

LINUX JOURNAL

Since 1994: The Original Magazine of the Linux Community

**Control
Your Own
Data
with the
Personal Cloud**

OCTOBER 2013 | ISSUE 234 | www.linuxjournal.com

EMBEDDED

**DIAGNOSE
YOUR CAR
WITH ANDROID**

**TEST NEW
BUILDS
QUICKLY
WITH A
U-BOOT TRICK**

**BUILD A
PERSONAL
COMPUTER
OR SERVER
WITH AN
EMBEDDED
SYSTEM**

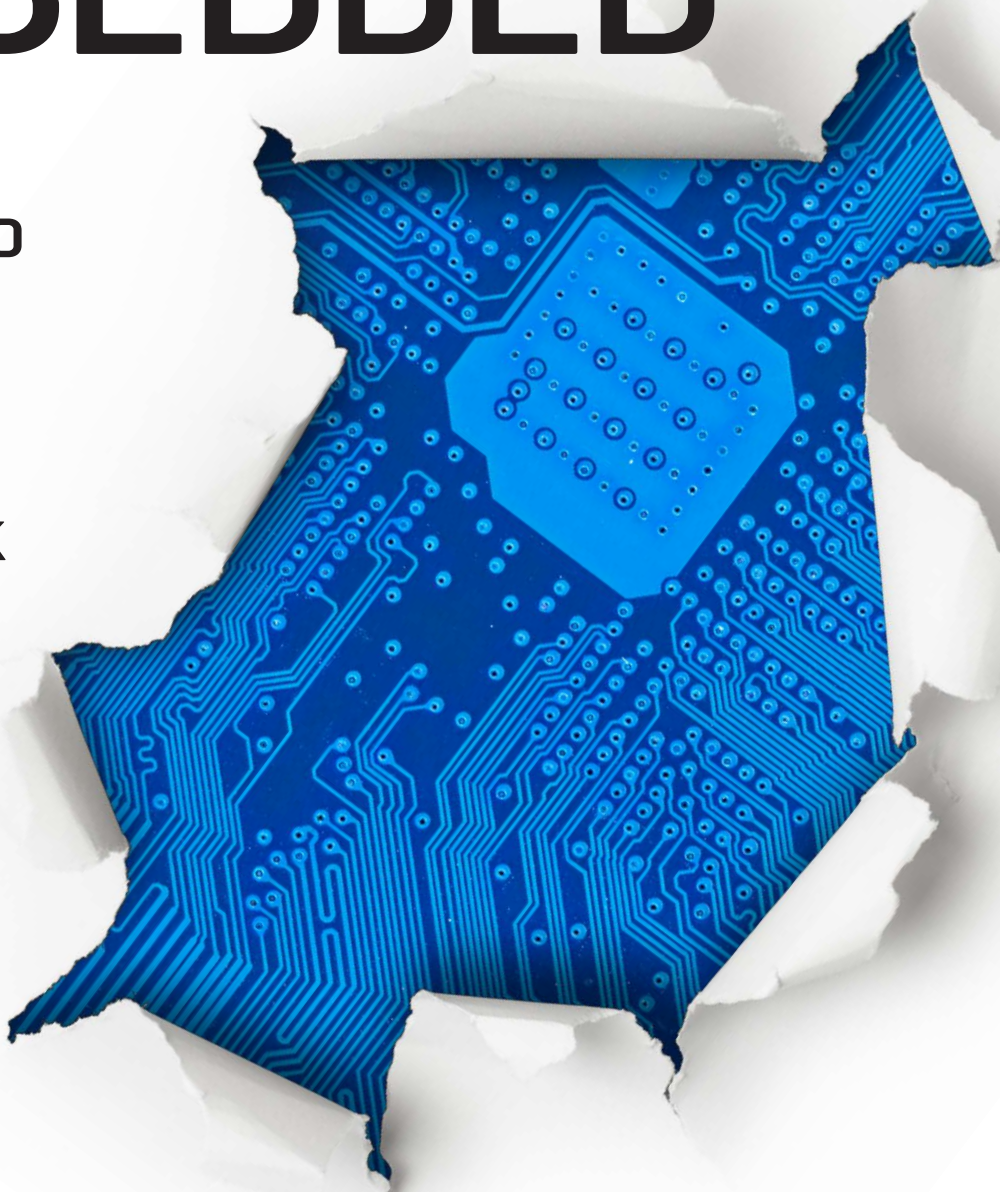


Shell Scripts and
ImageMagick

HOW-TO:
Set Up Your
Own BirdCam

rss2email
A Simple Approach
for Viewing RSS Feeds

ZURB FOUNDATION
Design a Responsive
Web Site



Attend the Largest Dedicated Android Conference in the Universe!

AnDevCon SAN FRANCISCO November 12-15, 2013

Registration
Now Open!

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 a great opportunity to take your Android skills to the next level, get exposed to technologies you haven’t touched yet, and network with some of the best Android developers in the world.”

—Joe Mitchell, Software Engineer, Quicken Loans

“It’s a blast learning and exchanging ideas with phenomenal speakers and cutting-edge experts who have the experience.”

—Brad Holmes, Software Developer, uShip



Register Early and Save at www.AnDevCon.com

AnDevCon™ is a trademark of BZ Media LLC. Android™ is a trademark of Google Inc. Google's Android Robot is used under terms of the Creative Commons 3.0 Attribution License.

A BZ Media Event  Follow us: twitter.com/AnDevCon

Accelerate Your App

Are you tired of watching a sad workstation chug through your data?

Sign up for a test drive on one of our GPU server solutions today. See how you can accelerate your code or applications with parallel processing on NVIDIA® Tesla® K20 GPUs.

<http://www.siliconmechanics.com/testdrive>



CONTENTS

OCTOBER 2013
ISSUE 234

EMBEDDED

FEATURES

64 **Be a Mechanic...with Android and Linux!**

Decode Your "Check Engine" light.

Bill Childers

74 **Create a Mini PC or Server with Olimex's Olinuxino A13/A13Micro**

Build a capable Linux personal computer or server with a cheap embedded system.

Ronald Kurniawan

91 **A Handy U-Boot Trick**

U-Boot over LAN.

Bharath Bhushan Lohray

ON THE COVER

- Control Your Own Data with the Personal Cloud, p. 104
- Diagnose Your Car with Android, p. 64
- Test New Builds Quickly with a U-Boot Trick, p. 91
- Build a Personal Computer or Server with an Embedded System, p. 74
- Shell Scripts and ImageMagick, p. 36
- How-To: Set Up Your Own BirdCam, p. 48
- rss2email: a Simple Approach for Viewing RSS Feeds, p. 42
- Zurb Foundation—Design a Responsive Web Site, p. 26

Cover Image © Can Stock Photo Inc. / pzAxe

INDEPTH

104 The Personal Cloud

What happens when you take all the functionality of enterprise commercial software and make it available to individuals? We're about to find out.

T.Rob

COLUMNS

26 Reuven M. Lerner's At the Forge

Zurb Foundation

36 Dave Taylor's Work the Shell

Image Manipulation with ImageMagick

42 Kyle Rankin's Hack and / Command-Line Cloud: rss2email

48 Shawn Powers' The Open-Source Classroom

It's a Bird. It's Another Bird!

114 Doc Searls' EOF The First Personal Platform—for Everything

IN EVERY ISSUE

8 Current_Issue.tar.gz

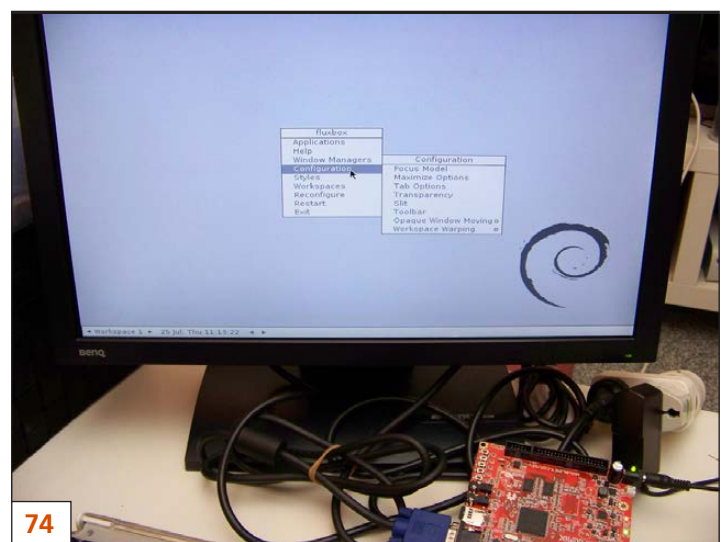
10 Letters

14 UPFRONT

24 Editors' Choice

60 New Products

117 Advertisers Index



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 • Adam Monsen

Publisher Carlie Fairchild
publisher@linuxjournal.com

Director of Sales John Grogan
john@linuxjournal.com

Associate Publisher Mark Irgang
mark@linuxjournal.com

Webmistress Katherine Druckman
webmistress@linuxjournal.com

Accountant Candy Beauchamp
acct@linuxjournal.com

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

PO Box 980985, Houston, TX 77098 USA

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

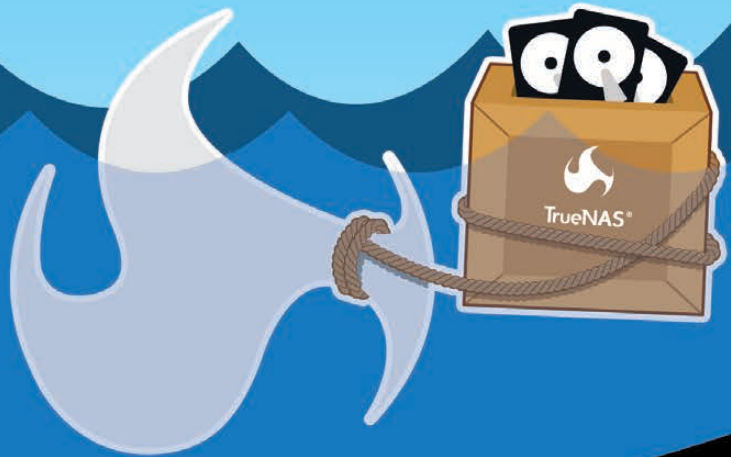
Subscriptions

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.



Caught **ADRIFT** in a sea of **STORAGE** options?

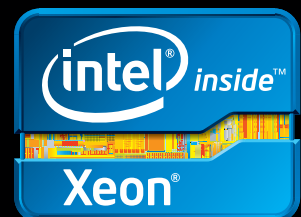


TrueNAS[®] has the tools to keep you afloat.

TrueNAS[®] Unified Storage features the Intel[®] Xeon[®] Processor 5600 series and supports high availability, remote replication, deduplication, encryption, compression, and snapshots. It has the tools to deal with any storage challenge you may face.

Key Features:

- Dual Intel[®] Xeon[®] Processors 5600 Series
- Support for CIFS, NFS, iSCSI, and more
- Active Directory, LDAP, and NIS integration
- Multi-Petabyte Scalability



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

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



SHAWN POWERS

Stealth Linux

My favorite scene from *The Karate Kid* (the original from 1984, sorry, I'm old) is when Mr Miyagi stops the Cobra Kai sensei, John Kreese, from beating up his defeated student. Miyagi is a small, feeble-looking old man, and he has no need to prove to anyone that he's an awesome warrior. Linux is much the same: awesome and unassuming. Linux is slowly but surely taking over the world, embedded into cell phones, televisions, wristwatches and server rooms. Because Linux itself isn't a commercial entity, it doesn't need to be flashy, and it doesn't need to brag. When it comes to embedded systems, Linux is "honking the nose" of proprietary alternatives, just like Mr Miyagi does to the bully.

We start out our Embedded issue with Reuven M. Lerner's column. This month, Reuven explores an alternative to the extremely popular Twitter Bootstrap. Zurb Foundation may sound like a character from *Starcraft*, but in fact, it's a competitor to Twitter Bootstrap that is worth a look. Kyle Rankin takes the path less traveled as well this month with his solution to

the Google Reader shutdown. One of the things I love about Linux is that there's no single way to do anything. While I want a bookmark to take me from RSS site to RSS site, Kyle shows us to have articles delivered via e-mail. If that sounds interesting to you, check out his column this month for instructions on how he does it.

Dave Taylor and I have a little crossover with our columns this month, which is completely coincidental, but very aptly timed. Dave explores the power of the ImageMagick suite of tools used in scripting. I follow later in the magazine with my column describing my homemade streaming Webcam of "BirdTopia", or "Backyard" as others refer to it. I use some ImageMagick tools to manipulate the images pulled from my cell phones, all from scripts on my Linux server. I've enjoyed creating my BirdCam setup more than most of my Linux projects, and I hope it inspires you to try something similar!

The truly embedded portion of this issue starts with Bill Childers and his article on interfacing with your

vehicle's OBD system. Using Android and a fancy dongle, Bill explains how to get diagnostic information directly from your car! If your "Check Engine" light suddenly pops on, but gives no reason as to why, Bill's article will be extremely helpful. Ronald Kurniawan goes one step deeper and shows how to use an \$80 embedded system from Olimex to create a desktop system, or even a complete server. Proving there's more to the tiny embedded world than just Raspberry Pi devices, Ronald walks through the entire process for getting a system running.

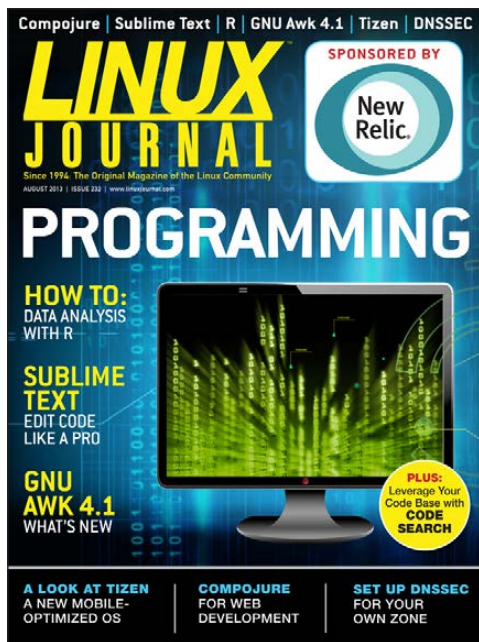
Most new embedded systems take advantage of the U-Boot system for starting up. Bharath Bhushan Lohray covers U-Boot and some of its features, along with a few tricks. Whether you need to boot your embedded device from an SD card, a USB stick or even over TFTP, U-Boot can make modifications as simple as a device reboot. Bharath walks through the process and gives configuration examples. And finally, T.Rob talks about the future of the Cloud—specifically, the Personal Cloud. While the fluid size and burstable expansion of cloud computing has revolutionized the way we think about server rooms, it also has moved sensitive data out of our personal control. How do we deal

with managing our personal, private data in a world focusing on selling services? T.Rob explores that and gives us a lot to consider.

Much like "cloud" computing, "embedded" computing has a fairly flexible definition. One thing is certain, however, and that is that Linux is perfect for the embedded world, however you define it. With its breathtaking variety of hardware support and unassuming happiness living behind the scenes, the embedded market may be the vehicle Linux finally uses to take over the world. Year of the desktop? Pshaw, more like year of "the everything else", with a little desktop on the side! Although Linux may not have the marketing campaign, or ad campaigns of the proprietary alternatives, much like Mr Miyagi, it doesn't need it. Just do the job, do it well, and the rest will fall into place. We hope you enjoy this issue (very likely reading it on your embedded device); we certainly enjoyed putting it together! ■

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

letters



Net Connectivity in Hotels

In Doc Searls' article "Dear Hotels: Quit Being A-holes" in the August 2013 issue, he writes: "In the US and Europe, the expensive hotels are the ones with inconvenient connectivity deals....It's the cheap hotels that offer free Internet, just like they offer free electricity, heat, air conditioning and running water."

I've recently stayed in youth hostels in northern Italy and in Alberta, Canada. They all had free Wi-Fi. Why would anyone stay in an expensive hotel?

And, I'm a "youth" of 66—most youth hostels seem to have no age limit.

—David Brown

Doc Searls replies: *Coincidentally, I am a youth of 66 as well. So let's toast staying young for the duration (clink!).*

As I said, mostly I stay in private homes (all with free Internet connections) when I'm paying my own way. And, I like AirBnB. But the youth hostel idea is also a good one, and even cheaper. Thanks!

Cerberus, I

In the August 2013 issue, Shawn Powers recommended the Cerberusapp application to track the location of your Android phone (see his UpFront piece titled "Android Candy: Hire a Cerberus to Find Your Phone"). This is indeed a great app; thanks for sharing.

However, you should be aware of the privacy policy (<https://www.cerberusapp.com/privacy.php>): "LSDroid uses your personally identifiable, location... information...to create new features, promotions, functionality and services....LSDroid uses cookies and log file information to...b) provide custom and personalized content, advertisements and information; c) monitor the effectiveness of our marketing campaigns....LSDroid

discloses aggregate, anonymous log file and usage information in reports to interested third parties....”

I just thought you and *LJ* readers should be aware of the privacy implications when using the app.

—Aviv

Ugh! Thanks for pointing that out. Creepy indeed.—Shawn Powers

Archive CD?

I already own the 1994–2009 *Linux Journal* Archive CD. Is there a way to just download/purchase the issues from 2010 onward, or do I have to purchase another archive disk? Also, do you plan to have another Linux and Amateur Radio issue? I enjoyed the January 2010 issue. P.S. I enjoy the articles in *Linux Journal*; keep up the good work.

—Micheal Trombley

The archive CDs are an all-or-nothing sort of thing. The intention isn't to resell the same thing every year, but rather to provide a full archive for those folks hoping to catch up. They're also nice for people like me who like to "collect the whole set", but I don't think there are any plans to release incremental updates. That said, subscribers have

access to the back issues—maybe that helps?—Shawn Powers

Google Reader


I saw Shawn Powers' article on Google Reader via the *Linux Journal* RSS feed ("The Google Giveth" in the May 2013 issue).

Go to <http://keepamericaatwork.com>. I decided to do this when Google

7" Panel PC

PPC-E7+

- ARM9 400Mhz Fanless Processor
- Up to 1 GB Flash & 256 MB RAM
- 7" 800 x 480 TFT LED Backlit 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 ports
- I2S Audio Interface w/ Line-in/out
- Operating Voltage of 12 to 26 Vdc
- Optional 2D Accelerated Video & Decoder
- Pricing start at \$550 for Qty 1



2.6 KERNEL

Designed and Manufactured in the USA the PPC-E7+ Compact Panel PC comes ready to run with the Operating System installed on Flash. Apply power and watch the Linux X Windows User Interface appear on the vivid 7" color LCD. Interact with the PPC-E7+ 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_e7+.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

[LETTERS]

Reader disappeared, so that I wouldn't be dependent on any system.

Basically, I host my own site, and I use a plugin called wp-o-matic that goes out and grabs the RSS feeds that I program it with.

Because I'm using WordPress, I can categorize those feeds (look on the right-hand sidebar for the categories), and now I can view articles by category or by searching. Most important, I can click on a day in the calendar and read all feeds for that particular day.

And, because I also have subscribed to Keep America At work via the e-mail subscription plugin, I automatically get a copy of everything that gets added via e-mail in case I become too busy to check the site itself. So far, it works like a champ.

The downside is that if you're subscribed via the e-mail plugin, and you add a bunch of new feeds, you will get a bunch of e-mails until it processes the new feeds, but then because most places release only one or two articles

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.

per day, it is very manageable.

—Virgil

Interesting concept. Over the past few months, I've written about a few ways to replace Google Reader, but creating a personalized blog that "writes itself" based on subscribed feeds...that's interesting. Of course, it makes me want to have you add your own RSS feed to your feed aggregation setup, to see if it blows itself up, but that's just the prankster in me coming out!—Shawn Powers

Cerberus, II

I often pass on Shawn Powers' "Android Candy" tips to my wife. I did so this month for the Cerberus app, but then almost immediately saw this story: "Google Unveils Android Version of 'Find My iPhone'" (<http://officialandroid.blogspot.com/2013/08/find-your-lost-phone-with-android.html>).

You might want to alert your users to this development in next month's *LJ*.

—Bob L.

Thanks Bob! This is especially interesting based on Aviv's letter regarding Cerberus. It does mean I'll be reading the fine print on the Google solution before installing though, that's for sure.—Shawn Powers

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

PHOTO OF THE MONTH

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

LINUX JOURNAL

At Your Service

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

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

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

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

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

diff -u

WHAT'S NEW IN KERNEL DEVELOPMENT

There recently was a flame war on the **linux-kernel mailing list** that got some attention outside the kernel community. It started when **Greg Kroah-Hartman** complained that people were sending patches for the stable series that didn't belong there, and **Linus Torvalds** suggested he learn how to yell at people more; then **Sarah Sharp** said that Linus shouldn't be advocating abuse.

It turned out to be an interesting culture clash. Sarah posed her argument as, "Violence, whether it be physical intimidation, verbal threats or verbal abuse is not acceptable. Keep it professional on the mailing lists." And, "In karate, or any other sport, if your opponent is motionless on the floor, you stop. You can't see the person you're emailing. You can't see if the first conversation-disabling blow has completely knocked them out. You can't see if you've misjudged their mental strength for the day and completely wiped out their ability to use their brain to correct the technical mistake you're trying to get them to fix."

