification and distribution, enabling developers to incorporate open-source code into proprietary software, which they then could protect by adding additional license terms.

In contrast, restrictive open-source licenses, such as the GPL, do not allow users of covered code to release derivative works under different license terms. In addition, these restrictive licenses require users that distribute modified programs to make their source code available to downstream users,

in order to maintain the copyleft community's goal of achieving software freedom. This concept of software freedom refers to the right of all downstream users to access, run, modify and redistribute software containing the covered code. This feature of restrictive licenses renders it impossible to incorporate open-source code into proprietary offerings. There is no way to avoid these stringent rules, and the failure to comply with such obligations can lead to severe consequences, including being forced to come into compliance by releasing the asset's source code or paying damages for intellectual property infringement.

In the pre-cloud environment, software vendors made their products available to end users through software distribution. Because there was no other means of making software available to users, it was impossible for vendors to escape the distribution clauses in restrictive open-source licenses. However, this has changed with the introduction of cloud computing.

Cloud Computing's Challenge to the Distribution-Based GPL Model

Restrictive open-source licenses, such as the GPL, operate to maintain

However, because software is not distributed in the cloud—it's simply made available to users as a service—providers do not have to pay these freedoms forward.

software freedom only to the extent that the underlying open-source code is part of a distribution. For example, the GPLv3 states that:

You have certain responsibilities if you distribute copies of the software: responsibilities to respect the freedom of others. If you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code.

Before the emergence of the cloud, this license term ensured that any time software incorporating covered code was deployed to third parties, that distribution would be governed by the GPL terms such that the distributor would be forced to make its code available to users. However, the proliferation of cloud-based SaaS solutions threatened to destabilize the GPL model by creating

an environment in which for the first time software was made available to users without being distributed.

GPL: Permissive within the Cloud

In instances where software containing GPL code is made available through network services, the distribution clause is bypassed, and the provider does not have to release its source code. Remember the free software reciprocity trigger: "If you distribute copies of such a program...you must pass on to the recipients the same freedoms that you received." However, because software is not distributed in the cloud—it's simply made available to users as a service—providers do not have to pay these freedoms forward. Rather, they can access the benefits of using free software without being forced to provide those same benefits to their users. This loophole enables SaaS enterprises to embed GPL-covered code into proprietary cloud offerings. Effectively what this means is that,

within this distribution-free model, the GPL assumes the attributes of a permissive license (think MIT, BSD).

AGPL: the Open-Source Empire Strikes Back

For anyone who thought that the cloud rendered the proprietary and open-source debate moot, think again—the battle is far from over, it simply relocated to another frontier. Before long, the copyleft faction of the Open Source movement regrouped and responded to the threat that the cloud-based SaaS model posed to its goal of maintaining software freedom. The weapon of choice that the movement developed and deployed to respond to the unique challenges imposed by the emerging cloud-based SaaS environment was the Affero GPLv3 (AGPLv3), which covers popular applications, such as PHP-Fusion, Launchpad and SugarCRM.

Unlike the GPL, which relies on the act of distribution to trigger the free software reciprocity clause, the AGPLv3 includes the following term that was articulated specifically for situations in which software is used on a network but is not technically distributed. This clause states that:

If you modify the program, your modified version must prominently offer all users interacting with it remotely through a computer network (if your version supports such interaction) an opportunity to receive the corresponding source of your version by providing access to the corresponding source from a network server at no charge, through some standard or customary means of facilitating copying of software.

This license term applies the distribution-based reciprocity clause to cloud-based software offerings in which users run programs from remote servers.

AGPL in the Private Cloud

The AGPL was drafted as a solution to the problem that the public cloud created. Its preamble states that whereas the GPL “permits making a modified version and letting the public access it on a server without ever releasing its source code to the public...the AGPL is designed specifically to ensure that, in such cases, the modified source code becomes available to the community.” But what happens if an organization uses AGPL code

internally? The remote network interaction clause states that:

If you modify the program, your modified version must prominently offer all users interacting with it remotely through a computer network an opportunity to receive the corresponding source of your version...through some standard customary means of facilitating copying of software.

It appears that the same principle applies in both the public and private cloud contexts—any users have the right to access the modified code and to create their own versions. In the private cloud scenario, these freedoms would extend to any employees, contractors and other parties using the server.

Consequences of Non-Compliance with Open-Source License Obligations

The failure to comply with open-source license obligations can lead to severe consequences, including being forced to come into compliance by releasing the modified code and paying damages. Non-compliant organizations are

exposed to risk, as courts in various jurisdictions including the United States, Germany and France have consistently ruled that open-source licenses are enforceable, leading to a proliferation of open-source litigation and settlements.