Linus posed his response as being,

"People are different. I'm not polite, and I get upset easily but generally don't hold a grudge—I have these explosive emails. And that works well for some people. And it probably doesn't work well with you. And you know what? That's fine. Not everybody has to get along or work well with each other. But the fact that it doesn't work with you doesn't make it 'wrong'." And he said that the issue was really about "how to work together DESPITE people being different. Not about trying to make everybody please each other." He gave the example of Sarah's work with Greg, saying that they worked a lot with each other, probably because they did work well together, and that that was as it should be.

The debate went on for a bit, and ultimately the two of them, and others, decided to continue the discussion at the upcoming **Kernel Summit**, over cookies and pot brownies.

The debate interests me personally because, on the one hand, no one should ever be abused. That's what abuse is—something that shouldn't happen. Otherwise, it's just an activity.

And on the other hand (in my opinion), not all of his “yelling” is really abuse. For one thing, it’s an e-mail list, and there is a long and glorious history of flame wars on e-mail lists. People aren’t required to participate, even when

Linus Torvalds is the one doing the flaming. So I’m interested to see what kind of public pressure ultimately will come to bear on Linus to stop doing something that’s really an ordinary on-line activity.—**ZACK BROWN**

Non-Linux FOSS: Launchy!



(Image from <http://www.launchy.net>)

With Unity’s method for launching and finding programs and applications, and OS X’s spotlight tool becoming the new way to launch programs, the entire way we think about launching programs is changing. Although I still like to have a few icon shortcuts on my task bar, many folks prefer a quick keystroke to bring up Gnome-Do, or Unity’s launcher, or even OS X’s spotlight. If you’re one of those people, but can’t seem to find a smooth way to accomplish your launching on Windows, check out Launchy.

Launchy is an open-source daemon

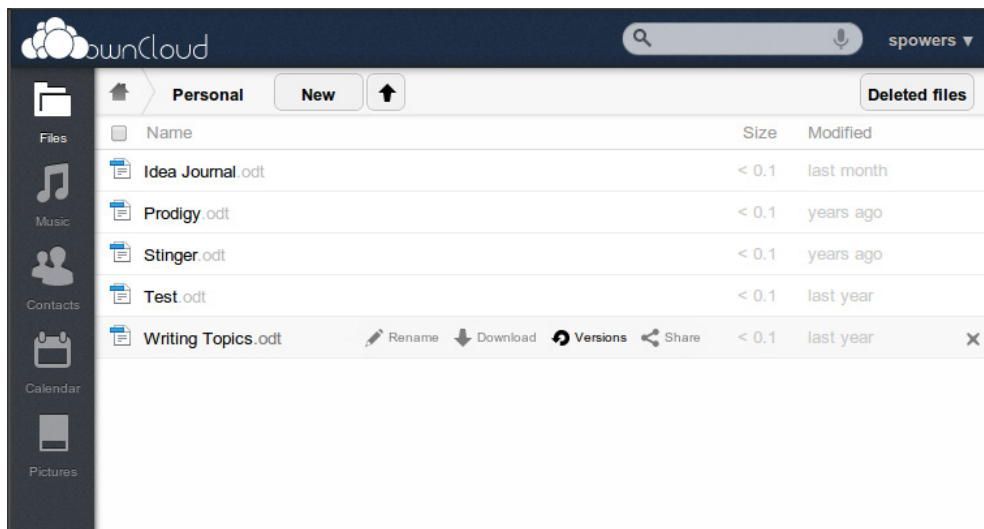
that runs on Windows, OS X or Linux. It listens for a predefined keystroke and then pops up a search window. You start typing the application, document or media file you want to

start, and Launchy autocompletes based on its index of your system. The main advantage is the ability to launch a program without moving your fingers from the keyboard.

Launchy is fast, accurate and doesn’t steal lots of resources on your system. I can’t find a way to make Windows behave this way on its own, so although Launchy is cross-platform, it probably will be the most useful for Windows users. You can download your copy today at <http://www.launchy.net>.

—**SHAWN POWERS**

Own Your Data with OwnCloud



I love Dropbox. I really do. With a Google AdWords campaign, and \$50 or so, I was able to max out my free storage. That means I have around 24GB of free Dropbox storage to fiddle with. Granted, that's a lot, but in the grand scheme of things, 24GB isn't very much space. During the past few years, I've mentioned several alternatives (like SparkleShare), but the new kid on the block, OwnCloud, is a Web-based application that provides a plethora of cloud-based services. The most popular is its file syncing.

Setting up OwnCloud isn't for the

faint of heart, as it requires some PHP tweaking and really should be SSL-encrypted, but for anyone comfortable with configuring LAMP applications, it's not insurmountable.

Once your server is installed, there are native syncing applications for Windows, OS X, Linux, Android and iOS. Because OwnCloud is hosted on your own server, your space limitation is based on your actual hard drive space!

If you've ever wished your Dropbox data was hosted on your own servers, or if you just don't have enough space, check out OwnCloud. It not only supplies file syncing, but with its extendible infrastructure, it also can do calendaring, sharing and pretty much anything else you'd want to do with cloud computing. Check it out today at <http://www.owncloud.org>.

—**SHAWN POWERS**

Surf Safely with sshuttle



In past issues, I've explained how to set up a SOCKS proxy with SSH. I've demonstrated how to tunnel traffic with SSH. I've even shown how to circumvent a company firewall with SSH. I've never been able to use SSH completely as a VPN, however, and that's always bummed me out—until I discovered sshuttle.

Mind you, sshuttle isn't a new program. It isn't even a new concept. What it is, however, is pure awesome. Basically, launching the sshuttle binary with root privileges will modify

your system firewall to tunnel all (yes *all*) traffic through a remote SSH connection. The remote connection doesn't even need administrator privileges, so your shell account at your Web host might suffice for securing your traffic in a hotel or coffee shop. sshuttle will even tunnel your DNS lookups, which

means your entire network interaction should be secure and encrypted.

sshuttle is in many OS repositories, or you can download it from <https://github.com/apenwarr/sshuttle>.

With a simple `sudo sshuttle --dns -vvr username@server 0/0`, all your traffic will be encrypted and funneled through the remote server. Because DNS also is tunneled, it means you won't be vulnerable to DNS poisoning either! Check out sshuttle today. You won't be sorry.

—**SHAWN POWERS**

Mapping Your GIS Data

I've already looked at some GIS applications available on Linux. Programs like GRASS and qgis provide a full set of tools to do GIS. Sometimes, that's really overkill though. You may just want to display some data geographically and create a map. For those cases, there is Thuban, an interactive geographic data viewer (<http://thuban.intevation.org>).

Most distributions should have a package available within their

package management systems. If not, you always can download the sources and build it from scratch. It does depend on Python, among several other libraries, so you need to do a bit of a dependency dance. Binary downloads even are available for Windows and Mac OS X, so you can point your non-Linux friends to them.

If you don't already have data of your own, sources of public-domain GIS data are available

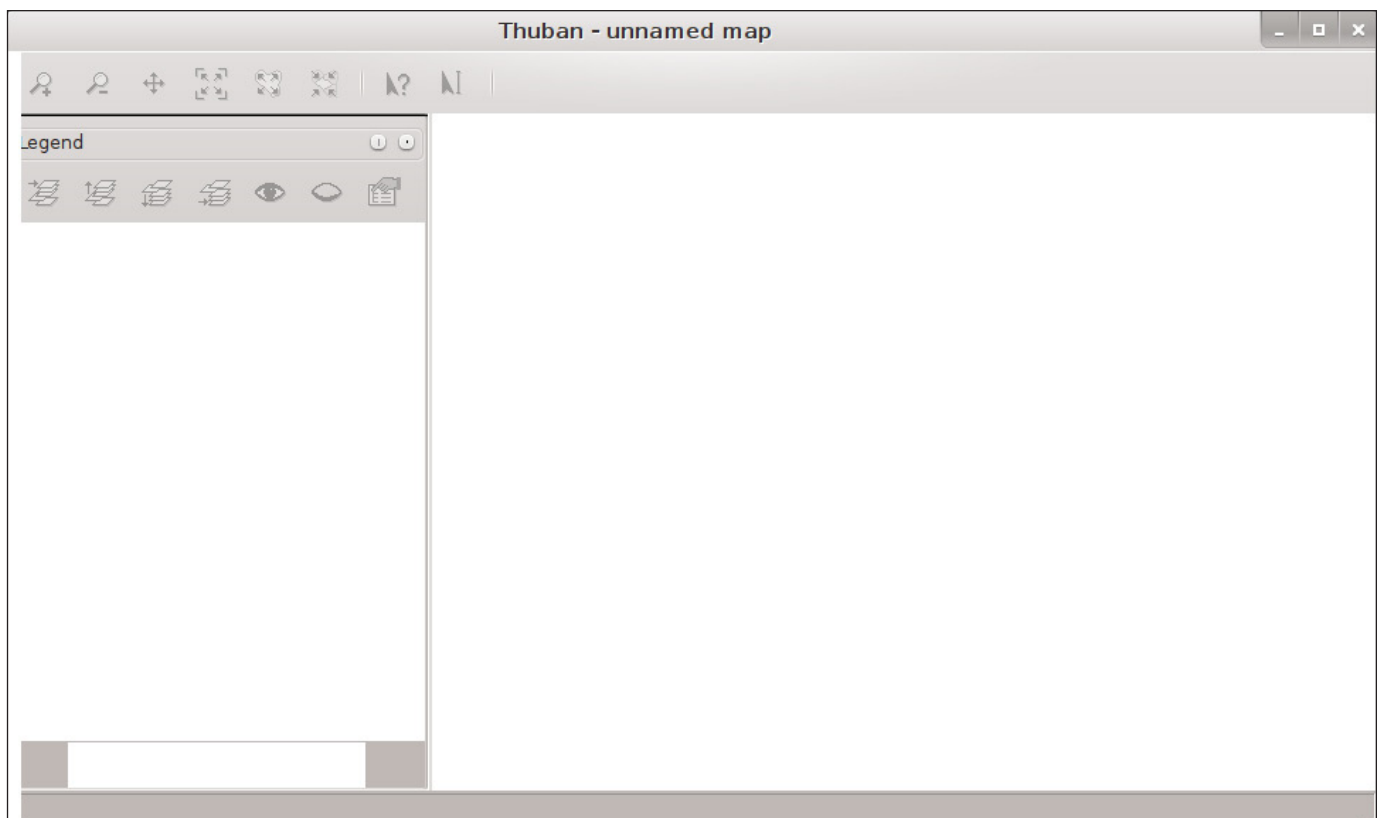


Figure 1. Starting Thuban gives you a blank slate.

on-line. Here are a couple:
<http://www.naturearthdata.com/features> and <http://wiki.openstreetmap.org/wiki/Shapefiles>.
 The files available on these sites will get you started with SHP files that contain at least basic features for most of the world.

Thuban is not as flexible as full-fledged GIS software and cannot handle very many data file formats. You can use SHP files, DBF database files and various image file formats. In the screenshots for this article, I

simply grabbed several of the data files available on-line.

When you start Thuban, you end up with a completely blank slate (Figure 1). The first step is to start a new session, which you can do by selecting the menu item File→New Session (not much will change on the screen). In order to start building your map, you need to add layers that can be manipulated. I started by selecting the menu item Map→Add Layer and adding in an SHP file to give me the basic geographic attributes for my

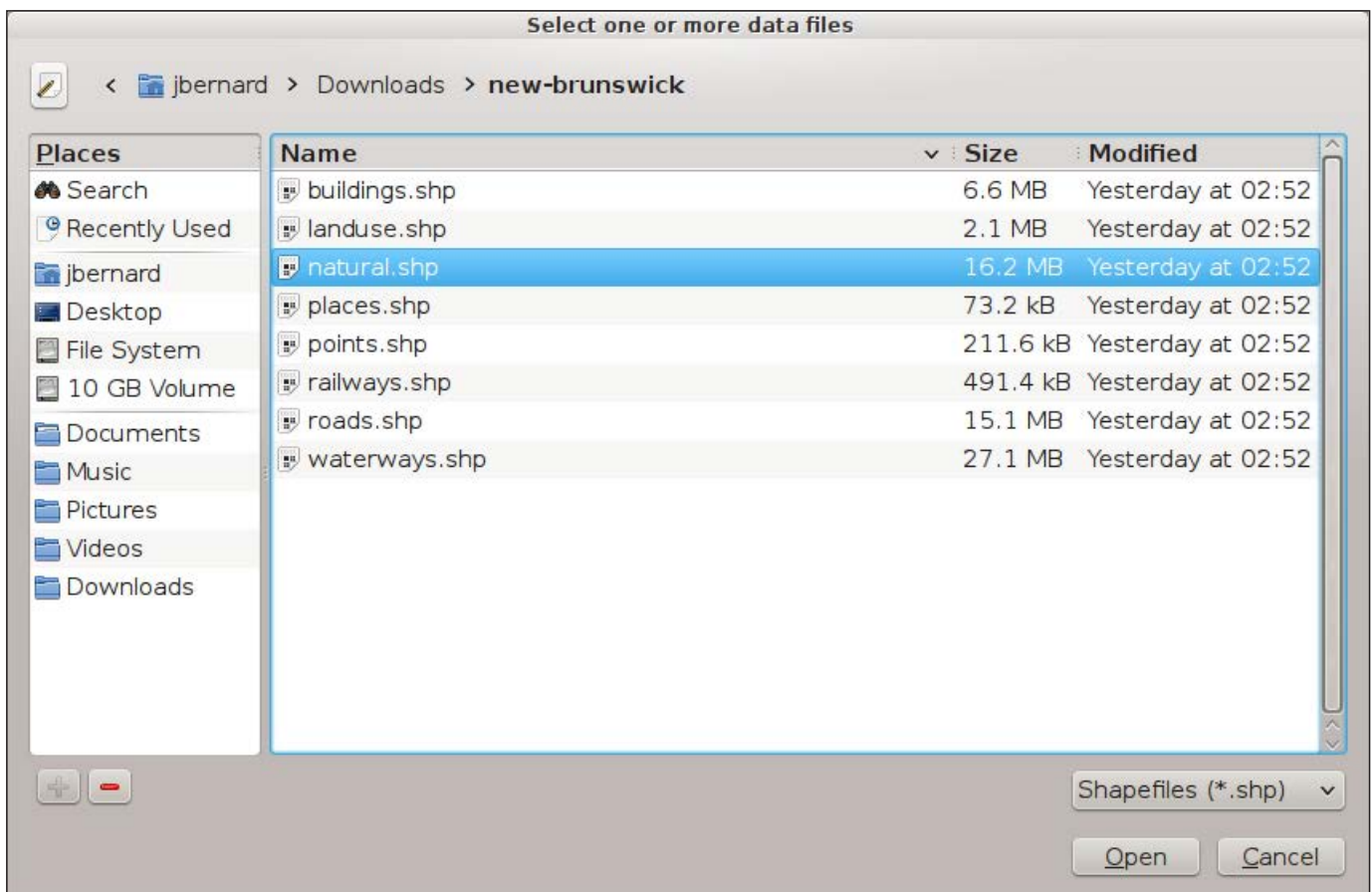


Figure 2. Adding a new layer opens a file selection dialog where you can choose an SHP file.

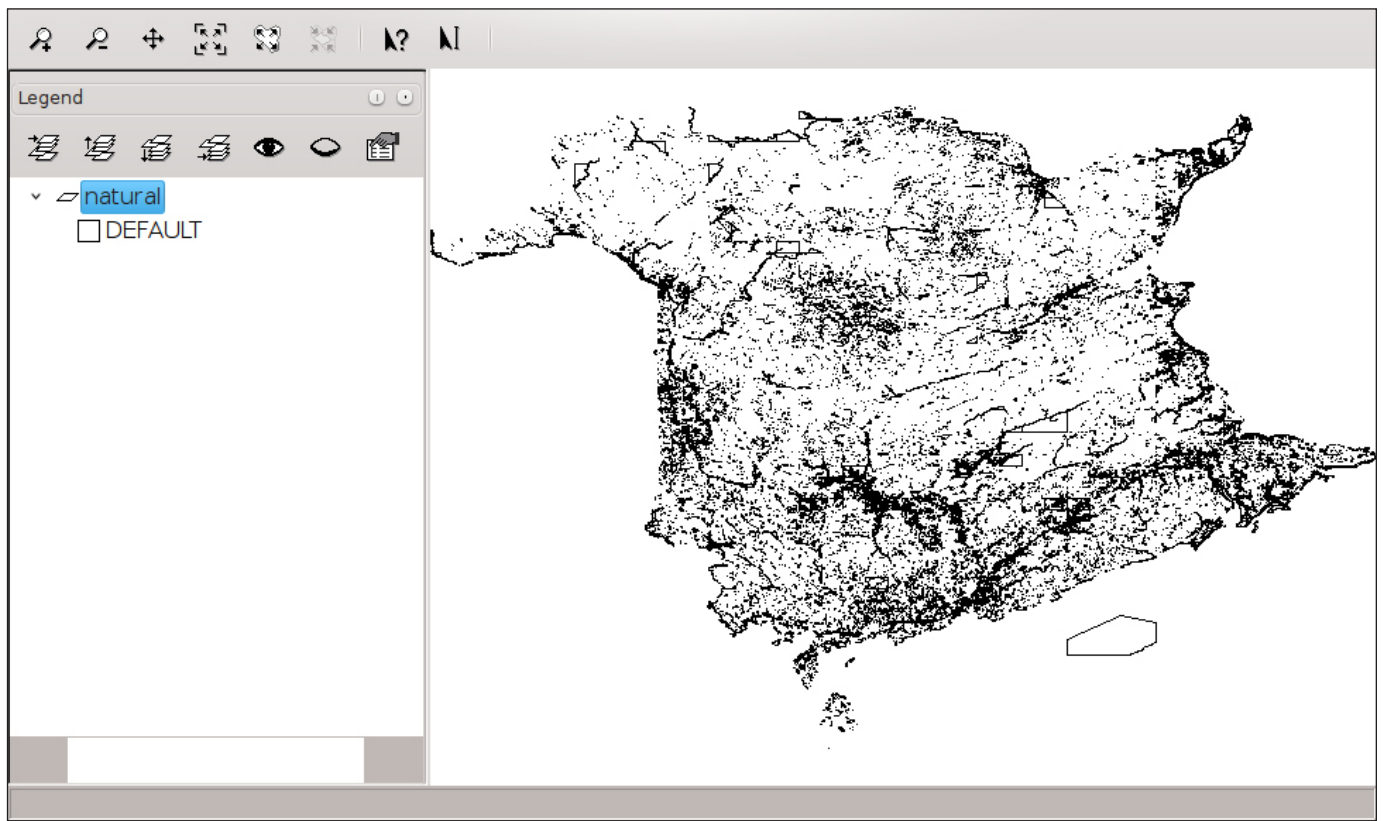


Figure 3. By default, Thuban just displays all of the data with a single symbol color.

home province of New Brunswick (Figure 2). This includes several different geographical items, such as water, river banks and parks. The default display is not very interesting yet (Figure 3).

You can edit the way a layer is displayed either by double-clicking the layer within the list in the legend pane or by right-clicking the layer of interest and selecting Properties. This will pop up a new window (Figure 4). In this case, I selected the “type” field within the classification pane. The easiest choice at this point is to click

the Generate Class button. The Generate Classification window will pop up, where you can click on the Retrieve From Table button to get a list of the possible values. I accepted the default gray-scale mapping for the colors, giving four new entries in the layer properties. But this is not very interesting either, yet. Selecting each of the new properties, you can edit the symbol and change the colors for each of the types (Figure 5). If you want to have a preview of what this will look like, you can click the Try button. If it doesn’t quite

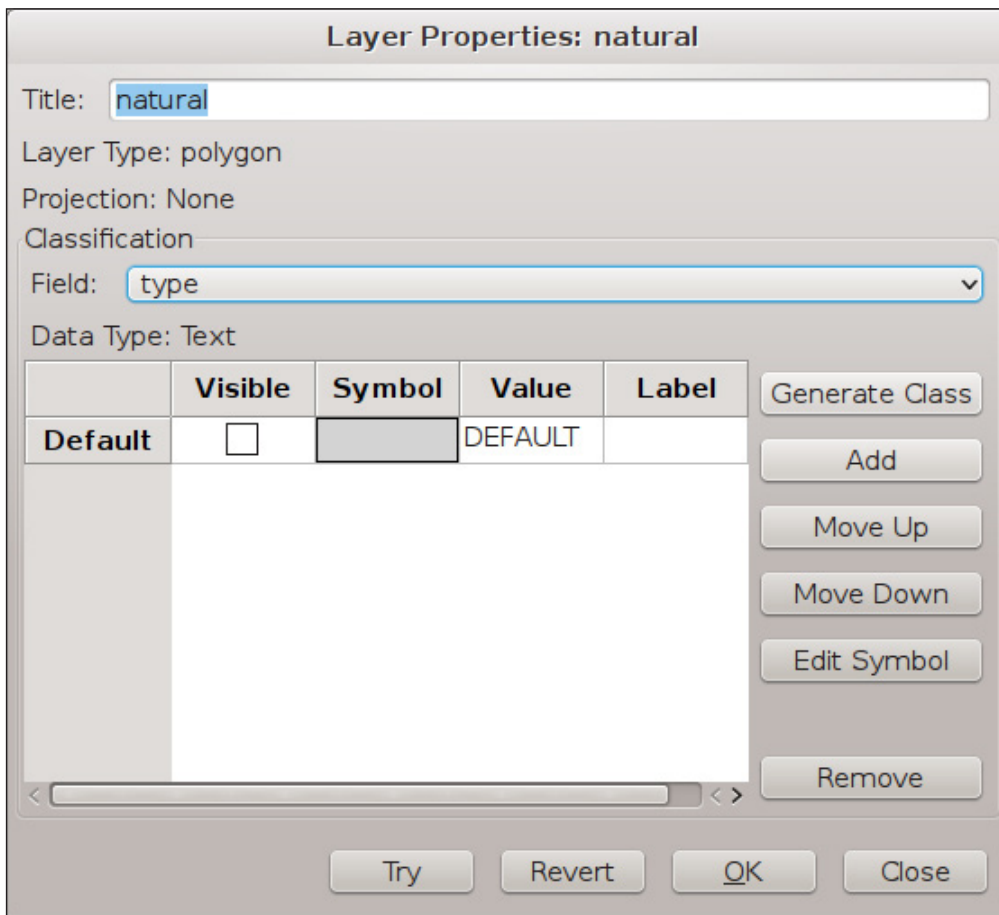


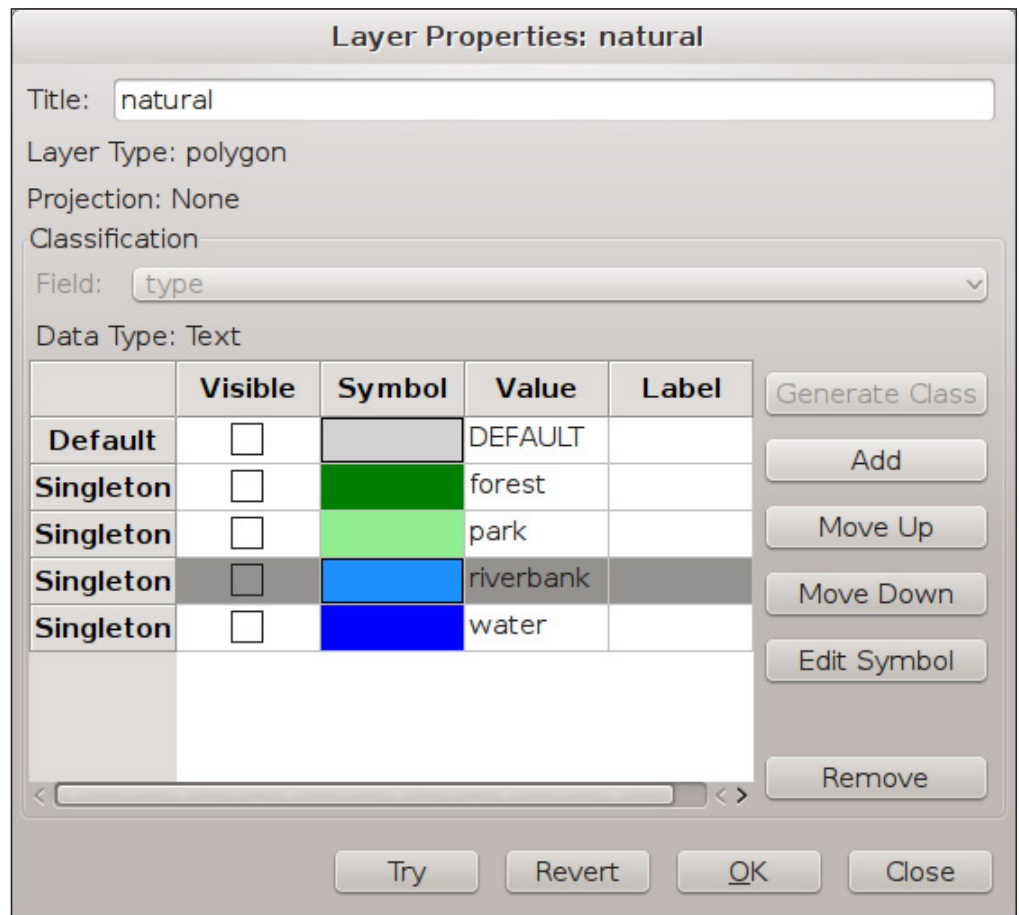
Figure 4. Each layer has a properties window where you can control how the data gets displayed.

look right, you always can click the Revert button to undo the changes and try something else.

Although every map begins with a single layer, it is very rare that a single layer is enough to show all the details you may want to have displayed. In this example, I don't have any roads on my map. A separate SHP file is available that has this information, however. So, I clicked on the menu item Map→Add Layer and added the file roads.shp. Opening up the properties dialog shows that this particular SHP file

has several different attributes to play with. For now, I selected four different road types and highlighted them with four different colors. There is still a default color for any road types other than the four I selected. To make them go away on the map, you can select the default property and simply make it transparent. Then, only the four selected road types will show up. Now the map is starting to look a bit more interesting, and I need to start worrying about what order the layers are in.

Figure 5. Using the Generate Class button is a shortcut to get you started.



Thuban will draw layers in the order they appear in the legend list, starting at the bottom and working its way up. You can move a particular layer up or down by selecting it and then using the buttons at the top of the legend pane.

Another type of layer you can use is an image layer. Obviously, the image needs to be geo-referenced in some way. Thuban supports the geoTIFF file format. If you place your image at the bottom of the layer list, you then can draw on top of it with the data in the SHP files.

To manipulate the map itself, Thuban uses a sort of mode system. To zoom in, you need to select the zoom button. Then, you either can use click and drag to select a region to zoom in on or simply click a spot on the map to re-center and zoom in. Once you have zoomed in, you can use the pan tool to move the view window around the map to highlight different regions. There are buttons to zoom you to specific scales such that the entire map is visible. This always takes you back to the default map view.

Two tools allow you to work with individual elements from an SHP file. The first is an information tool that pops up a detail window for any element you select. The second is a label tool. When you select an element, a dialog window pops up allowing you to select one of the properties to be displayed as a label.

Once you have a map you're happy with, you probably will want to save it for later use. Because Thuban works with sessions, all of your work in generating the map will be saved as a session within Thuban, as long as you remember to save it by clicking the menu item File→Save Session.

But, this doesn't help much if you want to use your map outside Thuban. There is an option to export a map as an SVG file by using the menu item Extensions→Write SVG Map. This is not the most efficient output available, however. My simple example here blew up to more than 50MB for a single map with two layers.

The other option is to print your map. Although you can print to actual paper, for a hard copy, you also can print to a file using the generic PostScript printer. This generates a PostScript file that will be a bit more manageable. You also can convert this PostScript file to other formats with relative ease. So, to get a PDF of your map, you can print to a PostScript file and then convert it to PDF with the ps2pdf utility. Now you have a map that you can share with friends and family.—**JOEY BERNARD**

They Said It

My home is not a place, it is people.
—*Lois McMaster Bujold*

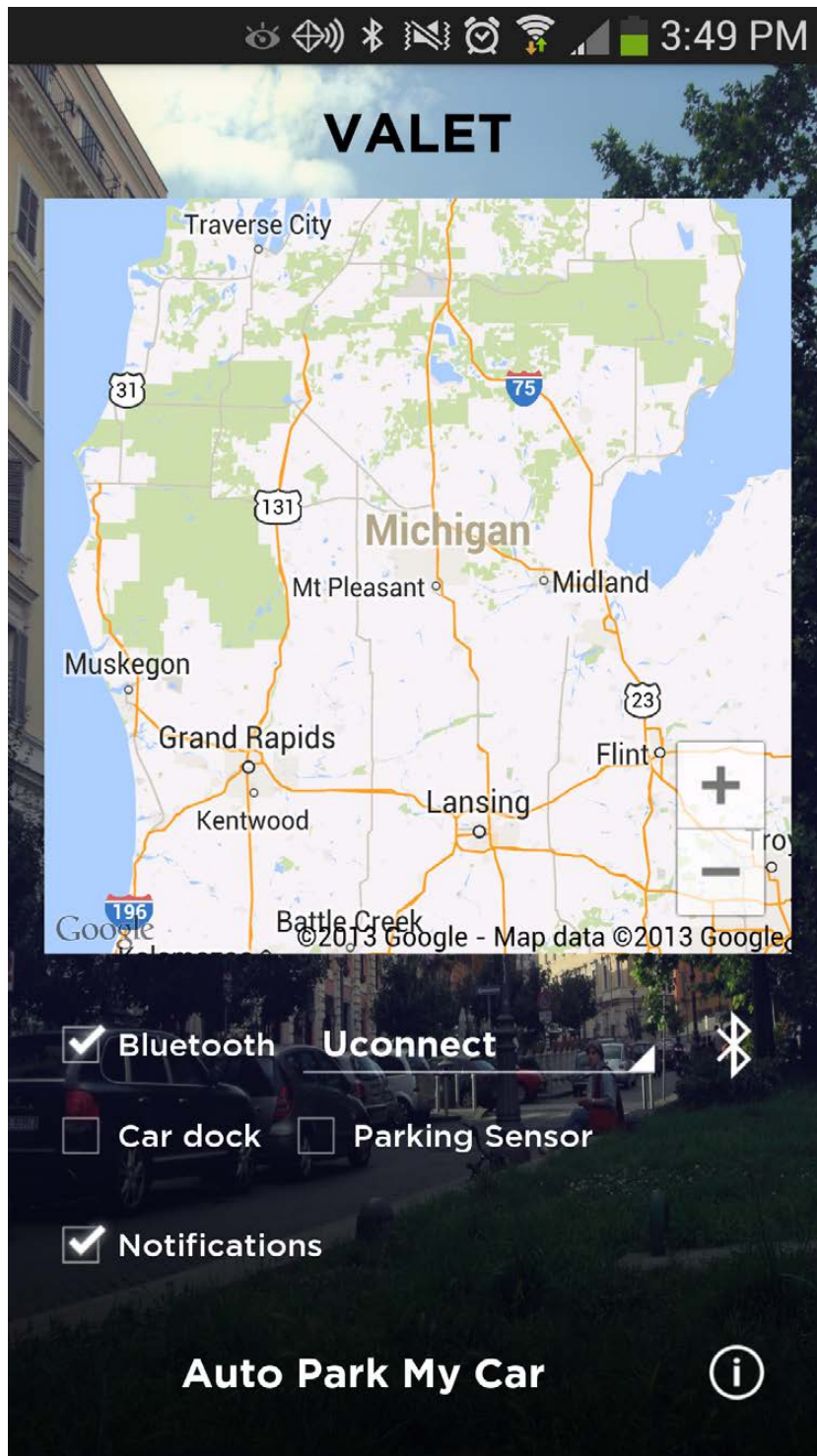
The middle of the road is where the white line is—and that's the worst place to drive.
—*Robert Frost*

The future will be better tomorrow.
—*Dan Quayle*

Some have been thought brave because they were afraid to run away.
—*Thomas Fuller*

Reading is no substitute for action.
—*Colleen Wainwright*

Dude, Where's My Car?



When my family moved to Grand Rapids, Michigan, last year, one of the biggest adjustments was dealing with city parking. While we usually remember what side of the mall we parked on, there was a time downtown that I couldn't remember what parking garage we used, much less what level or spot. Thankfully, I had the parking ticket in my pocket, which included the address for the particular parking lot we used. Although we had to walk up 15 levels one by one to find the car, at least we knew it was there somewhere!

If you've ever walked around a parking lot wondering if you were the victim of theft, or possibly going senile, Valet is the perfect app for you. It not only

remembers the GPS location of your parked car, but it also has a timer to remind you of parking meter timing. Plus, if your car has Bluetooth connectivity, Valet will record where you left your car without any interaction on your part. It just marks the location where Bluetooth disconnected, and it happily guides you back when you've finished spending your paycheck at Teavana in the mall.

(Maybe that's just me.)

In fact, Valet fills such a simple yet helpful purpose, it's earned this month's Editors' Choice award. Its automatic tracking based on vehicle Bluetooth is really the feature that puts it over the top for me. It's the best \$0.99 I've spent in a while. You can find it at the Google Play store, or check out the Web site: <http://valetapp.co>.

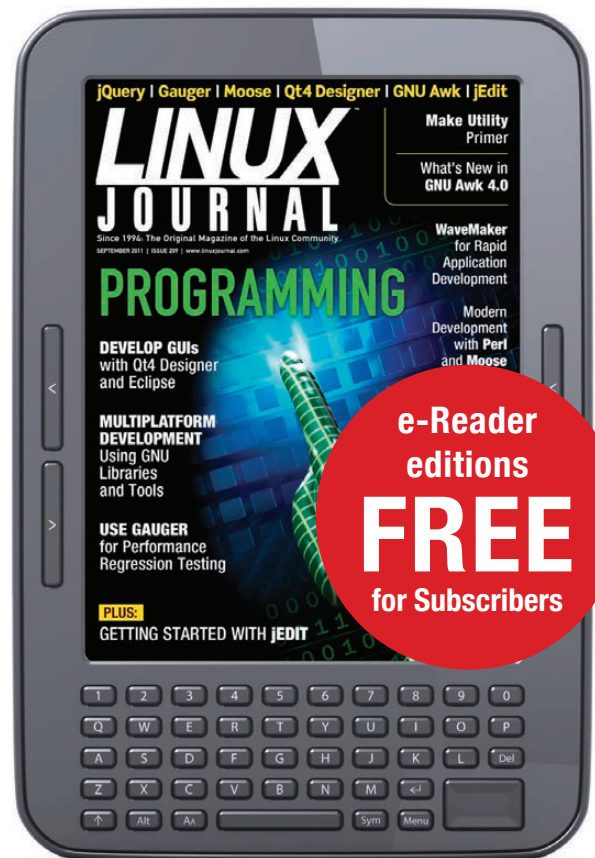
—**SHAWN POWERS**

LINUX JOURNAL

on your
e-Reader

Customized
Kindle and Nook
editions
now available

LEARN MORE





REUVEN M.
LERNER

Zurb Foundation

Designing a responsive Web site? Zurb Foundation is a worthy competitor to Twitter Bootstrap.

I was recently speaking with a new client about a consulting project, and we were discussing the technologies we were going to use. I said, “So, I assume that we’ll be using Twitter Bootstrap as a CSS framework, right?” Bootstrap, of course, is the open-source CSS framework written in the LESS language that recently released its third version, which has become wildly popular among Web developers. I’ve used Bootstrap on a number of projects in the past few years, and I’ve been pleasantly surprised to discover just how easily I could implement layouts, menus and sidebars, as well as produce nice-looking tables and buttons.

I was surprised when the client said, “No, we’re actually using Zurb Foundation.” Now, I had heard of Foundation before. Zurb is a Web design company based in California, whose Foundation framework preceded and inspired Bootstrap

in many ways. But it really hadn’t registered much on my radar, and I hadn’t paid much attention to it. It turns out that I should have been paying attention. Zurb is a worthy competitor to Bootstrap, and it has some advantages that make it more appropriate in some cases.

So in this article, I take a look at Zurb Foundation, considering it both as a standalone CSS framework and as something you can use from within an application framework, such as Ruby on Rails. Zurb recently released version 4 of the Foundation framework, and although it is not as popular as Bootstrap, the people who are using it seem to be quite passionate about its advantages.

Foundation Basics

If you aren’t yet familiar with CSS frameworks, the idea is pretty straightforward. When you want to lay out a Web application, you likely

will want some headlines, some text, some sidebars, a menu and all sorts of other design elements. You could design the entirety of the layout, classes and CSS from scratch each time. Or, you could use predefined styles, designed to work with all sites, with a great deal of flexibility.

Each modern CSS framework functions in this same way. You put text into divs and give each div a class indicating how many columns across it should be, up to the maximum number of columns defined by the framework. Foundation offers 12 columns, meaning that you can lay out your design in one very wide column, two equally sized columns (6+6) or even something wilder, such as 3+4+5.

Foundation, like other modern frameworks, goes way beyond just providing you with a nice grid. It also gives you control over navigation elements and forms, and even gives you some JavaScript-based elements for displaying images and dialog boxes.

The bulk of Foundation is implemented using SCSS, a modern implementation of SASS. SCSS is a superset of CSS that can be compiled (using an open-source tool) into regular CSS files. However, you don't need to know SCSS in order to

use Foundation.

How can you use Zurb Foundation? Download the package. You have several options, but the easiest one is to use the precompiled CSS files. Go to the Zurb Foundation download site at <http://foundation.zurb.com/download.php>, and from there, click on the button for "download Foundation CSS".

If you want to change the defaults, you could download the SCSS version and do it that way. Alternatively, the good people at Zurb have provided a Web-based selection system, such that you can indicate what parts of Foundation you want, as well as which colors and styles. The CSS that you download then will be precompiled, customized according to your needs.

If you download the simple CSS version and open the resulting zipfile, a sample HTML page (index.html) will be at the top level of the folders. There also will be several subfolders, whose names are fairly self-explanatory: js (JavaScript), css (stylesheets) and img (images).

Looking at the HTML page, you can see that Foundation is loaded in five steps:

1) First, you load the core Foundation CSS file:

```
<link rel="stylesheet" href="css/foundation.css">
```

Now, this is definitely a long and complex CSS file. However, remember that this defines all of the styles, for all of the aspects of Foundation. Moreover, this is the result of translating the SCSS source into CSS; expressed as SCSS, the file is much shorter and easier to understand.

2) After loading the Foundation CSS file, you then load Modernizr.

I've covered Modernizr in this column before; it allows you to test for certain HTML5 features and use alternatives if the feature doesn't exist. You can use Modernizr in your application if you wish, but it's loaded here so that Foundation can handle different browser versions.

3) Next, you load the JavaScript for Foundation. But here, things

Listing 1. Hello, world

```
<!DOCTYPE html>
<html>
<head>
  <title>Minimal Zurb Foundation file</title>
  <link rel="stylesheet" href="css/foundation.css" />
  <script src="js/vendor/custom.modernizr.js"></script>
</head>
<body>
  <h1>Hello, world headline</h1>
  <p>Hello, world paragraph</p>
  <script>
    document.write('<script src=' +
      ('__proto__' in {} ? 'js/vendor/zepto' :
      'js/vendor/jquery') + '.js'></script>')
  </script>

  <script src="js/foundation.min.js"></script>
  <script>
    $(document).foundation();
  </script>
</body>
</html>
```

get a bit more interesting. Rather than a `<script>` tag that loads a JavaScript library, Foundation allows you to use either jQuery or Zepto, a lightweight library that copies many of jQuery's most popular features. Foundation tests, at runtime, whether the `__proto__` property is defined on an empty object (`{}`). If so, it loads Zepto. If not, it loads jQuery. This is where copying the code that comes with Foundation is probably the best choice; just make sure that when you deploy your application, you have both Zepto and jQuery available.

4) Load the Foundation JavaScript file itself, which gives you the capabilities you wanted.

5) Activate Foundation, using the formula by invoking the "foundation" function on the document object. This syntax works with both jQuery and Zepto:

```
$(document).foundation();
```

Note that although steps 1 and 2 take place in the `<head>` section of the HTML file, the remaining steps (which involve JavaScript) are placed at the bottom of the page, just before the closing `</body>` tag. This is a common technique for speeding up JavaScript execution, and it ensures that the browser parses and displays

the page before loading and executing external JavaScript files.

A minimal file that uses Zurb Foundation (with a "Hello world" headline and paragraph in the body) is shown in Listing 1.

Layouts with Foundation

If you look at the file in Listing 1 in your browser, you might well be disappointed by how it looks. The text is too close to the edge of the page, and it doesn't have any of the pizzazz that you would have expected from a framework. Well, there's a simple reason for this. It hasn't used any of the styles that Foundation provides.

So, let's begin by putting the text inside a `<div>` with a class of "row". That shouldn't come as a surprise to anyone who has used Bootstrap or other CSS frameworks; they often work this way, and also use a "row" class to indicate that you're now enclosing one horizontal band of text. The `<body>` (without the Foundation-specific JavaScript), thus, will look like this:

```
<div class="row">
  <h1>Hello, world headline</h1>
  <p>Hello, world paragraph</p>
</div>
```

If you reload your browser, you'll see that the text has moved closer to the

center of the page. That's because using the "row" class has defined margins, as well as allowed your content to be more centered and more responsive. While you're at it, make your window smaller. At a certain point, the text will make itself smaller, changing (on my computer, at least) from 2.75em to 2.15em. The idea, of course, is that people with smaller browser windows still should be able to see as much text as possible, and that the text should adjust itself proportionally.

Of course, your sites generally will include more text than just "Hello, world." I'll add some dummy text inside the paragraph, so that it takes up a few lines. But this time, I don't want the text to extend all the way to the right side. Rather, I want it to take up only half the screen.