One of the earlier infringement suits that solidified the enforceability of open-source software resulted from the acquisition of Linksys by Cisco in 2003. Shortly after the acquisition, Cisco was sued for infringement relating to the use of GPL-covered code in its router firmware. It turned out that the infringing chipset was provided to Linksys by Broadcom, which in turn outsourced the development to a third party. As a part of the settlement that was reached between the parties, Cisco was forced to make the infringing code available on its Web site, appoint an open-source compliance officer and make a monetary contribution to the Free Software Foundation.

BusyBox also launched a string of successful infringement suits against companies that incorporated its code and leveraged the resulting assets in violation of the GPL. The first of these involved the use of BusyBox code in embedded systems provided by Monsoon Multimedia,

Inc. BusyBox alleged that Monsoon utilized BusyBox code without making its modified code available to downstream users pursuant to the GPL. The parties settled for an undisclosed amount, and Monsoon agreed to publish its code and appoint an open-source compliance officer. A similar settlement was reached between BusyBox and Verizon Communications. More recently, BusyBox filed a suit against 14 electronics suppliers, including Samsung and Best Buy, alleging that the defendants distributed devices containing BusyBox code without making their modified code available to users. While some of these defendants opted to settle, in the case of Westinghouse, a District Court in New York found in favor of the plaintiff. In that case, the Court determined that Westinghouse willfully infringed BusyBox's copyright in the code, and consequently the damages were tripled.

The proliferation of open-source infringement suits and resulting settlements have solidified the enforceability of open-source software. Because of the immense financial and reputational damage that is associated with intellectual property infringement suits, it is crucial for organizations to ensure

compliance with open-source license obligations. Although the cloud environment poses new uncertainties for organizations relying on open-source software, there are various tools that can be engaged to minimize the risk of non-compliance.

How to Transition Your Organization into the Cloud

Given the new obligations imposed by the AGPLv3, it is critical for cloud-based SaaS providers to take inventory of the open-source code embedded in their product offerings and to ensure that their intellectual property policies are in line with the obligations imposed by the various open-source licenses covering the code being used. A variety of tools are available that can assist SaaS enterprises to ensure open-source compliance in the cloud. For example, enterprises can scan their software with tools that are specifically designed to detect open-source code and provide a list of the license obligations that accompany each component. In addition, a structured Open Source Software Adoption Process (OSSAP) can be used to define acceptable intellectual property license policies for the organization, audit the current software

portfolio and incoming code, and ensure compliance through all of the software development and procurement stages.

Open-source license management solutions now are accessible to companies in the cloud. Because these solutions are hosted in the cloud environment, they eliminate the need for enterprises to install or update code-scanning software. Instead, companies can sign up with a service provider and are given access to software that scans their code, identifies open source and provides a breakdown of the associated license obligations. Such open-source license management services are invaluable to SaaS enterprises, particularly given the uncertainties associated with open source in the cloud. In addition to ensuring that organizations understand and are able to meet their open-source license obligations, these management solutions position enterprises to respond efficiently and effectively to any instances of non-compliance that are detected. For example, by understanding which components of the software are used in a non-compliant fashion, SaaS enterprises are positioned to replace the infringing code with code that

offers similar functionality or to adapt their policies to ensure adherence to obligations.

Conclusion

The emerging cloud-based SaaS model offers immense opportunities but also raises new risks for organizations in relation to intellectual property infringement. Various open-source license management solutions are available to assist enterprises in making a safe transition into the cloud. For enterprises planning on navigating the cloud environment—and for those that already have made the migration—it is important to take an inventory of the code incorporated in the software being offered and to determine if open-source licensing obligations are being met. Keep in mind that the intellectual property policies that were developed for the traditional software distribution model will need to be assessed and updated to meet the distinct obligations associated with the cloud environment. ■

Diana Marina Cooper obtained a BA in Politics and Governance and an MA in Globalization Studies. She is currently a JD Candidate (2013) and is pursuing a concentration in Law and Technology. She has been working with Protecode (<http://www.protecode.com>) as an open-source corporate strategy consultant since 2011. Follow Diana: @Diana_M_Cooper.



DOC SEARLS

One Hand Slapping

The upper hands of big-tech business are changing their game.

Adobe Systems, the world's leading supplier of graphical software, is gradually shifting its business to a subscription model, baiting customers with features and tools available only to Creative Cloud subscribers (<http://html.adobe.com/edge/creativecloud>). Among the first of these is Edge Reflow (<http://html.adobe.com/edge/reflow>), which Adobe expects to become a must-have for Web designers—just as Photoshop, Illustrator and other tools in Adobe's Creative Suite are long-standing must-haves for many photographers and graphic designers. The difference is that you *own* Creative Suite when you buy it. You only *rent* Creative Cloud when you subscribe to it.