Foundation makes that easy to do. Create a new `<div>` tag, and give it two classes. First, give it one that represents the number of columns (out of 12) that you want your text to use. So to extend all the way across, you would say `large-12`. To extend halfway across, you would say `large-6`, and to extend one quarter of the way across, you would use `large-3`. (If columns within a row don't add up to 12, you can get some funny-looking results.) Second, you add the "columns" class, allowing Foundation

to display your site using appropriate columns. Given these, the text looks like this:

```
<div class="row">
  <div class="large-6 columns">
    <h1>Hello, world headline</h1>
    <p>Hello, world paragraph. And even more hello, world. And
    even even more more hello, world. And so on, and so forth.
    And again. And even more hello, world. And even even more
    hello, world. And so on, and so forth.</p>
  </div>
</div>
```

If you want two side-by-side columns, you easily can do that:

```
<div class="row">
  <div class="large-6 columns">
    <h1>Hello, world headline</h1>
    <p>Hello, world paragraph. And even more hello, world.
    And even even more more hello, world. And so on, and
    so forth. And again. And even more hello, world. And
    even even more more hello, world. And so on,
    and so forth.</p>
  </div>
  <div class="large-6 columns">
    <h1>Hello, world headline</h1>
    <p>Hello, world paragraph. And even more hello, world.
    And even even more more hello, world. And so on, and
    so forth. And again. And even more hello, world. And
    even even more more hello, world. And so on,
    and so forth.</p>
  </div>
</div>
```

And, if you want both paragraphs of text under the same headline, you can do that as well:

```
<div class="row">
  <h1>Hello, world from a long headline</h1>
  <div class="large-6 columns">
    <p>Hello, world paragraph. And even more hello, world.
    And even even more more hello, world. And so on, and
    so forth. And again. And even more hello, world. And
    even even more more hello, world. And so on, and
    so forth.</p>
  </div>
  <div class="large-6 columns">
    <p>Hello, world paragraph. And even more hello, world.
```

```
And even even more more hello, world. And so on, and
so forth. And again. And even more hello, world. And
even even more more hello, world. And so on, and
so forth.</p>
</div>
</div>
```

Now, could you have accomplished this on your own? Of course. But everyone who ever has worked with CSS knows it can be difficult to get this right. Moreover, what happens when someone wants to browse this site with a smartphone or a small browser window? Zurb Foundation is

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 \$1375
- 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
- 12.5" 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 \$1920
- W530, T430, T530, X1 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
- 2.4-2.8 GHz Core i5
- Up to 16 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



responsive, meaning that it adjusts automatically. This means that although you might have defined your paragraphs to be side by side, Foundation is smart enough to stack them vertically if the browser window is too narrow. You can see this for yourself by changing the width of your browser window with this page.

It gets even better. Perhaps you want narrow browser windows, such as those used on cell phones, to continue displaying the columns side by side. This is where Foundation's "dual grid" layout system shines. Just as you can use `large-*` classes for large browser windows, you can use `small-*` classes for small windows. Foundation automatically will apply the appropriate class, making it responsive in multiple ways. This means that each div will look like this:

```
<div class="large-6 small-3 columns">
```

Foundation also provides a number of button styles. You can use these in forms or by themselves for actions that you'll then hook up with JavaScript callbacks. A simple button can be given the "button" class:

```
<input class="button" value="Press me" />
```

You can use classes to change the button's size (tiny, small, medium or large), color (secondary, alert or success) or even rounded corners (with a "round" class). The documentation indicated that "radius" should work as well as round, but it didn't operate correctly within my Chrome browser window. You can combine these classes, such as in:

```
<input class="button success large round" value="Press me" />
```

Buttons in Foundation handle hovering nicely as well, such that moving the mouse cursor over a button dims it, and clicking on it also gives visual feedback. Again, these are things you could do on your own, but why bother, when Foundation already has defined it?

Navigation

Most sites on which I work need to have a top navigation bar. Foundation handles this easily as well. For example, let's say you want to have a title on the top navigation bar, and then links to pages 2, 3 and 4 of your site.

Doing this is a bit trickier than what you've seen so far, in that it requires some additional elements. Because I'm using HTML5 for this

example, let's take advantage of two special elements defined there, `<nav>` and `<section>`, which Foundation expects.

The top-level navigation bar is defined in a `<nav>` element with a class of "nav-bar". Within that, you start a `` element, normally used for bulleted lists, with a class of "title-area". Now the magic begins. If you create an `` element with classes of "name" and "active", the navbar now will have a title. It's typical to put that name in an `h1` tag, and then to put the enclosed text inside a link, so that people can navigate to your organization's home page:

```
<nav class="top-bar">
  <ul class="title-area">
    <li class="name active"><h1><a href="#">Test site</a></h1></li>
  </ul>
</nav>
```

You then can have one or more menu items, with the items flush left (with the title) or right (with the right side of the browser). Here, I've chosen to align it to the right. My menu items all go within a `<section>` tag, with the class of "top-bar-section", and then an inner ``, with a class of "right" (indicating the alignment I want) and containing the menu items on the right side. The result looks like this:

```
<nav class="top-bar">
  <ul class="title-area">
    <li class="name"><h1><a href="#">Test site</a></h1></li>
    <section class="top-bar-section">
      <ul class="right">
        <li class="active"><a href="2.html">Page2</a></li>
        <li class="active"><a href="3.html">Page3</a></li>
        <li class="active"><a href="4.html">Page4</a></li>
      </ul>
    </section>
  </ul>
</nav>
```

You can get even fancier if you want. Put the entire `<nav>` element inside a `<div>` whose class is "fixed", and the navigation bar will stay at the top of the screen.

Finally, your navigation bar may include drop-down menus. If, for example, you want the "page4" link to be a menu, add the "has-dropdown" class to the `` tag. Then, within the ``, add a new `` with a class of "dropdown". The inner `` elements then will be treated as menu items. For example:

```
<li class="has-dropdown"><a href="4.html">Page4</a>
  <ul class="dropdown">
    <li><a href="4.html">Page4a</a></li>
    <li><a href="4.html">Page4b</a></li>
    <li><a href="4.html">Page4c</a></li>
  </ul>
</li>
```




Dedicated Server

You are in Control!



Dedicated Server KS 1

5 TB traffic
24/7 North American Support

CPU: **Intel i3 (2 Cores / 4 Threads)**
Frequency: **3.4GHz+**
RAM: **8 GB DDR3**
Hard disk: **2x 1TB SATA2**

\$39

 /month

Find the Entire Range:
www.ovh.com/dedicated-servers



Dedicated Servers



Dedicated Cloud

For more details:



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



DAVE TAYLOR

Image Manipulation with ImageMagick

Think shell scripts can deal only with text? You'll be amazed at what you can do as Dave begins his exploration of ImageMagick and its many useful tricks.

I've spent a lot of time in this column talking about text processing and analysis, with the basic assumption that if you're using the command line, you're focused on text. That's not always true, and if you work with images at all—whether JPEG, PNG, GIF or another format—there's a free-to-download suite of image-related utilities available that offers rather amazing capabilities direct from the command line and, therefore, also from within shell scripts.

I'm talking about ImageMagick, a set of programs that has grown and expanded through the years and now includes powerful Perl and Ruby interfaces too. But, pshaw! We don't need no stinkin' Perl or Ruby. We'll stick with our

hard-core shell commands, thank you very much.

You'll find a downloadable binary and source both at <http://www.imagemagick.org>, and as always, I recommend you download source and compile it on your system if you can. It's far more reliable than hoping someone else's compiled version is optimized for your own hardware configuration.

A variety of different commands are included with the ImageMagick distribution that I divide into "analysis" and "editing" tools. For this article, let's stick with the analysis tools. Let me start by showing you how much more information it offers on a typical image file than the standard Linux command line.

Analyzing Images for Non-Optimized Resolutions

If you've been using Linux for even a short time, you've probably learned about the `file` command. It can be helpful with some file types:

```
$ file wp-content.tar.gz
wp-content.tar.gz: gzip compressed data, from Unix
```

But, the command is generally useless with images:

```
$ file pvp.jpg
pvp.jpg: JPEG image data, EXIF standard
```

Um, what about image size? How about any useful info at all? Jeez.

Enter the ImageMagick `identify` command:

```
$ identify pvp.jpg
pvp.jpg JPEG 970x311 DirectClass 114kb 0.010u 0:01
```

Ahh...so this particular image has the dimensions (the suite refers to dimensions as the "geometry" of the image) of 970x311. That's useful.

Do you want even more information though? The `-verbose` option spits out a somewhat overwhelming amount of data:

```
$ identify -verbose pvp.jpg
Image: pvp.jpg
```

```
Format: JPEG (Joint Photographic Experts Group JFIF format)
Geometry: 970x311
Class: DirectClass
Colorspace: RGB
Type: TrueColor
Depth: 8 bits
Endianness: Undefined
Channel depth:
  Red: 8-bits
  Green: 8-bits
  Blue: 8-bits
Channel statistics:
  Red:
    Min: 0
    Max: 255
    Mean: 180.72
    Standard deviation: 74.2122
  Green:
    Min: 0
    Max: 255
    Mean: 168.593
    Standard deviation: 76.0343
  Blue:
    Min: 0
    Max: 255
    Mean: 169.459
    Standard deviation: 77.244
Colors: 21864
Rendering-intent: Undefined
Resolution: 72x72
Units: Undefined
Filesize: 114kb
Interlace: None
Background Color: white
Border Color: #DFDFDF
```

```
Matte Color: grey74
Dispose: Undefined
Iterations: 0
Compression: JPEG
Orientation: Undefined
JPEG-Quality: 94
JPEG-Colorspace: 2
JPEG-Sampling-factors: 1x1,1x1,1x1
signature: bc8a6a698ca35fd8feab71452423386ff98b1fb7b5ec ...
Profile-xmp: 811 bytes
Profile-exif: 22 bytes
  unknown
Profile-app12: 15 bytes
Tainted: False
User Time: 0.020u
Elapsed Time: 0:01
```

Truth be told, dimensions and resolution are the most useful pieces of information from this crazy-long output.

With a tiny bit of effort, you can extract just those items of information:

```
$ identify -verbose pvp.jpg | grep -E '(Resolution:|Geometry:)'
Geometry: 970x311
Resolution: 72x72
```

Now imagine you are working on a Web site and want to ensure that no images on the site are greater than 72dpi, a standard screen resolution. Higher print-ready resolutions are rather pointless, because a 300dpi image will render the same on a screen as its lower-resolution

brethren—it'll just load slower.

Here's one way you can identify images in a directory with incorrect resolutions:

```
#!/bin/sh
identify=/usr/bin/identify
# check images to ensure that they're all 72x72 resolution.
for filename
do
    resolution=$(($identify -verbose $filename | \
        grep -i "Resolution:" | grep -v 72x72)
    if [ ! -z "$resolution" ] ; then
        echo "Warning: Image $filename has $resolution"
    fi
done
exit 0
```

When I run this on a directory of images on my own system, a set of JPEG format files on my <http://www.AskDaveTaylor.com> site, here's what I get:

```
$ checkres.sh *.jpg
Warning: Image auction-seller-img1.jpg has Resolution: 75x75
Warning: Image auction-seller-img2.jpg has Resolution: 75x75
Warning: Image browsing-the-photo-folder.jpg has Resolution: 96x96
Warning: Image brushed-metal.jpg has Resolution: 300x300
...
```

That's a surprise! I didn't realize that I had 300x300 and these other weird resolutions. An easy way to speed up my site, therefore, is to lower the resolution on these images to the

standard 72dpi. This is something that also can be done with a call to a different ImageMagick utility, but let's tackle that in another article.

Working with Image Size

Since I write a lot of scripts that harvest images or other content from sites and repurpose them for my own (generally private, not public-facing) use, I also find it is darn helpful in a shell script to be able to ascertain the size of an image I've just grabbed.

If you've guessed that `identify` is the key, you're right. In fact, given an image, this is an easy way to grab its

height and width:

```
height=$(identify $image | cut -d\  -f3 | cut -dx -f1)
width=$(identify $image | cut -d\  -f3 | cut -dx -f2)
```

There's no need for verbose output, because the geometry of the image is included in the default output.

Now it's easy to produce higher-quality HTML, for example, by including images with their proper dimensions:

```
echo "<img src=$image height=$height width=$width>"
```

What's better is that Web browsers are able to scale images automatically,

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.

so if you specify a height and width that are different from the default dimensions (oops, sorry, “geometry”) of the image, it’ll scale automatically.

This means if I want to include the pvp.jpg image on an automatically generated page, but decide 970 pixels is just too wide, I can simply include it as:

```
<img src=pvp.jpg height=207 width=646>
```

and the browser—be it Chrome, Safari or even MS IE—will scale it appropriately.

Calculating the smaller size is straightforward with `bc`, another underappreciated Linux command. The entire sequence might look like this to scale the image to 66% of its original dimensions:

```
#!/bin/sh
identify=/usr/bin/identify
scale=0.666
image=$1 # add input validation code

height=$(($identify $image | cut -d\  -f3 | cut -dx -f1)
width=$(($identify $image | cut -d\  -f3 | cut -dx -f2)
newwidth="$(echo $width \* $scale | bc | cut -d. -f1)"
newheight="$(echo $height \* $scale | bc | cut -d. -f1)"
echo "<img src=$image height=$newheight width=$newwidth>"
exit 0
```

In practical use:

```
$ scaledown.sh pvp.jpg
```

```
<img src=pvp.jpg height=646 width=207>
```

That’s easy enough!

With some creativity, you can see how even just the `identify` command that’s included with ImageMagick opens up a world of image file scripting possibilities, whether you’re working with Web sites directly or simply seek to analyze directories of images for unusual values or settings.

I’ll dig into some of the really slick editing and modification capabilities, including an easy way to add a so-called watermark to your graphics, along with ways you can automate fixing 300dpi resolution images or even scaling images in an upcoming article.

As a final note, although I explain how you can take a large image and have it show up smaller on a Web page by using different values for height and width, it would be remiss of me not to mention that if you’re going to use only the smaller size, it’s smarter to resize the original image. It makes your page faster to load, less unneeded data is transferred and everything just generally is happier (including the search engines). Now you know. ■

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>.

The Open Source World comes to Raleigh, NC

ALL THINGS OPEN™

presented by

IT-ology™

OCTOBER 23-24, 2013 | RALEIGH CONVENTION CENTER

SCHEDULED TO SPEAK



Scott Chacon

github
SOCIAL CODING



Chris DiBona

Google



Chris Aniszczyk

twitter



Jessica McHellar

python™



Whurley

CHAOTIC
MOON
STUDIO



Andy Hunt

The Pragmatic
Programmer



Angie 'Webchick' Byron

Acquia



Lee Congdon

redhat.

Open Source in the Enterprise



ALLTHINGSOPEN.ORG





KYLE RANKIN

Command-Line Cloud: rss2email

Now that Google Reader is dead and buried, I've switched to a much simpler approach to viewing RSS feeds.

In my last article, I started a series called Command-Line Cloud. The intent of the series is to discuss how to use the cloud services we are faced with these days without resorting to a Web browser. I spend most of my time on the command line, so that's where I'd most like to interface with cloud services. My last article described how to use Google Calendar from the command line, and in this article, I talk about a more general cloud service—RSS feeds. If I had written this column a few months ago, it would have been more focused on replacing Google Reader itself, because that was the primary RSS aggregator I used, but Google preemptively killed off the service and left a lot of users, including myself, scrambling to find a replacement. Although a number of people were able to find some sort of Web-based replacement, I realized the main features I wanted (sorting stories

by date and vi key bindings to view the next story) were absent in a lot of the existing Google Reader replacements. What's worse, a lot of people were using this as an opportunity to make a quick buck by selling access to RSS services (and of course, still capturing everyone's valuable Web-viewing habits).

I decided to take a completely different tack and convert my RSS feeds to e-mail in a special mailbox and use an interface I already was used to: e-mail on the command line using mutt. I decided to use the rss2email program, written by the great Aaron Swartz, to manage my feeds. This software pulls down RSS feeds and converts each story into its own e-mail message that it sends to you. This means you can use whatever e-mail program you want to read your feeds, but of course, because we are focusing on the command line here, I am going to talk about only mutt.

Installation and Configuration

The rss2email program already had Debian packages, so on my system, installing it was as easy as typing `apt-get install rss2email`. If for some reason it isn't packaged for your distribution, follow the steps on <http://www.allthingsrss.com/rss2email/getting-started-with-rss2email> to download and extract the rss2email tarball. This is Python software, so you will need Python 2.x on the system as well as some sort of local Sendmail program (Postfix or Exim works as well), or alternatively, you'll need to identify an outbound mail server you can use to send these e-mail messages.

Once rss2email is installed, you interface with it via the `r2e` command. To set up a new rss2email database containing your feeds, type:

```
$ r2e new youremail@yourdomain.net
```

Note that the e-mail address you use here will be the e-mail to which rss2email will send the e-mail messages. Once the database is set up, it's time to add feeds to it. You can do that with:

```
$ r2e add http://feed.someurl.com/rss
```

Or, if you happen to have an OPML

file (such as you may have exported from Google Reader when you jumped ship), you can import that:

```
$ r2e opmlimport file.opml
```

At any point, you also can export all of your configured feeds from rss2email as an OPML file:

```
$ r2e opmlexport
```

Once you have added some feeds, you will want to poll them for new stories. Now the first time you run `r2e` against these feeds, it will pull in all stories in the feed, which probably includes some you already have seen. If you want to avoid that, the first time you will want to run:

```
$ r2e run --no-send
```

Otherwise, run:

```
$ r2e run
```

The first time it may take a while, because it reads all of your feeds and generates e-mail. Of course, by default, it will send all the stories to your INBOX, so because I control my own mail server, I created a special e-mail address for rss2email to use and then set up a procmail

rule so that it forwarded all e-mail messages sent to that address to a special rss mailbox.

Of course, rss2email updates your feeds only when you run the command, so you probably will want to run this within cron so it updates automatically. Just run `crontab -e` as your regular user and add:

```
* * * * * r2e run 2>/dev/null
```

This will run r2e every minute and output any random error (such as when feeds are temporarily down) to /dev/null instead of sending you e-mail every time. For the most part, rss2email works as is, but in my case, I wanted to change two extra settings. To do this, just open up `~/.rss2email/config.py` in a text editor and add the following settings:

```
HTML_MAIL = 1
DATE_HEADER = 1
```

The first setting tells rss2email to send the e-mail as HTML mail, and the second dates the message based on the date of the news story, not the date it picked it up. Although you might be surprised that I opt for HTML e-mail in my text-based e-mail client, mutt automatically converts HTML e-mail to text for me. Plus,

when I tell mutt to open the e-mail in an external viewer, on pure shell sessions, it means I can view the full article in a text-based Web browser, such as w3m, very easily. And, when I run mutt on a machine with a Web browser, it can open the full article there instead.

Managing Feeds

Managing feeds in rss2email is relatively straightforward. First type `r2e list` to see a numbered list of all of your feeds. You will use the number associated with a feed to manage it. For instance, to get rid of a feed numbered 12, you would type:

```
$ r2e delete 12
```

You also can pause feeds if you want to ignore them temporarily with `r2e pause number` and then unpause it with `r2e unpause number`.

Mutt as the Front End

What makes this set up work so well for me is that I can use my regular mail program, mutt, to view my feeds. This means I quickly can scan my feeds and skip uninteresting or duplicate stories. In my case, I found I needed to tweak how mutt displays the index for this mailbox so I more easily could see who the feed

ManageEngine[®] ServiceDesk Plus

ITIL-READY HELP DESK SOFTWARE

IT MADE EASY



Incident Management . Problem Management
Change Management . Service Catalog
Asset Management . Project Management . CMDB

www.servicedeskplus.com | demo.servicedeskplus.com



Big Data gets real at Big Data TechCon!

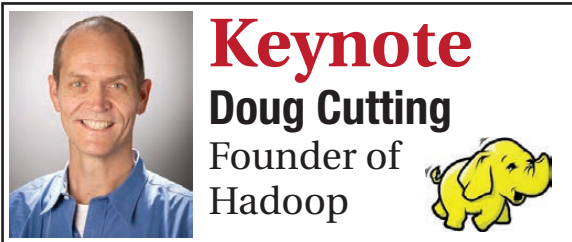
Discover how to master Big Data from real-world practitioners – instructors who work in the trenches and can teach you from real-world experience!

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

- Collect, sort and store massive quantities of structured and unstructured data
- Process real-time data pouring into your organization
- Master Big Data tools and technologies like Hadoop, Map/Reduce, NoSQL databases, and more

Over 60
how-to
practical classes
and tutorials
to choose
from!

- Learn HOW TO integrate data-collection technologies with analysis and business-analysis tools to produce the kind of workable information and reports your organization needs
- Understand HOW TO leverage Big Data to help your organization today



“Big Data TechCon is great for beginners as well as advanced Big Data practitioners. It’s a great conference!”

—Ryan Wood, Software Systems Analyst, Government of Canada

“If you’re in or about to get into Big Data, this is the conference to go to.”

—Jimmy Chung, Manager, Reports Development, Avectra

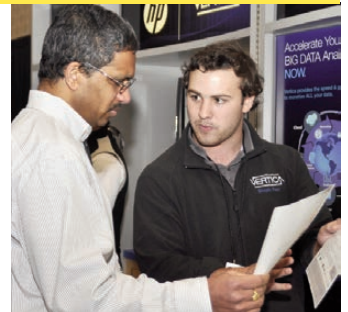
BigData TECHCON

San Francisco

October 15-17, 2013

www.BigDataTechCon.com

The **HOW-TO** conference for Big Data and IT professionals





SHAWN POWERS

It's a Bird. It's Another Bird!

Build your own backyard Ustream—sorta.

My new full-time job is one that I can do from my home office. One of the perks of working at a home

office is that an office with a window is almost guaranteed. Because I have an office window for the first time in



Figure 1. BirdTopia as Seen via BirdCam

So, what does my obsession with bird watching have to do with Linux?

my career (not counting the one year I had a part-time office facing the dumpster), I figured it would be the perfect opportunity to put up some bird feeders.

Unfortunately for my family, but very fortunately for the local birds, when I decide to do something, I usually go all in. Rather than a simple feeder with mixed bird seed, I decided to get various types of feeders, specialized seed, a bird bath with flowing water and trees planted for shade and cover. My family refers to the area outside my office window as “BirdTopia” (Figure 1). And they haven’t even seen the heated bird bath and peanut feeders I have planned for winter!

So, what does my obsession with bird watching have to do with Linux? Well, obsession demands that either I stare out my window all day and lose my job, or I figure out some way to watch my birds while staring at a computer screen. Enter: BirdCam. I needed a way to stream a live video feed of BirdTopia, without spending any more money. (The “not spend money” part was implied by my wife.)

The Camera

Because I don’t share an office with anyone, my camera options didn’t have to be pretty. I considered a USB Webcam, but all the Webcams I have are really low quality.

Thankfully, I have a drawer full of old cell phones that have been replaced with newer models. I had three iPhone 3GS handsets and a Samsung Galaxy S2 with a cracked screen. The iPhones seemed to be in better shape, so first I tried to use one of them. I purchased a \$5 application called iWebcamera, which turns an iOS device into an IP camera with a built-in Web server. Unfortunately, the iPhone 3GS has a pretty cruddy camera, so although the application worked well, I wasn’t satisfied.

Next up was my Galaxy S2 phone with the cracked screen. Obviously the crack didn’t matter, and the camera is much nicer. Also, the Google Play store has an app called IP Webcam that is completely free and completely awesome. The application puts a big-ugly ad on the screen of the phone, but the remotely viewed video has no ads



Figure 2. This suction cup is an improvement over my original “lean against the window” design.

at all. I highly recommend using an old Android device instead of using an old iOS device, if you happen to have the choice. I mounted the phone on the inside of my office window using a suction-mount cradle designed for a car (Figure 2).

Viewing

Both the iOS app and the Android

app have a built-in Web server that allows for direct viewing of the video stream (Figures 3 and 4). The iOS application’s interface is far less advanced than the free Android program, but they both allow for either viewing the mjpeg video stream or a real-time snapshot. With the Android application (which is what I focus on from here out,

ipCam - Links

Note : Some links may not be supported by all browsers.

Web Browser Links

- [JPEG Video](#)
- [MJPEG Video](#)

Raw Video Links

- [JPEG Image](#)
- [MJPEG Video](#)

Help

- [ipCam Support](#)

Figure 3. The iOS Web interface is functional, but sparse.

because it's free, Linux-based and far better), the resolution of the full-motion video is less than that of the photo snapshot.

The built-in Web server on the phone is probably sufficient if you just want to watch from one or two computers on your network. For

me, however, it wasn't enough. I wanted to view my bird feeders from multiple computers, both internal and on the Internet. I also wanted to be able to share my BirdCam with the world, but I wanted to serve everything myself, rather than depend on a service like Ustream.

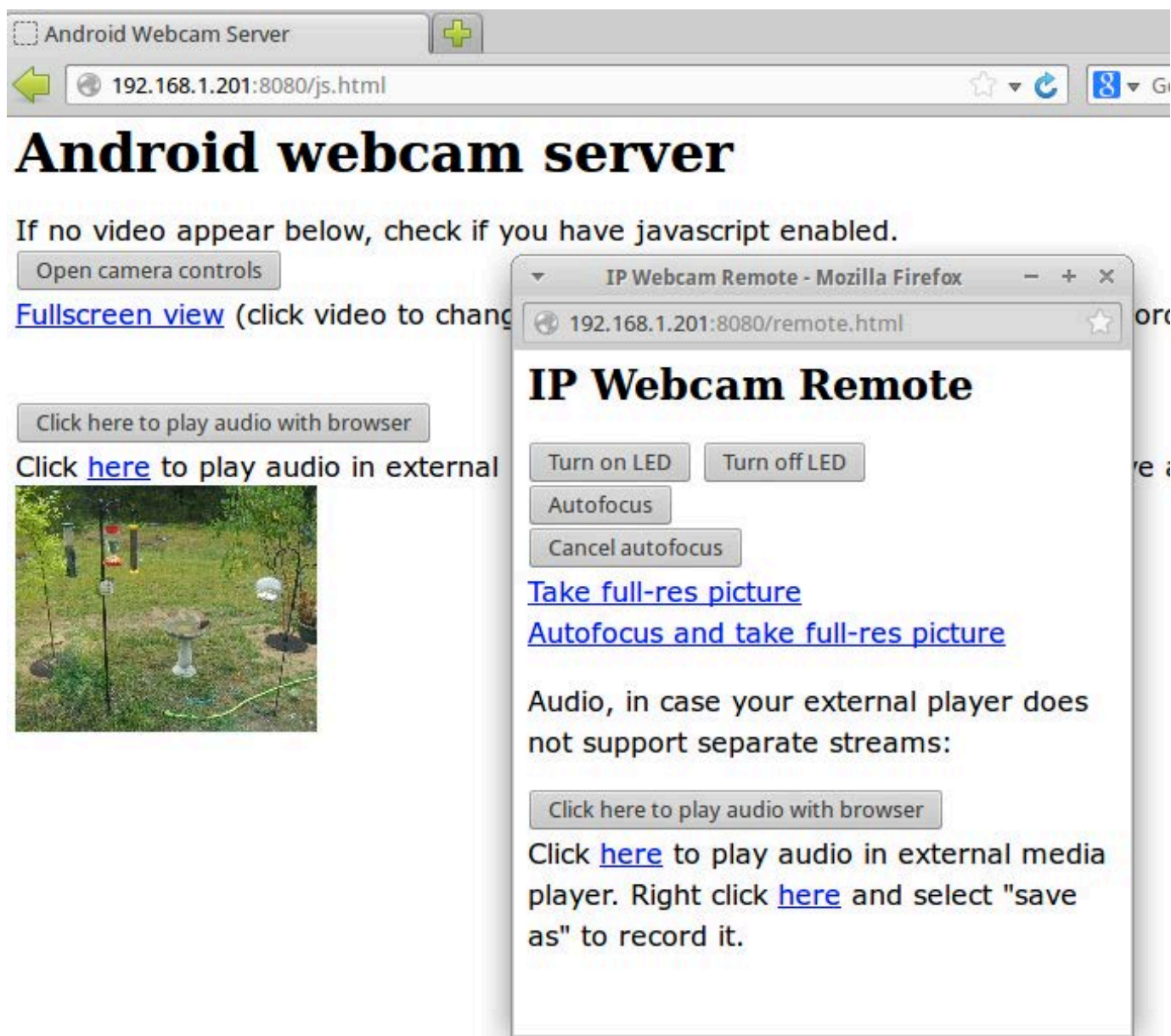


Figure 4. The Android Webcam software is far more robust.

And, that's where things started to get really, really fun.

Video Is for Chumps

Well, it's for chumps with insane bandwidth anyway. Although my business Internet connection here

in my home office has 5Mbit upload speeds, it turns out that streaming multiple video feeds will saturate that type of bandwidth very quickly. I also had the problem of taxing the embedded Web server on the phone with more than one or two

connections. I still hadn't given up on full streaming, so my first attempt at "Global BirdCam" was to re-encode the phone's video on my Linux server, which would be able to handle far more connections than an old Android handset.

Thankfully, VLC will run headless and happily rebroadcast a video stream. Getting just the right command-line options to stream mjpeg properly proved to be a challenge, but in the end, this long one-liner did the trick:

```
cvlc http://PHONE_IP:8080/videofeed --sout \
'#std{access=http{mime=multipart/x-mixed-replace; \
boundary=7b3cc56e5f51db803f790dad720ed50a}, \
mux=mpjpeg,dst=0.0.0:2000/}'
```

The `cvlc` alias just starts VLC headlessly. The `mime` and `boundary` stuff took the longest to figure out. Basically, I had to get that right, or Web browsers would just try to download a file instead of playing a stream. This method did work, actually, and I could connect multiple clients to the server on port 2000 and get the remuxed stream without overtaxing the phone. (The phone served out only the single feed to the server, and the server is far more robust.) Unfortunately, that didn't solve my bandwidth issue.

My Flipbook Solution

Although the VLC solution did work, it didn't really fit my needs. I couldn't stream to the Internet due to lack of bandwidth, and even if I could, my server could handle only a handful of clients before it petered out as well. What I ended up with as my final solution is rather elegant and very efficient.

You may recall I said the Android application allows for high-resolution snapshots to be taken along with a direct video feed. Rather than streaming video, I figured if I took a high-res photo every second, I could get a far better image and also save boatloads of bandwidth. I still wanted a video-like experience, so I concocted a handful of scripts and learned some JavaScript to make a sort of "flipbook video" stream on a regular Web page. This was a two-part process. I had to get constantly updated photos, plus I had to build a Web page to display them properly.

Step One: Getting the Photos

My first instinct was to use a cron job to fetch photos regularly from the Android phone. Because cron jobs run only every minute, I dismissed my first plan right away. I didn't need full-motion video, but "One Frame Per Minute" is pathetic by any standard.

Listing 1. bird_update Script

```
#!/bin/bash
while true
do
    bird_getphoto
    sleep 1
done
```

Listing 2. bird_getphoto Script

```
#!/bin/bash
#Variables -- change to fit your needs
ORIGINAL_PHOTO=/dev/shm/birdtemp.jpg
MODIFIED_PHOTO=/dev/shm/birdmod.jpg
FINISH_PHOTO=/dev/shm/birds.jpg
CAMERA_IP=192.168.1.201
PHOTO_URL=http://192.168.1.201:8080/photo.jpg

if eval "ping -c 1 $CAMERA_IP > /dev/null"
then
    /usr/bin/wget -r --timeout=10 --quiet -O \
        $ORIGINAL_PHOTO \
        "$PHOTO_URL"

    convert $ORIGINAL_PHOTO \
        -quality 70% \
        -pointsize 64 \
        -fill white \
        -annotate +675+60 " `date +%I:%M:%S %p`" \
        $MODIFIED_PHOTO

    rm $ORIGINAL_PHOTO
    mv $MODIFIED_PHOTO $FINISH_PHOTO
fi
```

I ended up making a few scripts, one of which I launch via rc.local on system boot (Listings 1 and 2).

The first script, bird_update (Listing 1), is started via rc.local on my server. I could have called the larger script directly from rc.local and had it loop, but this way, I could make a change to the main script (bird_getphoto, Listing 2) and not worry about restarting the rc.local stuff. bird_update runs bird_getphoto, sleeps for a second and starts over. That means if I make a change to bird_getphoto, the changes would be reflected on the next iteration of the loop, with nothing to start over. Since I tweaked bird_getphoto about 6,000 times, this method was ideal.

The bird_getphoto is what does the “dirty work” so to speak. Stepping through the commands should be fairly self explanatory, but basically:

1. See if phone is on-line.

2. Get photo from phone (with a short timeout—by default, timeout is absurdly long, and an occasional hiccup shouldn't stall the entire system).
3. Store photo in ramdisk. I did this to save on hard drive wear. I figure I'm saving a file every second, and it's silly to do that to spinning media every time.
4. Compress and annotate the photo. At first I had my phone in portrait mode, so I had to rotate as well. The `convert` program, which is part of the ImageMagick package, is very powerful. I added a timestamp to the photo, mainly because I could.
5. After download and conversion is complete, `mv` the temporary file to the live image. I added this extra step, because if `convert` stores directly to the final filename, it gets displayed as a corrupt image if the Web server tries to serve it out during the conversion process. The `mv` command is almost instantaneous, so I haven't seen any weird corruption after adding the extra step.

I'm almost embarrassed to admit

how long it took me to fiddle around with commands, ideas, loops and so forth before coming up with the scripts shown here. As with all my articles, please feel free to change and/or improve on my ideas to best fit your needs. These scripts have been running smoothly for weeks now, and they seem to be fairly bulletproof when it comes to network failures and such. Getting the photos regularly updated, however, was only half the problem—as it turns out, the easier half.

Step 2: JavaScript, and Breaking the Internet

In order to display the constantly updated bird photos, it was easy enough to create a symbolic link from `/dev/shm/birds.jpg` to `/var/www/birds/`, where my Apache virtual host folder was located. I created a simple HTML file with an `img` tag, and I could see my bird feeders from anywhere. In order to get a refreshed image, however, I had to refresh the entire Web page. It worked, but it was ugly. I had to reach out for some JavaScript help.

Before getting to the final HTML file, it's important to explain that while getting JavaScript to refresh a single image on a page isn't terribly difficult, browsers are designed to cache as much as possible, so making

Listing 3. birds.html

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  ⤵ "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>The birds. Or not.</title>
<script type="text/javascript">
refreshImage = function()
  {
    img = document.getElementById("cam");
    img.src = "http://example.com/birds.jpg?rand="
      ⤵+ Math.random();
  }
</script>
<meta http-equiv="Content-Type" content="text/html;
  ⤵ charset=iso-8859-1" />
</head>
<body onload="window.setInterval(refreshImage, 1*2500);">
<center>