Microsoft has been doing the same thing with Office 365 (<http://office.microsoft.com/en-us>). Originally built for enterprise customers, Office 365 now is

available for “home” customers as well. “Your complete Office in the cloud”, will work on “up to 5 PCs or Macs”, and is priced at \$99.99/year or \$9.99/month. That's competitive with the price of Microsoft's boxed Office software, which becomes stale as soon as the next major version comes out. And, like Adobe's Creative Cloud, Office 365 baits customers with lots of features lacking in the old-fashioned sold versions of the suite.

What both want you to rent is SaaS: Software as a Service.

The tide of history is with them. Nicholas G. Carr's *The Big Switch* (<http://www.nicholasgarr.com/bigswitch>) accurately predicted a shift to “computing as a utility” back in 2004, long before utility services became known as “the cloud”. This always has made sense for enterprises, which can save IT costs by renting software as well as compute and

The difference is that you own Creative Suite when you buy it. You only rent Creative Cloud when you subscribe to it.

data storage services. We also have a model for SaaS at the individual level, in the form of smartphone apps that are either user interfaces for cloud-based services or highly tethered to clouds for updates and data flows. While Wikipedia lists a few open-source iOS (http://en.wikipedia.org/wiki/List_of_open_source_iOS_applications) and Android (http://en.wikipedia.org/wiki/List_of_open_source_Android_applications) apps, those are just drops in an ocean of proprietary apps and services.

In fact, even the packages of boxware you buy from Adobe and Microsoft are less owned than licensed. The collection of bits that come in a box, or are downloaded from a virtual store, are not tangible property. Instead, what you have is a bundle of rights for usage, governed by a seller to which the software remains attached by a service contract. So, the thinking then goes, why bother with old-fashioned ownership at all?

Why not just rent?

We already rent many other things in the digital world, such as domain names. Here at *Linux Journal*, we don't own LinuxJournal.com. We rent it. The same goes for all the other domain names in the world. Every one is rented for finite periods of time. They can be bought or sold, but they can be possessed only so long as the owner keeps paying a domain registrar for the right to use them exclusively—meaning we rent them.

Changing software into a subscription business is a slippery slope down which control of software moves from individual owners to corporate suppliers.

Predictably and correctly, Richard M. Stallman began saying “the cloud” was “a trap” in 2008 (in this *Guardian* interview: <http://www.guardian.co.uk/technology/2008/sep/29/cloud.computing.richard.stallman>).

Here's a sample:

But there has been growing concern

So, the thinking then goes, why bother with old-fashioned ownership at all? Why not just rent?

that mainstream adoption of cloud computing could present a mixture of privacy and ownership issues (<http://www.guardian.co.uk/technology/blog/2008/aug/06/whengoogleownsyouyourdata>), with users potentially being locked out of their own files.

Stallman, who is a staunch privacy advocate, advised users to stay local and stick with their own computers.

“One reason you should not use Web applications to do your computing is that you lose control”, he said. “It’s just as bad as using a proprietary program. Do your own computing on your own computer with your copy of a freedom-respecting program. If you use a proprietary program or somebody else’s Web server, you’re defenseless. You’re putty in the hands of whoever developed that software.”

Free software and open-source principles and methods don’t fit well with cloud-based models of

software development, deployment and money-making.

One name for the end-state of this shift is the “subscription economy”. Here at *Linux Journal*, we’ve been living in the subscription economy, as have all subscription periodicals. We also have done our best from the start to make our goods as free (as in freedom) as possible. We’re not perfect at that, and we may never be, but free is where we come from. (In fact, *Linux Journal* originally was conceived, way back in 1993, as a free software magazine.) Meanwhile, most of the rest of the world lacks the same consciousness. Instead, the general consciousness has moved backward in time. In “When It Comes to Security, We’re Back to Feudalism” (<http://www.wired.com/opinion/2012/11/feudal-security>), Bruce Schneier says we “pledge allegiance to the united states of convenience” when we become “vassals” to the “feudal lord” called Google, Amazon, Facebook and Apple. He explains:

Feudalism provides security.

Classical medieval feudalism depended on overlapping, complex, hierarchical relationships. There were oaths and obligations: a series of rights and privileges. A critical aspect of this system was protection: vassals would pledge their allegiance to a lord, and in return, that lord would protect them from harm....And it's this model that's starting to permeate computer security today.

Traditional computer security centered around users. Users had to purchase and install anti-virus software and firewalls, ensure their operating system and network were configured properly, update their software, and generally manage their own security. This model is breaking, largely due to two developments:

1. New Internet-enabled devices where the vendor maintains more control over the hardware and software than we do—like the iPhone and Kindle; and
2. Services where the host maintains our data for us—like Flickr and Hotmail.

Now, we users must trust the security of these hardware manufacturers, software vendors and cloud providers. We choose to do it because of the convenience, redundancy, automation and shareability.

He doesn't mention the phone and cable companies that intermediate our connections over the Internet, but their hearts are feudal to the core, and their enclosure plans are clear. While on the one hand they increase speeds and give us other benefits, they also have a highly vested interest in transforming the Internet from a free and open space to an enclosed one from which rents can be extracted to the max—just like they always were in telephony and cable television. At risk is the Net itself.

Ten years ago, in my *LJ* article "Saving the Net" (<http://www.linuxjournal.com/article/6989>), I began with this:

Who Owns What?

That's the fundamental question....Much is at stake, including Linux and its natural habitat: the Net. Both have

If your ISP's narc-ware detects what it thinks is copyright infringement happening over your Internet data flows, they can punish you.

been extraordinarily good for business. Its perceived "threat" to Microsoft and the dot-com crash are both red herrings. Take away Linux and the Net, and both technology and the economy would be a whole lot worse.

Both the Net and Linux were created, grew and flourished almost entirely outside the regulatory sphere. They are, in a literal sense, what free markets have done with their freedoms.

Yet, there are some who do not care. Unfortunately, they're driving the conversation right now. Hollywood has lawmakers and news organizations convinced that file sharing is "piracy" and "theft". Apple, Intel and Microsoft are quietly doing their deals with the Hollywood devil, crippling (or contemplating the crippling of) PC functionalities, to protect the intellectual property of "content producers".

Where we are headed is toward Hollywood's model of the Net: one where we pay on a subscription basis for everything that matters, and where everybody—creators and consumers alike—need constantly to "clear rights" for usage, whether they know that's what they're doing or not. Here's more from my "Saving the Net" article:

Two oddly allied mentalities provide intellectual air cover for these threats to the marketplace. One is the extreme comfort certain industries feel inside their regulatory environments. The other is the high regard political conservatives hold for successful enterprises. Combine the two, and you get conservatives eagerly rewarding companies whose primary achievements consist of successful long-term adaptation to highly regulated environments.

Hollywood's latest subscriber-screwing success story is the Copyright Alert

System (http://en.wikipedia.org/wiki/Copyright_Alert_System), better known as “six strikes”. If your ISP’s narc-ware detects what it thinks is copyright infringement happening over your Internet data flows, they can punish you. The details of both detection and punishment matter less than the fact that they are spying on your private activities and can strangle your Net connection. And they can easily do that, because they have the upper hand.

So, instead of two hands clapping, we have one hand slapping.

Linux is in that hand, because Linux is in pretty much everything. Same with the Internet. Both have won, yet the promise of freedom, which drove Linux and the Net to success, is being traded for away for convenience.

Eternal vigilance is the price of liberty, the saying goes ([http://wiki.monticello.org/mediawiki/index.php/Eternal_vigilance_is_the_price_of_liberty_\(Quotation\)](http://wiki.monticello.org/mediawiki/index.php/Eternal_vigilance_is_the_price_of_liberty_(Quotation))). And, right now, the silence is killing us. ■

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.

Advertiser Index

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

ADVERTISER	URL	PAGE #
1&1	http://www.1and1.com	17
ANDEVCON SPRING	http://www.AnDevCon.com	31
CLOUD WORLD FORUM	http://www.cloudwf.com/	53
EMAC, INC.	http://www.emacinc.com	23
EMPERORLINUX	http://www.emperorlinux.com	33
IXSYSTEMS, INC.	http://www.ixsystems.com	7
KINGSTAR USA	http://www.kingstarusa.com	28, 29
MANAGE ENGINE	http://www.manageengine.com	51
O'REILLY VELOCITY	http://velocityconf.com/sc	2
OVH	http://www.ovh.com/us/index.xml	10, 11
SILICON MECHANICS	http://www.siliconmechanics.com	3
SOUTHEAST LINUX FEST	http://SouthEastLinuxFest.org	97
TEXAS LINUX FEST	http://2013.texaslinuxfest.org/	112
USENIX FEDERATED CONFERENCES WEEK	http://www.usenix.org/conference/fcw13	95

ATTENTION ADVERTISERS

The *Linux Journal* brand's following has grown to a monthly readership nearly one million strong. Encompassing the magazine, Web site, newsletters and much more, *Linux Journal* offers the ideal content environment to help you reach your marketing objectives. For more information, please visit <http://www.linuxjournal.com/advertising>.



TEXAS LINUX FEST

COME BE A PART OF TXLFF 2013:
THE LARGEST LINUX AND OPEN SOURCE
SOFTWARE CONFERENCE IN THE
LONE STAR STATE!



MAY 31-JUNE 1



THE AT&T CONFERENCE CENTER
IN AUSTIN, TEXAS

**MORE INFO AND REGISTRATION AT
[HTTP://TEXASLINUXFEST.ORG](http://TEXASLINUXFEST.ORG)**