<br />
<small><em>This should constantly refresh</em></small>
</center>
</body>
</html>

```

sure the image was actually fetched from my server every couple seconds proved challenging. Listing 3 shows my final HTML file.

A large part of the top of the file shown in Listing 3 is just defining the specific HTML standards in use. I'll be honest, most of it is over my

head. The key part of the script is the JavaScript code, which defines an action for an image with a specific ID. You can see the ID is "cam" in the JavaScript and in the img tag below. The peculiar part of the script is the bit of random info after the photo URL. That's actually the part

of the script that not only reloads the image every 2.5 seconds, but also loads the image with a `?rand=RANDOMNUMBER` at the end. That's basically me fooling the browser into thinking there is a new image to download every time, so I don't get shown a cached image. There are several ways to do such a thing, but this proved to be the simplest and most cross-browser-friendly in my testing. There is concern of filling up caches or buffers on the server, but so far I haven't experienced any issues.

The End! Or Is It?

To fulfill my personal needs, the bash scripts and the bit of JavaScript really did all I needed. I can view the BirdCam from multiple computers in my house, and even from the Internet. Each "video frame" is around 600K, so although it still uses significant bandwidth, it's nothing like trying to stream full video. I have noticed that with slow cellular connections, sometimes the image freezes, because it tries to refresh before the original image is loaded. I settled on

2500 milliseconds as the refresh time, because it seems to work from most



Figure 5. With the ability to view BirdCam remotely, I can get some great shots, even when I'm not home!

locations (Figure 5).

If you visit my BirdCam now (<http://birds.brainofshawn.com>), you'll probably notice I've done a little more tweaking. I've added a query to the local weather station, and I added the current temperature to the annotation. The big change, however, is one you hopefully shouldn't notice.

Because I knew I'd be sending this to tens of thousands of potential bird watchers, I figured I probably should scale my solution to the cloud. Granted, 5Mbit upload speed is decent, but not if 100 people are trying to check out my backyard!

My simple solution was to replace the `sleep` command in the `bird_update` script with an `scp` command that uploads the `bird.jpg` file to my Web hosting provider. I still try to be a good netizen and upload the actual file to the ramdisk on my provider's server, but with the `.jpg` file being served from the cloud, I'm not worried about an influx of bird watchers. If you

were worried about saturating my home connection, fear not; I planned ahead.

Winter Is Coming

If you set up a similar Webcam, I'd love to hear about it. You're also welcome to watch the exploits of my bird obsession as winter approaches. The bubbling bird bath soon will be replaced with a



Figure 6. Bird water just tastes better.

heated version, and I'll be adding more suet and peanuts for the winter birds. If you happen to be watching, and see a cool bird, don't hesitate to send me the photo! I had a visit from an Indigo Bunting one day that I saw via BirdCam, but that was when I was using an old iPhone, so the image is very poor quality. BirdTopia attracts more than just birds too. While the occasional squirrel is brave enough to visit, the most common non-aviary visitor is Zoey (Figure 6).

I wish you an awesome weekend project! I have enjoyed playing with BirdCam almost as much as I enjoyed

making a MAME cabinet years ago. There's just something about building with Linux that warms my heart. ■

Shawn Powers is the Associate Editor for *Linux Journal*.

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

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

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



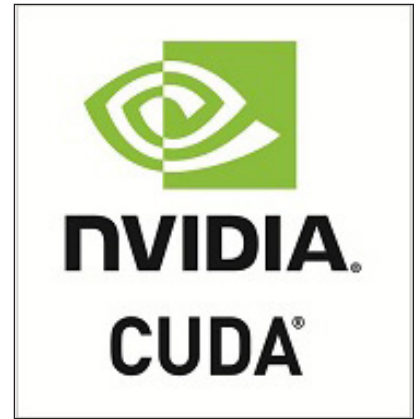
Subscribe now for instant access! For only \$29.50 per year—less than \$2.50 per issue—you'll have access to *Linux Journal* each month as a PDF, in ePub format, in Mobi format, on-line and through our Android and iOS apps. Wherever you go, *Linux Journal* goes with you.

SUBSCRIBE NOW AT:
WWW.LINUXJOURNAL.COM/SUBSCRIBE

NVIDIA CUDA

Version 5.5 is the latest and greatest iteration of NVIDIA CUDA, NVIDIA's parallel computing platform and programming model that harnesses the power of GPUs. CUDA 5.5, sayeth NVIDIA, provides programmers with a robust, easy-to-use platform to develop advanced science, engineering, mobile and HPC applications on x86 CPU-based systems, and now on ARM-based ones as well. Beyond native support for ARM platforms, CUDA 5.5 delivers a number of new advanced performance and productivity features, including enhanced Hyper-Q support, MPI workload prioritization, and new guided, step-by-step performance analysis and fast cross-compile on x86. The latter feature enables developers to compile ARM code on fast x86 processors and transfer the compiled application to ARM. The v5.5 release also offers a full suite of programming tools, GPU-accelerated math libraries and documentation for both x86- and ARM-based platforms.

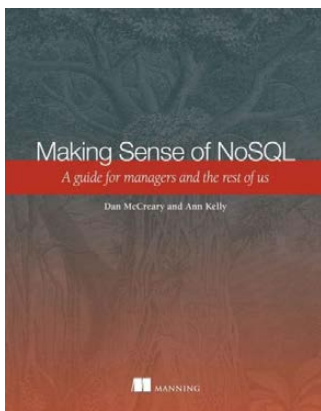
<http://www.nvidia.com>



Gumstix's Geppetto Platform

Gumstix continues to add fire power and options to Geppetto, the company's modular, drag-and-drop platform for designing and building customized single-board Linux solutions. The most recent enhancement to Geppetto involves the addition of the Texas Instruments Sitara AM3354 processor, which Gumstix says will offer more flexibility to users and raise Geppetto's ability to enable a rapid go-to-market strategy for its customers. While designing a board using Geppetto, users simply drag and drop the processor onto a board and then connect the desired features to implement it. Fully assembled single-board computers are ordered at the touch of a button and arrive within 20 business days. Furthermore, notes Gumstix, Geppetto's support for the Yocto Project build system makes it easy for developers to create a complete, portable solution with minimal time and effort.

<http://geppetto.gumstix.com>



Dan McCreary and Ann Kelly's *Making Sense of NoSQL* (Manning Publications)

NoSQL tools like MongoDB, Neo4j and Redis take innovative approaches to the unique problems of handling data in modern distributed and Web-based systems. The new book, *Making Sense of NoSQL: A guide for managers and the rest of us* by Dan McCreary and Ann Kelly, is a resource for learning about NoSQL solutions. According to publisher Manning Publications, the book clearly and concisely explains the concepts, features, benefits, potential and limitations of NoSQL technologies. Using examples and use cases, illustrations and plain, jargon-free writing, this guide shows how one can assemble a NoSQL solution to replace or augment a traditional RDBMS effectively. After reviewing database concepts alongside the new NoSQL patterns, authors McCreary and Kelly explore topics including Big Data, search, reliability, business adaptability, cloud computing, large CPU-count data centers and customized solutions.

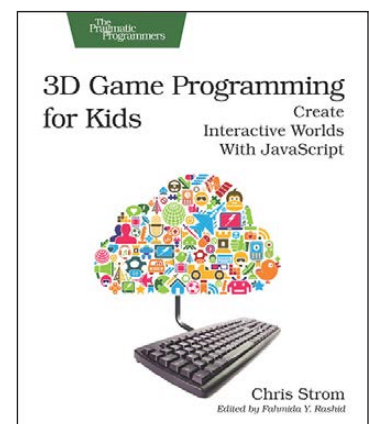
<http://www.manning.com>

Chris Strom's *3D Game Programming for Kids* (Pragmatic Programmers)

What's even better than playing games? Creating your own, of course. And thus you shall be so empowered if you digest the contents of Chris Strom's new book *3D Game Programming for Kids: Create Interactive Worlds with JavaScript*. Targeted at younger readers, Strom's book illustrates how to create

on-line games. Using nothing more than a browser and the language of the Web, JavaScript, readers will learn programming and visualize cool, 3-D results as they type. To make things easier for beginners, the kid-friendly ICE Code Editor was created especially for this book. Want a red donut? Make hundreds of them, spinning around like crazy, right next to the entered code. Readers can create games quickly by focusing on the project-based lessons. If they want to go further and understand the theory or mathematical functions, they can turn to the chapters that explain the programming concepts.

<http://www.pragprog.com>

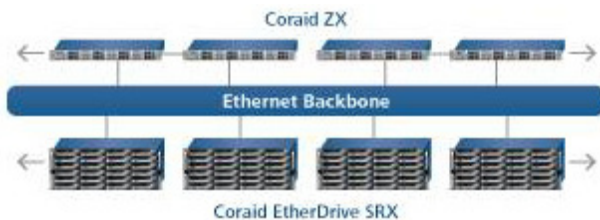


Directory Wizards Inc.'s UnitySync



Into the big Linux tent we welcome Directory Wizards, whose UnitySync solution is now available on the Linux platform. UnitySync is a centralized service that synchronizes data between directories so that organizations enjoy a unified view of their disparate directories. Our sources at Directory Wizards say that UnitySync can scale from small directories with hundreds of objects to enterprise directories consisting of hundreds of thousands of objects without requiring extensive training or installation/configuration time. UnitySync users can synchronize account information between different directories so that each directory contains a unified view of the other non-connected directories. In addition, users can define authoritative data sources, enabling one system to update individual attributes of existing objects of another. No scripting, programming or extra databases are required. The Windows Intel 32-bit platform also is supported.

<http://www.dirwiz.com>

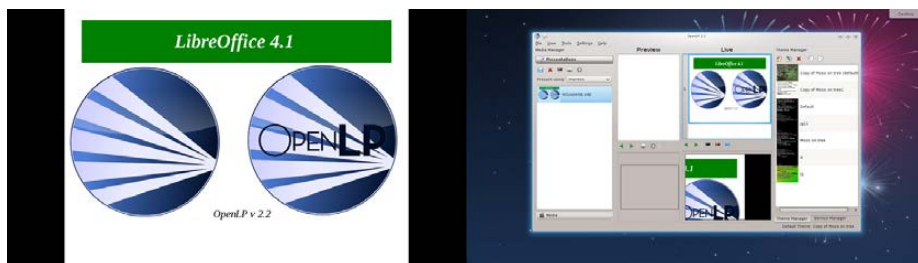


Coraid ZX4000 File Storage Appliance

Public cloud scalability, economics and resilience models are three macroforces

affecting IT. Storage specialist Coraid's response to these trends is the new ZX4000 File Storage Appliance, a storage building-block solution for both enterprise and cloud deployments that combines Flash performance, multi-petabyte scale and distributed resilience. Coraid suggests that the ZX solution's unique architecture allows for a potentially limitless pool of elastic, shared, block storage that easily scales performance via the addition of more ZX appliances. The ZX4000 can be deployed in a resilient architecture that offers protection against up to three shelf or drive failures while retaining high storage efficiencies of around 80%. This enterprise-class storage solution, says Coraid, is available at a cost comparable to consumer-grade public cloud services and at a fraction of the cost of legacy NAS storage. Since its introduction in 2012, the ZX product line has been deployed by customers in a wide variety of applications including backup, public and private clouds, and media.

<http://www.coraid.com/products>



The Document Foundation's LibreOffice

The Document Foundation proudly calls its new

LibreOffice 4.1 office suite “not only the best but also the most interoperable free office suite ever”. The 4.1 milestone release features a large number of improvements in the area of document compatibility, which increases the opportunities of sharing knowledge with users of proprietary software while retaining the original layout and contents. Numerous improvements have been made to Microsoft OOXML import and export filters, as well as to legacy Microsoft Office and RTF file filters. Other new interoperability-focused features include font embedding in Writer, Calc, Impress and Draw and import and export functions new in Excel 2013 for ODF OpenFormula compatibility.

<http://www.libreoffice.org>

MWR InfoSecurity's drozer

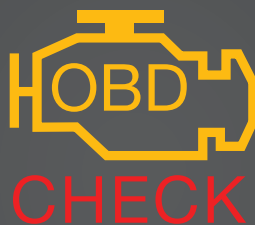
The Android security testing framework formerly known as Mercury—and now called drozer—was released recently by MWR InfoSecurity. With drozer, companies using Android mobile devices now can safeguard their assets and IT infrastructure by running full, dynamic security assessments. A new drozer feature is the ability to compromise Android devices through publicly available exploits, something that helps organizations understand how a technical vulnerability on a mobile device can become a real threat to their business. Drozer unifies these publicly available exploits into a single framework and improves the quality of the exploitation code and payloads available to the penetration tester. Drozer provides support for any Android device running Android 2.1 and beyond, covering 99% of the devices in the market. The open-source tool is available to download from the MWR Labs Web site.

<http://www.mwrinfosecurity.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.

BE A MECHANIC...

with **ANDROID** and **LINUX!**



“Check Engine Soon”—that little orange light on your car’s instrument panel is possibly one of the more annoying things about modern automobiles. Ever had it pop on during a trip and wonder whether it was just something mundane, like your gas cap being loose, or whether it’s something deathly serious and a piston could come shooting out the side of your engine block at any time? Well, thanks to an inexpensive little piece of hardware and an Android tablet, I’ll help you decode that little orange light in your car.

BILL CHILDERS

The human race has had automobiles for more than 100 years now, but we've had computer monitors and control engine operation only for around 30 years or so. The first computer controls were primitive, hard to work with and expensive. Each automotive manufacturer had its own computer systems, protocols, connectors and trouble-code definitions. I worked as a mechanic during the late 1980s and early 1990s, and I remember those systems well—not fondly, of course, but well. Some of those systems required you to do crazy things like jump a connector with a piece of wire, then turn the key on and off three times and observe the Check Engine light as it flashed on and off. You'd have to count the number of flashes accurately and then look up the "trouble code" that flashed in a service manual, and you *might* get a clue as to what was wrong with the vehicle. Those early diagnostic systems made seasoned mechanics who were used to troubleshooting the machinery of an engine rather than its electronics shudder with trepidation. Over time, the manufacturers made the systems better. The Society of Automotive Engineers made the connector, protocol and trouble codes a standard

in 1996, and with that, we've got the system in place today: OBD-II (Onboard Diagnostics, 2nd revision).

OBD-II Basics

Any car sold in the United States after 1996 uses the OBD-II computer system, so the majority of cars on the road today have this system. Thanks to OBD-II's standardization and age, lots of tools have been released to work with the system. Because OBD-II defines the connector and protocol, that means you need both a hardware device to interface with the connector and some software to speak the protocol.

The hardware I use is the Soliport ELM327 Bluetooth OBD-II Scanner (see the Amazon link in the Resources section of this article). It's a very inexpensive (less than \$20) dongle that plugs in to the OBD-II port under your dashboard, draws its power directly from the car and converts the OBD-II-specific signals to serial-over-Bluetooth. There are other OBD-II scan tools on the market. Some are just plain-old cables to hook straight into a computer's RS-232 serial port, and others are as fancy as full-on bridges to a Wi-Fi network. And, there are other manufacturers of Bluetooth OBD-II scan tools, but just make sure whatever you get is based on the

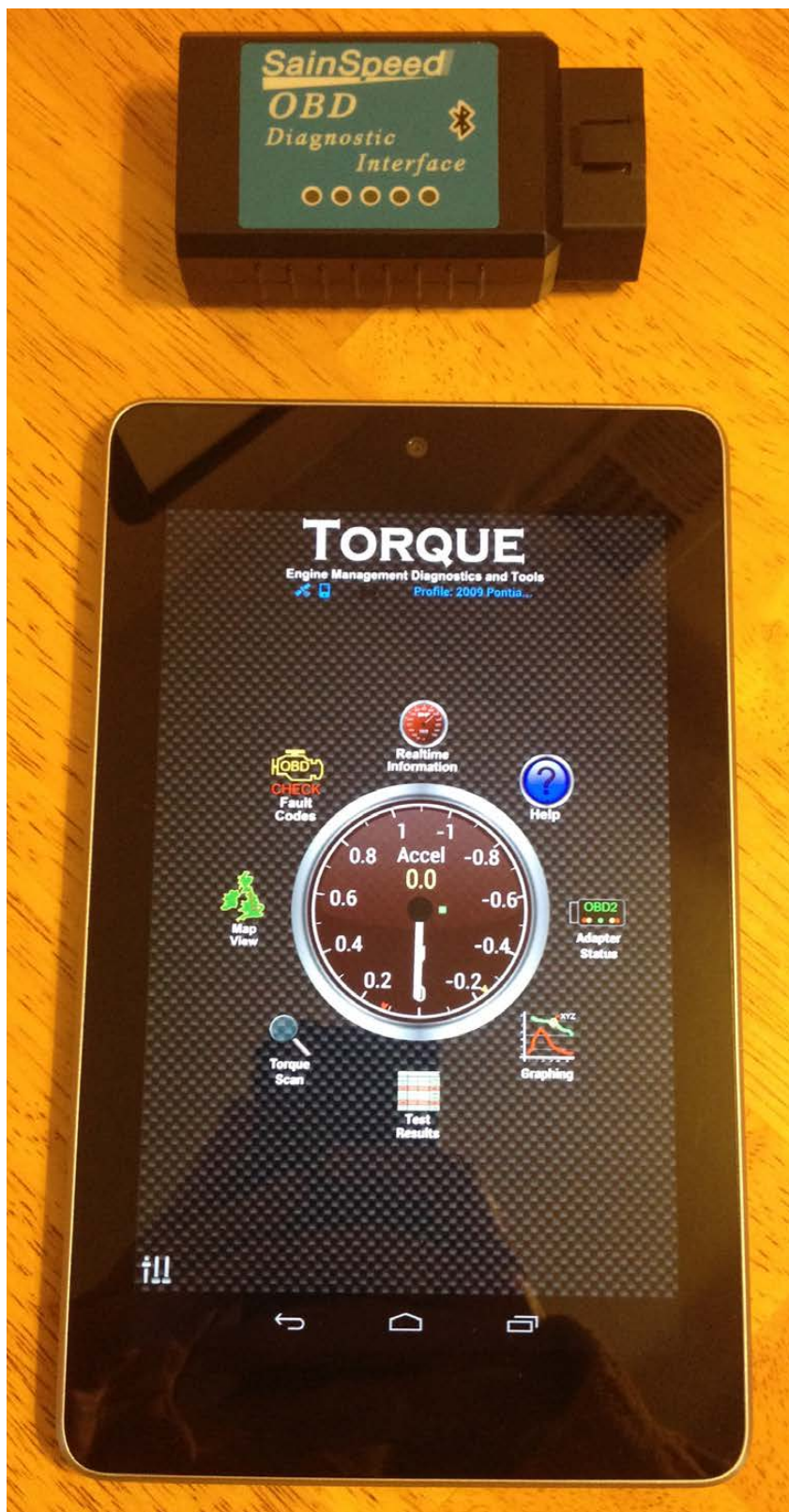


Figure 1. Tools of the Trade: a Nexus 7 Tablet and a Soliport Bluetooth OBD-II Scanner

ELM327 chipset.

Another piece of hardware you'll need is a computer of some kind. Any Linux-powered laptop with Bluetooth will suffice, but the form factor is kind of clunky when you're dealing with a cramped automotive cockpit. (See the Using a Laptop to Scan Your OBD-II System sidebar if you want to use a laptop.) My personal preference is to use an Android device to interpret the signals coming from the OBD-II system in the car. I use a Nexus 7 tablet for this, but any Android device should work. I've used a Motorola Droid RAZR and the very first HTC G1 Android phone for this as well.

Software-wise, my choice for this on Android is Torque, an excellent app that not only can collect all the OBD-II stats, but also graph and log them in myriad ways. (There's also a free version, Torque Lite, that has a fair deal of the functionality of the full

Using a Laptop to Scan Your OBD-II System

Although it's possible to use a laptop to do the same duty as an Android device, it's a little more involved, as the Bluetooth protocol stack on a Linux laptop requires some more massaging than simply pairing up an Android device. However, if you're comfortable with the command line and Bluetooth commands like `rfcomm`, it's absolutely possible, and there are some good OBD-II packages like `pyobd` and `openobd`. You won't get some of Torque's value-add features, like accelerometer and GPS integration, but you still can use the laptop for diagnostic purposes and data logging.

Using an Old Nokia Internet Tablet to Scan Your OBD-II System

In the December 2008 issue of *LJ*, I wrote an article called "Hacking the Nokia Internet Tablet", and I talked about ways to hack and extend the Nokia N800 tablet. It turns out there's an application for the N800 and N810 called Carman that was designed to work with wired OBD-II adapters, but it works just fine with the Soliport Bluetooth Scanner. Carman used to be in the Maemo repositories. I no longer have a working N800, so I can't check that now, but when my N800 *did* work, I used it a few times to diagnose the car. So, if you've got a Nokia device sitting in a drawer gathering dust, pull it out and put it to use!

version.) Grab either one from the Google Play store.

Note that this solution is for read-only access to the OBD-II system in the car. You can't modify the running parameters of the vehicle with this adapter, unfortunately (or fortunately,

perhaps, as it's very easy to make a mess of things). Flashing your car's computer with a new fuel curve or ignition timing map is a nontrivial exercise that requires an adapter with different voltage levels and different software. So, don't worry about

breaking your car with this solution—you're just "peeking under the hood".

Using Torque and the Soliport Bluetooth Adapter

The Soliport adapter comes with a little CD-ROM in the box, but it's not required for use with an Android device. To get started, you first need to find the OBD-II port in your car. In most cars sold in the United States, the port is under the dash on the driver's side of the car. Find the port, and plug the Soliport in to it.

Next, start the car, because the

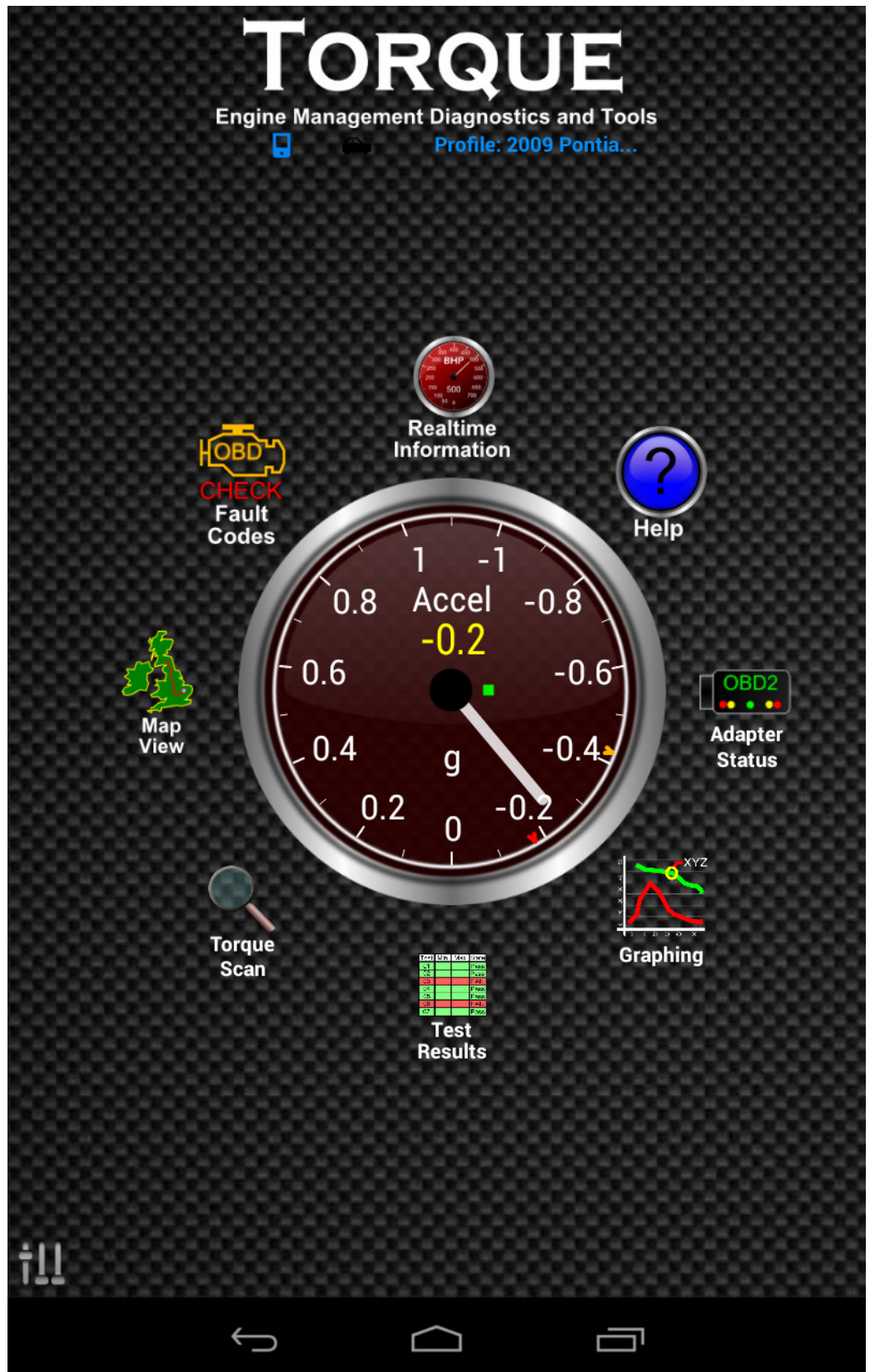


Figure 2. The Torque Home Screen

OBD-II port isn't powered until the car's ignition is on. (Make sure your garage door is open, please. I don't want to receive hate mail from your relatives on how you suffocated from carbon monoxide poisoning!) Next, you need to go through the standard Bluetooth pairing process to pair your Android device to the Soliport adapter. (The pairing code is 1234 if you can't find it in the instructions—coincidentally, it's the same combination that's on my luggage.)

Once you've got your Android device all paired up to the Soliport, you're ready to fire up Torque. Start Torque on your Android device, and you'll be greeted with the Torque main screen.

Setting Up Torque

Now that you have Torque up, select the little "settings" slider on the bottom left of the screen, and select "OBD2 Adapter Settings". Set the connection type as Bluetooth, and choose the Soliport if prompted. Go back to the main settings screen and select your desired units (Imperial or Metric), and any other preferences you choose, then flip back to the Torque main screen.

Next, you're going to create a "profile" for your vehicle. Select the settings slider from the main screen

as before, then select "Vehicle Profile" and "Create new profile". Then, fill in the pertinent information about your vehicle. This information is used by Torque to compute things that can be calculated, like horsepower, fuel economy and other metrics. When you're done, go back to the main screen.

Checking for Fault Codes

Let's start by doing basic diagnostics on your car. From the main screen, select Fault Codes, then press the large magnifying glass to start a scan of your car's computer for trouble codes. If your Check Engine light is on, you'll probably find your issue represented as a code here. My father's 2001 Chevy Silverado pickup was showing a Check Engine light, and I ran a quick scan on it with Torque. It resulted in a trouble code of P1416. A quick Google search of that trouble code showed that it was the Secondary Air Valve, Bank 2. It turned out that was a little smog system valve right on top of the engine, on the passenger side. Amazon.com had that particular part for \$37, and we had it at his house in two days.

I had the Check Engine light on my wife's Durango pop on not long after, and I used the tool to scan her car. Her car came back with a P0440 code,

which means “Evaporative Emission Control System Malfunction”. I searched a bit more on the Internet and found that the most common cause of this code is a broken or mis-installed fuel filler cap. It turned out that was exactly the case—her fuel filler cap wasn’t tightened all the way. I tightened the cap and cleared the code via the Torque app, and it never returned.

Getting Performance Data

Those two cases listed above more than paid for the cost of the Soliport adapter and the

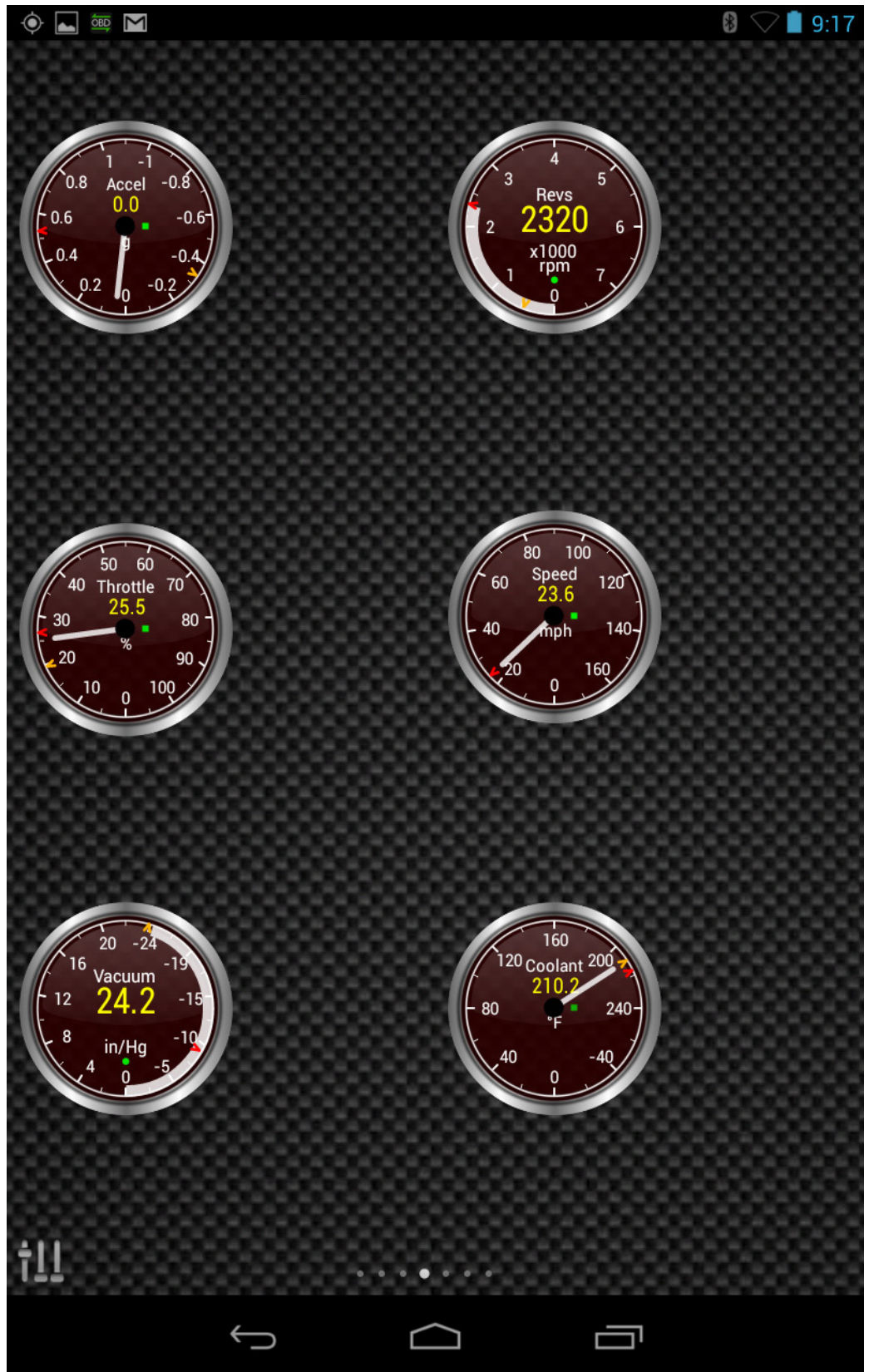


Figure 3. Torque’s Virtual Instrument Panel

Torque application, but Torque can do so much more. Torque can pull data from sources other than your OBD-II sensors. It also can poll your Android device's accelerometer and GPS. This means it can do performance calculations, such as 0–60 mph time (or 0–100 kph time), 1/4 mile time or even horsepower calculations. This requires that you get your car's data entered correctly into the vehicle profile during setup time, particularly the vehicle weight (including your weight as driver, and any other stuff you may have inside the car). If you do performance testing, make sure you're doing it safely—and don't violate any laws in your locality.

However, I think one of the coolest things that Torque and the Soliport adapter can do is they can act as an auxiliary instrument panel for your car. Any bit of information that passes through the OBD-II sensors can be logged, graphed or placed on a digital dial. You can pick and choose how you want that information presented as well—including the size and position of the graphs and dials. This information can be extremely valuable, for instance, displaying the current engine manifold vacuum. As a general rule, under cruise conditions, higher manifold vacuum means higher fuel economy, so having this gauge up

can be handy on long trips.

Torque also has other features, like the ability to log your data for future analysis. It also can graph that data and correlate it to your GPS position and accelerometer data. This can be useful if you happen to be an amateur racer and would like to get information about your car's performance at certain points on the racetrack. Most people won't need that ability, but it's nice to know that the developer of Torque thought that out. All the data necessary to do those

New on
LinuxJournal.com,
**the White Paper
Library**



www.linuxjournal.com/whitepapers

NOVEMBER 3-8, 2013 • WASHINGTON, D.C.

Lucky LISA '13

27th Large Installation System Administration Conference



Keynote Address: “Modern Infrastructure: The Convergence of Network, Compute, and Data” by Jason Hoffman

Join us for **6 days of practical training** on topics including:

- ▶ **SRE Classroom: Non-Abstract Large System Design for Sysadmins** by John Looney, *Google*
- ▶ **Root Cause Analysis** by Stuart Kendrick, *Fred Hutchinson Cancer Research Center*
- ▶ **PowerShell Fundamentals** by Steven Murawski, *Stack Exchange*
- ▶ **Introduction to Chef** by Nathen Harvey, *Opscode*

The **3-day Technical Program** includes:

- ▶ Plenaries by Hilary Mason, *bitly*, and Todd Underwood, *Google*
- ▶ Invited Talks by industry leaders such as Ariel Tseitlin, *Netflix*; Jeff Darcy, *Red Hat*; Theo Schlossnagle, *Circonus*; Matt Provost, *Weta Digital*; and Jennifer Davis, *Yahoo!*
- ▶ Paper presentations, workshops, vendor exhibition, posters, Guru Is In sessions, BoFs, and more!

New for 2013: The LISA Lab Hack Space!

Register by October 15 and save. Additional discounts are available!

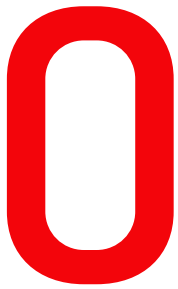
www.usenix.org/lisa2013

Sponsored by  **usenix** ASSOCIATION in cooperation with LOPSA

Create a Mini PC or Server with Olimex's Olinuxino A13/A13Micro

**STEP-BY-STEP
INSTRUCTIONS
ON HOW TO CREATE
A PERSONAL COMPUTER
OR EVEN A SMALL SERVER
WITH THIS GREAT
EMBEDDED SYSTEM
FOR LESS THAN \$80 US!**

RONALD KURNIAWAN



olimex is a Bulgarian company known for its innovative hobbyist products. It has a wide array of microcontroller-based products, ranging

from the small Arduino clones to the very able system that has the Allwinner A13 microcontroller as its brain. In this article, I describe how you can create a working Linux system for the Olinuxino A13 and Olinuxino A13Micro from scratch.

Let's begin by obtaining and compiling the kernel, creating the U-Boot system, preparing the root filesystem and getting the necessary packages to create a comfortable minimal computing environment. At the end of this article, I also explain how to install a compact desktop environment.

I am using Ubuntu 12.04 (Precise Pangolin) for my build system. Any Debian-based system users should be able to follow the instructions in this article with relative ease. Before you begin, you should create a directory under your home directory to contain all your work. I am going to call mine "A13System".

What Are Arm and eabihf?

As you progress further into the article, you will encounter the terms

Arm and eabihf more than once. Let me clarify those terms in order to avoid confusion with other terms that you might encounter if you decide to go further into the world of cross-compilation.

Arm is a general name for a family of microcontroller architectures designed by ARM Holdings, a British company. You can find Arm microcontrollers inside most portable modern gadgets, ranging from mobile phones, Nintendo DS portable game consoles to Apple iPhone and Apple TV. ARM Holdings does not manufacture these microcontrollers; rather, it licenses the designs to other companies. These companies then add their own "secret recipes" into the designs and then manufacture and sell the finished microcontrollers. This is why there are so many variants of Arm architecture and so many companies that produce Arm microcontrollers.

EABI stands for Embedded Application Binary Interface. It specifies the low-level conventions for embedded software application. When it comes to Arm microcontrollers, they come in many sizes, ranging from very small to large. The smaller variants don't have the necessary memory or power to process floating-point computation on the hardware itself, thus making it

necessary to do it by software. These variants are called Arm soft float. There are other variants that can process the floating-point calculation by hardware, like vector floating point (vfp). These two EABIs (soft float and vfp) are what is usually known as *armel*. A newer EABI that targets the higher end of Arm microcontrollers with more efficient floating-point instructions than vfp is called hard float, thus *armhf*.

Olimex's A13WiFi, A13 and A13Micro boards are powered by the Allwinner A13 Arm microcontroller, which are based on ARMv7 design. ARMv7, as with the newer ARMv8, fully supports armhf.

Prerequisites

You can find all the necessary URLs for file downloads and other information in the Resources section of this article. You need to install the following programs before you can commence the build process:

- build-essential.
- gcc-4.6-arm-linux-gnueabi (the version might vary from one distro to another; the latest I came across at the time of this writing is version 4.7).
- ncurses-dev.
- u-boot-mkimage.
- git.
- debootstrap.
- debian-archive-keyring (if you decide later that you want to use Debian rootfs).
- qemu-user-static.

Once you finish installing the prerequisites, you then need to create several softlinks in the same directory where gcc-4.6-arm-linux-gnueabi is installed (in my case, it is located in /usr/bin). Use the which command to find the installation directory:

```
$ which arm-linux-gnueabi-gcc-4.6
```

Next, create softlinks for arm-linux-gnueabi-gcc-4.6, arm-linux-gnueabi-gcov-4.6 and arm-linux-gnueabi-cpp-4.6:

```
$ sudo ln -s /usr/bin/arm-linux-gnueabi-gcc-4.6 \
/usr/bin/arm-linux-gnueabi-gcc
$ sudo ln -s \
/usr/bin/arm-linux-gnueabi-gcov-4.6 \
/usr/bin/arm-linux-gnueabi-gcov
$ sudo ln -s /usr/bin/arm-linux-gnueabi-cpp-4.6 \
/usr/bin/arm-linux-gnueabi-cpp
```

Preparing the Kernel and U-Boot

The good people at Linux Sunxi are kind enough to share the kernel and U-Boot code tailored to run on Allwinner chips. You have the option of getting and compiling version 3.0 or 3.4 of the Linux kernel. The compilation procedures are similar. For the purpose of this article, I am using kernel version 3.4. Get the kernel and U-Boot source from Linux Sunxi's GitHub repository:

```
$ git clone -b sunxi-3.4
https://github.com/linux-sunxi/linux-sunxi.git
$ git clone -b sunxi
https://github.com/linux-sunxi/u-boot-sunxi.git
```

Let's compile U-Boot first. Depending on the target system (A13 or A13Micro), go to the U-Boot directory and issue the following command:

```
$ make a13-olinuxino \
CROSS_COMPILE=arm-linux-gnueabihf-
```

or:

```
$ make a13-olinuxinom \
CROSS_COMPILE=arm-linux-gnueabihf-
```

Note: the dash (-) at the end of the commands are not typos. After the make process finishes,

if everything goes correctly, you should end up with `u-boot.bin` and `spl/sunxi-spl.bin`.

Go to the kernel source directory. Check the configuration directory (`$KERNEL_DIR/arch/arm/configs`) for the A13 configuration file (`a13_defconfig`) or A13Micro (`a13om_defconfig`). If you do not have the configuration file for A13Micro (which is usually the case), you can find the download URL in the Resources section.

Now you need to check the configuration file for a specific line. I learned the hard way that without this line, the compilation will fail. Add the following line to your configuration file if it does not exist or uncomment it:

```
CONFIG_GPIOLIB=y
```

Once again, depending on the target system, issue one of these sets of commands to compile the kernel source:

```
$ make ARCH=arm a13_defconfig
$ make menuconfig
```

or:

```
$ make ARCH=arm a13om_defconfig
$ make menuconfig
```

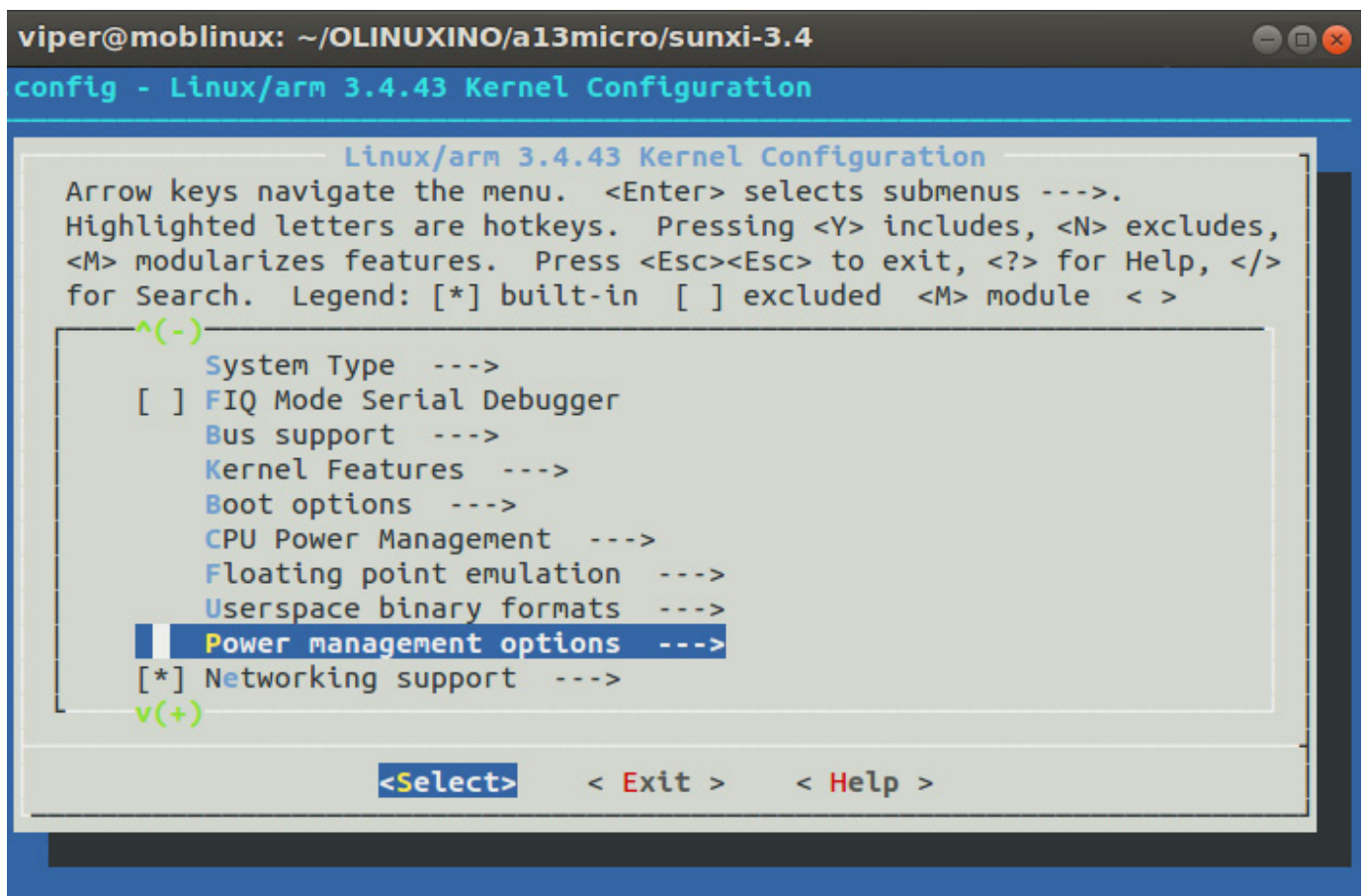


Figure 1. Selecting Power Management Options

The last step allows you to customize your kernel. In order to avoid a long and painful debugging process, always make sure you are able to compile the minimal kernel (that means compiling without any options added) successfully first. Once you succeed, you can add more options to your kernel. Note any options you add to the kernel configuration, as it will aid you in figuring out which feature(s) does not work.

There is a special step you should adhere to if you are compiling the kernel for A13Micro boards. You need to remove the option to "Suspend to RAM and standby", which is located under "Power Management options". A13Micro boards do not support this option.

If you are planning to use any of the board's GPIO pins, make sure that you select "An ugly sun4i gpio driver" option under "Device Drivers" and "Misc devices".

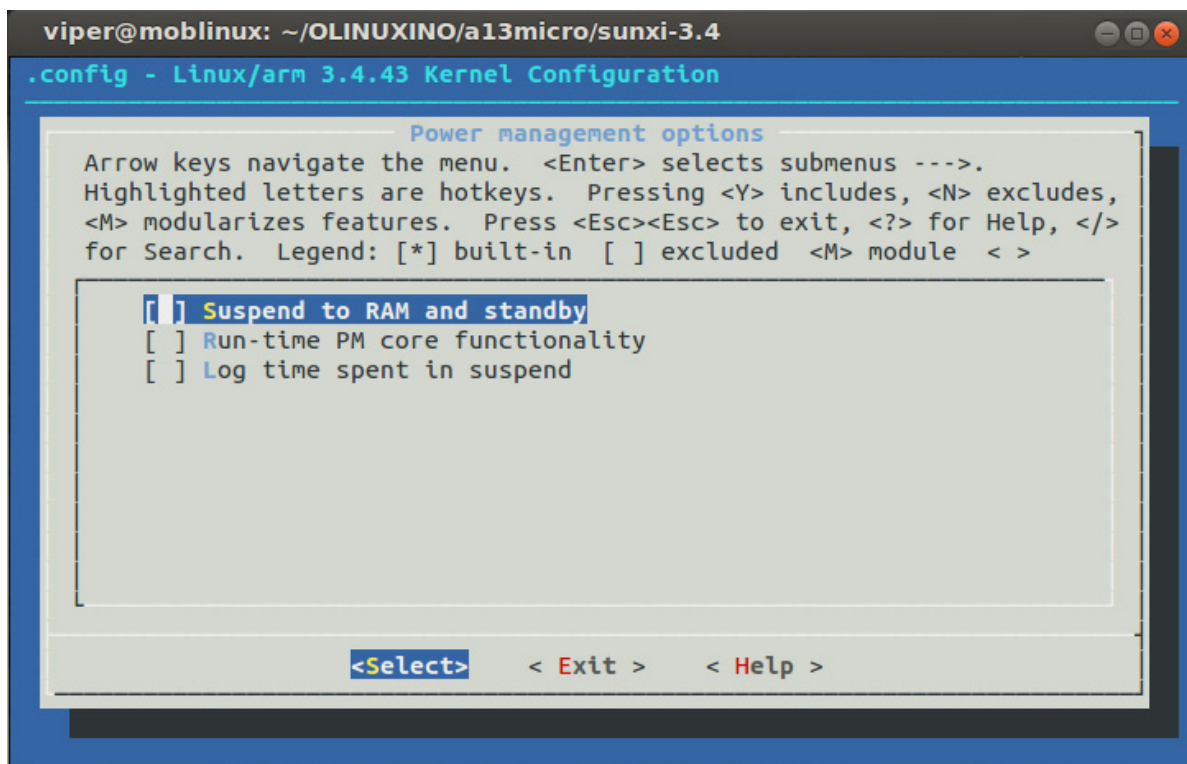


Figure 2. Uncheck Suspend to RAM and Standby

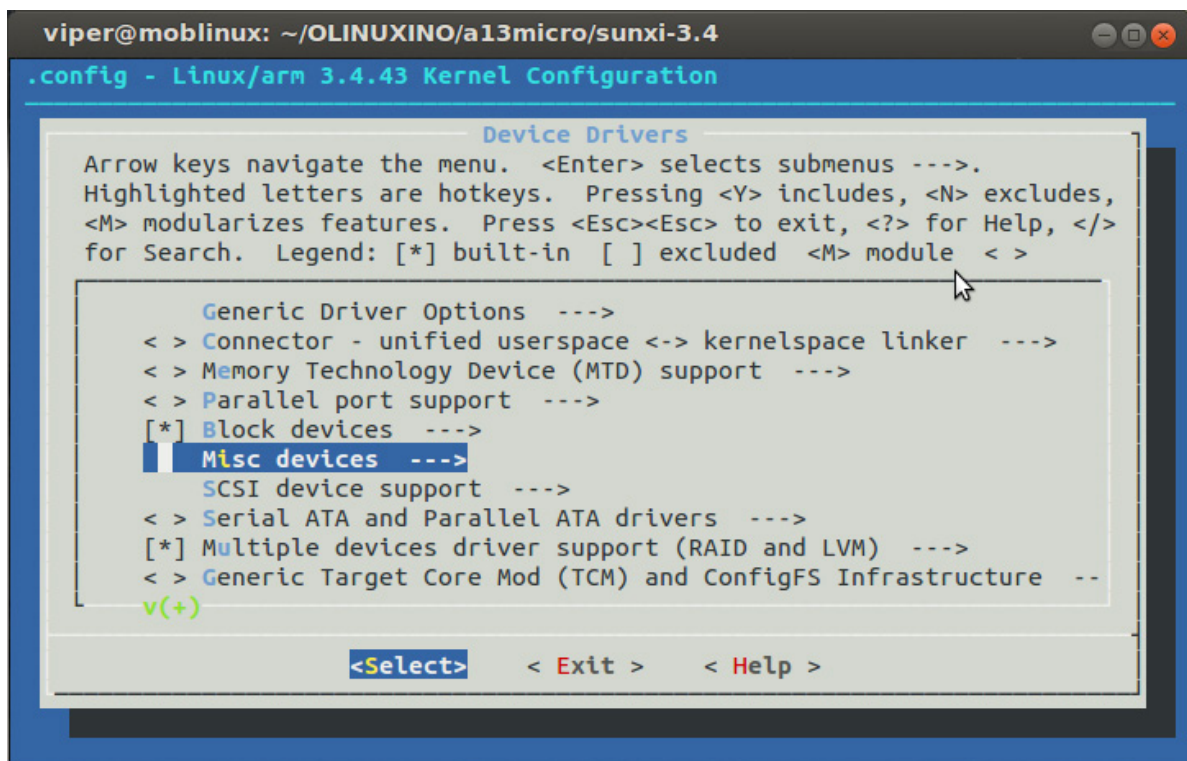


Figure 3. Selecting Misc Devices under Device Drivers

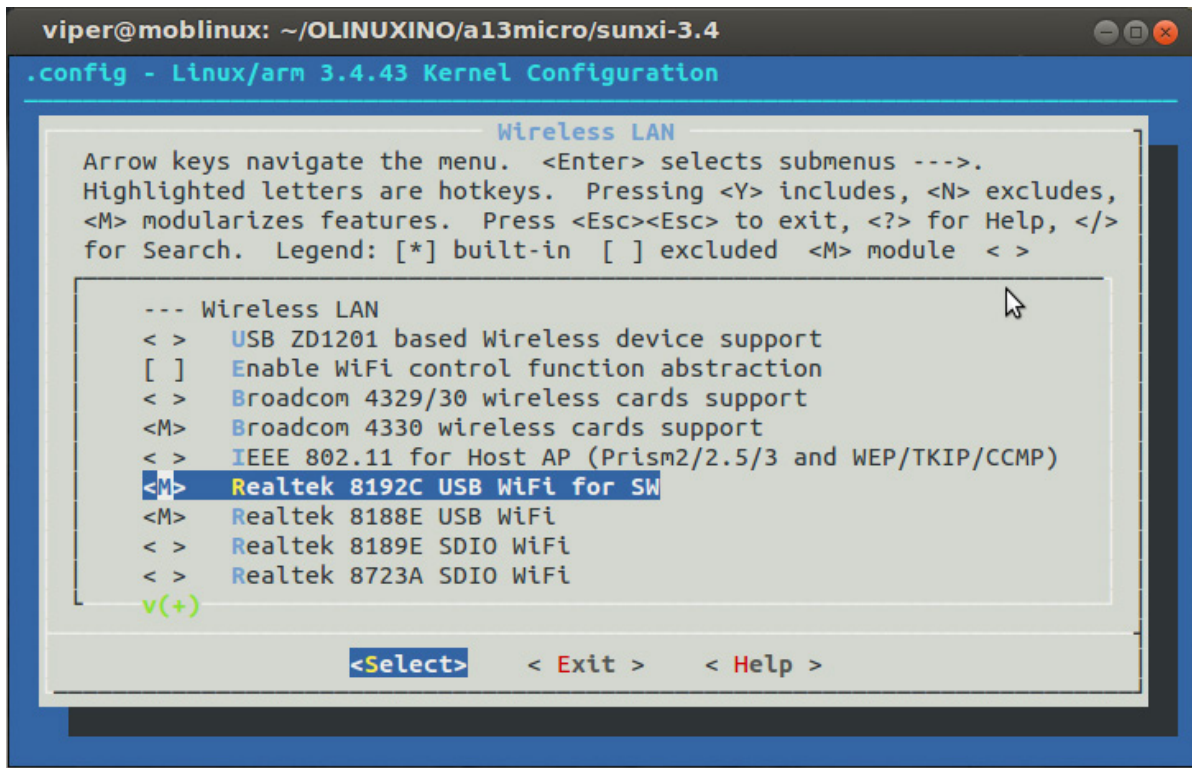


Figure 4. Realtek 8192C Driver as a Module

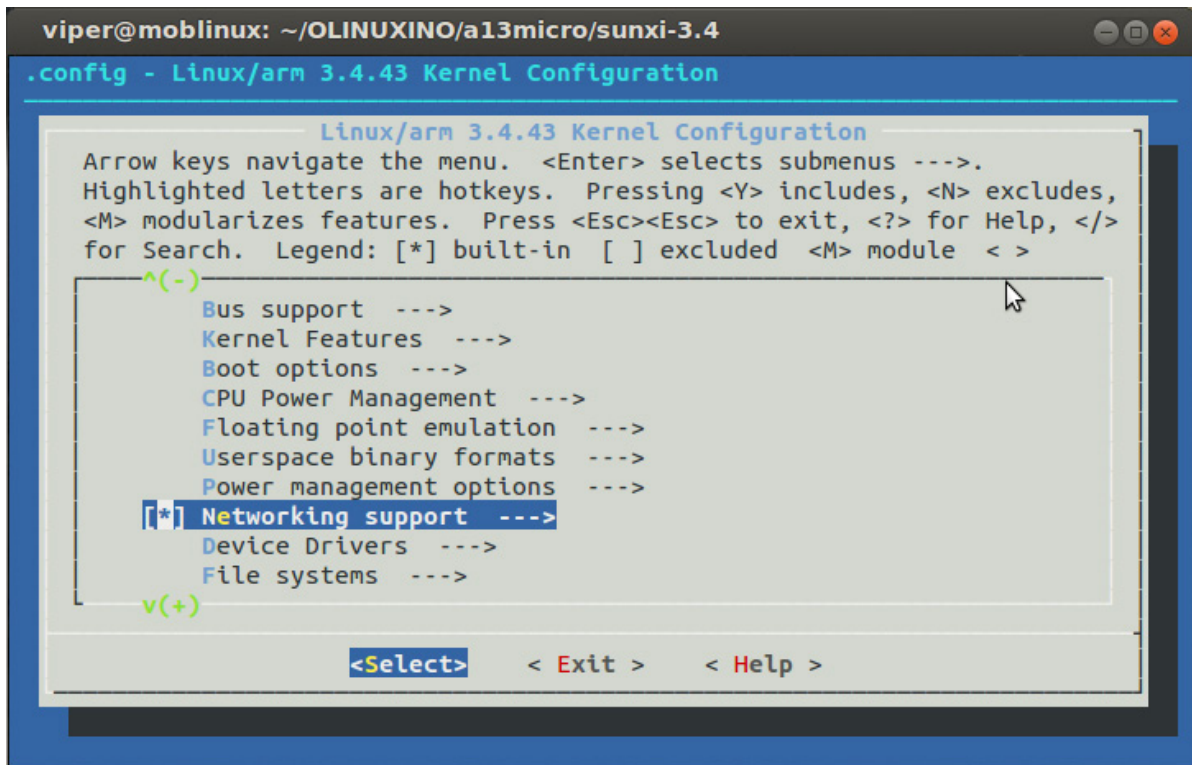


Figure 5. Networking Support Option

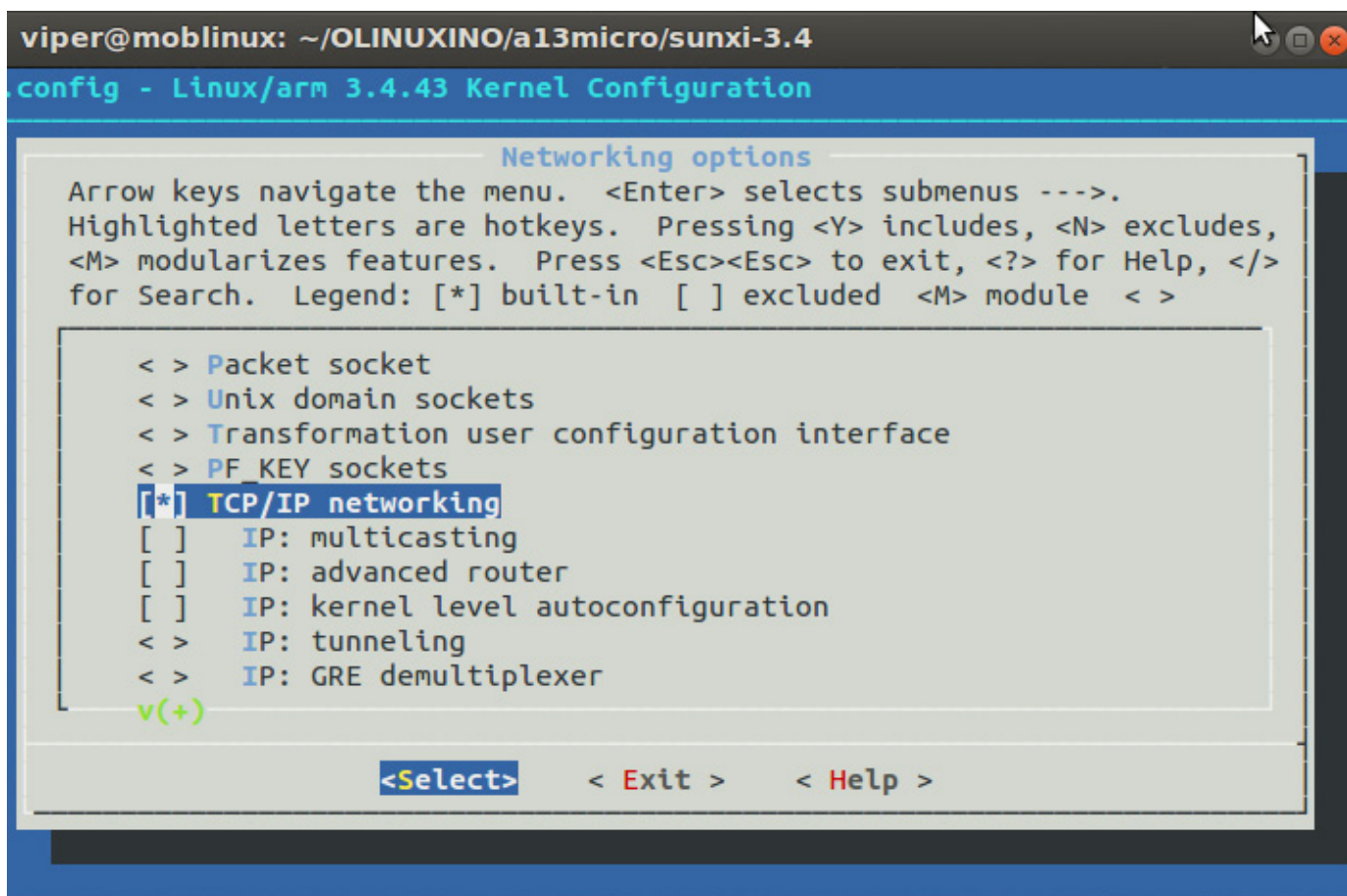


Figure 6. Make sure that TCP/IP networking is selected.

I also plan to use Olimex's MOD-WIFI USB Stick Module² that adds the Wi-Fi capabilities to the board. To do that, you need to include the driver for the module, which is called "Realtek 8192C USB Wifi for SW" and is located under "Device Drivers/Network device support/Wireless LAN". You are welcome to experiment with other Wi-Fi devices. I can vouch that I successfully run the A13 board with the Netgear WG111v2 USB Wifi

Stick module. The driver I used for this Wi-Fi device was "Realtek 8187 and 8187B USB support".

You also have to make sure that TCP/IP is selected and included in the kernel. Tick Networking Support and press Enter to select it.

Go inside Networking options and make sure that TCP/IP networking is selected.

Once you are satisfied with your configuration, save it and go back to the command prompt. Issue the

following commands to compile the kernel and build the drivers:

```
$ make ARCH=arm \
  CROSS_COMPILE=arm-linux-gnueabihf- uImage
$ make ARCH=arm \
  CROSS_COMPILE=arm-linux-gnueabihf- \
  INSTALL_MOD_PATH=out modules
$ make ARCH=arm \
  CROSS_COMPILE=arm-linux-gnueabihf- \
  INSTALL_MOD_PATH=out modules_install
```

When the compilation finishes, you will end up with the kernel image in `$KERNEL_DIR/arch/arm/boot/ulmage` and the modules and drivers in `$KERNEL_DIR/out/lib/modules/$KERNEL_VERSION`.

The next step is to prepare a minimal filesystem for your board. The easiest option I've found so far is by using the root filesystem from the Debian project or Ubuntu, as both distributions provide armhf binaries for the essential applications. I explain how to prepare both options next.

Preparing the Filesystem: Debian Wheezy

Start by creating a new directory for your root filesystem. For the sake of clarity, I call mine `debian-rootfs`. You'll use an application called `debootstrap` to

pull the basic filesystem structure from a Debian repository. You are free to use a repository that is closer to you, rather than the same one I use in this example. Enter the following as root or using `sudo`, inside your newly created directory:

```
# debootstrap --foreign --arch armhf wheezy \
  /home/user/A13System/debian-rootfs \
  http://ftp.debian.org/debian
```

Note that the resulting structure is still not a complete filesystem. The next step is to create a chroot system within your new directory. For those of you who are not familiar with chroot, this command effectively creates an isolated system within your "host" system:

```
# cp $(which qemu-arm-static) \
  /home/user/A13System/debian-rootfs/usr/bin
# mount -t proc proc \
  /home/user/A13System/debian-rootfs/proc
# chroot /home/user/debian-rootfs /bin/bash
```

```
I have no name!# ./debootstrap/debootstrap \
  --second-stage
```

Copy the `qemu-arm-static` binary into your root filesystem's `/usr/bin` directory. The `qemu-arm-static` binary helps run the armhf binaries from your x86/64-bit systems. You

also need to mount the host's proc filesystem into your chroot system. When you first get inside the chroot system, you might find a strange prompt greeting you ("I have no name!"). This is not a cause for concern, and you can safely disregard it. Once you are inside your chroot system, execute another call to `debootstrap` to complete the base system (with `--second-stage`).

If you are curious whether you really are running an armhf system within your chroot system, issue the `uname` command to check. If you see something like "armv7l" somewhere in the output, it is an indication that your chroot is running the armhf system.

The next step is to update your apt source list file. Within your chroot system, or using the build host's editor, go and edit the file `/etc/apt/sources.list` that resides inside your root filesystem directory. Add the following lines to this file (remember, you can use other Debian repositories as well):

```
deb http://ftp.debian.org/debian wheezy main \
  contrib non-free
deb-src http://ftp.debian.org/debian wheezy main \
  contrib non-free
```

```
deb http://ftp.debian.org/debian wheezy-updates \
  main contrib non-free
deb-src http://ftp.debian.org/debian \
  wheezy-updates main contrib non-free

deb http://security.debian.org/ wheezy/updates \
  main contrib non-free
deb-src http://security.debian.org/ \
  wheezy/updates main contrib non-free
```

Preparing the Filesystem: Ubuntu

If you are feeling adventurous, you always can try to debootstrap your Ubuntu root filesystem, just like I described in the previous section. (You also can find instructions on the Internet for that.) Here, let's opt for an easier way and just download a ready-made minimal root filesystem provided by Ubuntu. Several packages are available, including Ubuntu 12.04, 12.10 and 13.04. Make sure that you are downloading the armhf version of the root filesystem package.

Create a directory for your Ubuntu root filesystem and extract the contents of the file you just downloaded into it. Your next move should be to edit `/etc/resolv.conf` and add your nameserver in there. Also, take a look at your `sources.list` file in `/etc/apt/`. You might want to add `universe` and `multiverse` at the end of each `deb` and `deb-src` line.

You should check the version number of your `qemu-arm-static`. Version 1.0.50 that comes with standard install of Ubuntu 12.04 generates errors when running the following steps on my build system for the Ubuntu root filesystem. To solve the problem, I had to compile my own `qemu-arm-static`. I used version 1.0.91 (see Resources for the download URL of the source package). Do the following steps to configure and compile the binary, and copy the resulting `qemu-arm` to `qemu-arm-static` inside your Ubuntu root filesystem's `/usr/bin` directory:

```
$ ./configure \
--prefix=/home/user/A13System/qemu-arm-static \
--static --disable-kvm \
--target-list=arm-linux-user

$ make
$ make install
```

Finishing Touches for the Root Filesystem

What you have so far is just a very basic filesystem. Now let's improve it so that you have the tools required for a comfortable basic computing environment. Change the locales generated according to your own locale. All the processes described next are done inside the

`chroot` system:

```
root@host:/# apt-get update
root@host:/# apt-get install apt-utils ncurses-dev
root@host:/# apt-get install dialog locales tzdata
root@host:/# locale-gen en_AU en_AU.UTF-8
root@host:/# dpkg-reconfigure locales
root@host:/# dpkg-reconfigure tzdata
root@host:/# apt-get install iputils-ping \
wpasupplicant dhcpcd5 sudo openssh-server ntp \
openssh-client
root@host:/# apt-get install nano vim gettext \
bison automake autoconf
root@host:/# apt-get install python rsyslog \
network-manager alsa-utils
```

Now let's configure Wi-Fi connectivity. I'm assuming that you're using a Wi-Fi USB adapter for your connectivity and that your wireless network connection configuration is using WPA for security. Change the steps accordingly for your configuration. Edit your `/etc/network/interfaces` and add the following lines, changing the values as needed:

```
auto wlan0
iface wlan0 inet dhcp
wpa-ssid YOUR_ESSID
wpa-psk YOUR_PASSPHRASE
```

Next, if you want your bash shell to have autocompletion, edit `/etc/bash.bashrc` and uncomment some

of the lines to be something like the following:

```
# Commented out, don't overwrite xterm -T
# "title" -n "icontitle" by default.
# If this is an xterm set the title to
# user@host:dir
case "$TERM" in
xterm*|rxvt*)
    PROMPT_COMMAND='echo -ne \
"\033]0;${USER}@${HOSTNAME}: ${PWD}\007"'
    ;;
*)
    ;;
esac

# enable bash completion in interactive shells
if [ -f /etc/bash_completion ] && ! shopt -oq \
posix; then
    . /etc/bash_completion
fi
```

Check `/etc/shadow` for the following:

```
root:*:15629:0:99999:7:::
daemon:*:15629:0:99999:7:::
```

If you see an asterisk (*) after the first colon on the line for root, you should remove it. This will allow you to set the root password yourself on the first run.

You have completed the process of building the root filesystem for your board. Next, let's compress the

entire root filesystem so you can deploy it easily to your MicroSD card later. Exit the chroot environment and do the following inside your root filesystem directory:

```
# umount proc
# rm ./usr/bin/qemu-arm-static
# tar -zcvf /home/user/my-rootfs.tar.gz *
```

Preparing the MicroSD Card

I am using a 4GB MicroSD card for my board. I am sure that a 2GB MicroSD card would be sufficient to contain all your files, but it is nice to have some room for additional applications. You need to create two partitions on your empty MicroSD card. The first one is a VFAT partition of around 17MB for U-Boot and the kernel image. The rest will be used to store your root filesystem.

Mount the MicroSD card. Take note of the device name your computer gives the MicroSD card. Some computers recognize the card as `/dev/sdX`, while others call it `/dev/mmcblkX` (for this example, I assuming that your card is recognized as `/dev/sdb`):

```
# fdisk -u=sectors /dev/sdb
```

Type "p" to list the partitions inside the card. If you have any

```
viper@moblinx: ~
viper@moblinx:~$ sudo fdisk /dev/sdb

Command (m for help): p

Disk /dev/sdb: 3965 MB, 3965190144 bytes
122 heads, 62 sectors/track, 1023 cylinders, total 7744512 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x000b7165

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1             2048         34815       16384    83  Linux
/dev/sdb2            34816        7744511      3854848    83  Linux

Command (m for help): █
```

Figure 7. List of Partitions on a 4GB MicroSD Card

partitions at all listed, delete them by pressing "d". Once the card is empty, create a new partition by pressing "n". Make this the first primary partition. fdisk is going to ask you for starting and ending sector numbers. Type "2048" and "34815", respectively. Repeat the process for the second partition. This time, just press Enter when asked for starting and ending sector numbers; fdisk will use the default

values, which will fill the remainder of the card.

Type "p" again to list the partitions. You should see something like what is shown in Figure 7.

Type "w" to write the changes permanently onto the card. Now, create the two filesystem types on the partitions:

```
# mkfs.vfat /dev/sdb1
# mkfs.ext3 /dev/sdb2
```

Don't forget to do the sync after every command you type for the MicroSD card. Sync ensures that the changes are flushed and keeps the card in the correct state. Next, mount the partitions. I am assuming the mountpoints are /media/card1 for /dev/sdb1 and /media/card2 for /dev/sdb2. First, populate the root filesystem and copy the kernel modules onto the first partition:

```
# cd /media/card2
# tar -xzvf /home/user/A13System/my-rootfs.tar.gz
# sync
... [THIS WILL TAKE SOME TIME]
# cp -a $KERNEL_DIR/out/lib/modules/3.4.43+/ \
./lib/modules/
# sync
```

Copy the ulmage file from your kernel directory (in arch/arm/boot) to the first partition, along with a file called script.bin. script.bin stores the system configuration settings necessary for Allwinner chips. If you want to edit these settings, convert this .bin file into a .fex file using a tool called bin2fex. You can edit the resulting file with any text editor.

For the last step, you need to write U-Boot onto the card itself. Pay extra attention to what you

type here, as you are not going to write to /dev/sdb1 or /dev/sdb2 but to /dev/sdb:

```
# cd /home/user/A13System/u-boot
# dd if=spl/sunxi-spl.bin of=/dev/sdb bs=1024 \
seek=8
# dd if=u-boot.bin of=/dev/sdb bs=1024 seek=32
# sync
```

Now your MicroSD card is ready to use.

First Run

Plug in the card in the slot on the board. Also plug in the Wi-Fi USB Stick, a keyboard and the VGA monitor (use a USB hub if you have to). Plug in the power cord and wait for the login prompt.

Log in with the root account. You shouldn't need a password for the first run. After you get in, set a secure password for your root account and create another account for your daily use. Put this new user into the sudoers file. Check whether you have network connectivity. Test the board remotely by connecting to it via SSH. If you can do all that successfully, congratulations! You have a great minimalist PC/server at your disposal.

IF YOU ARE INTERESTED IN USING THIS BOARD WITH A GRAPHICAL USER INTERFACE, YOU NEED TO USE A LIGHTWEIGHT GUI ENVIRONMENT, BECAUSE THE BOARD DOES NOT HAVE MUCH RAM TO SPARE.

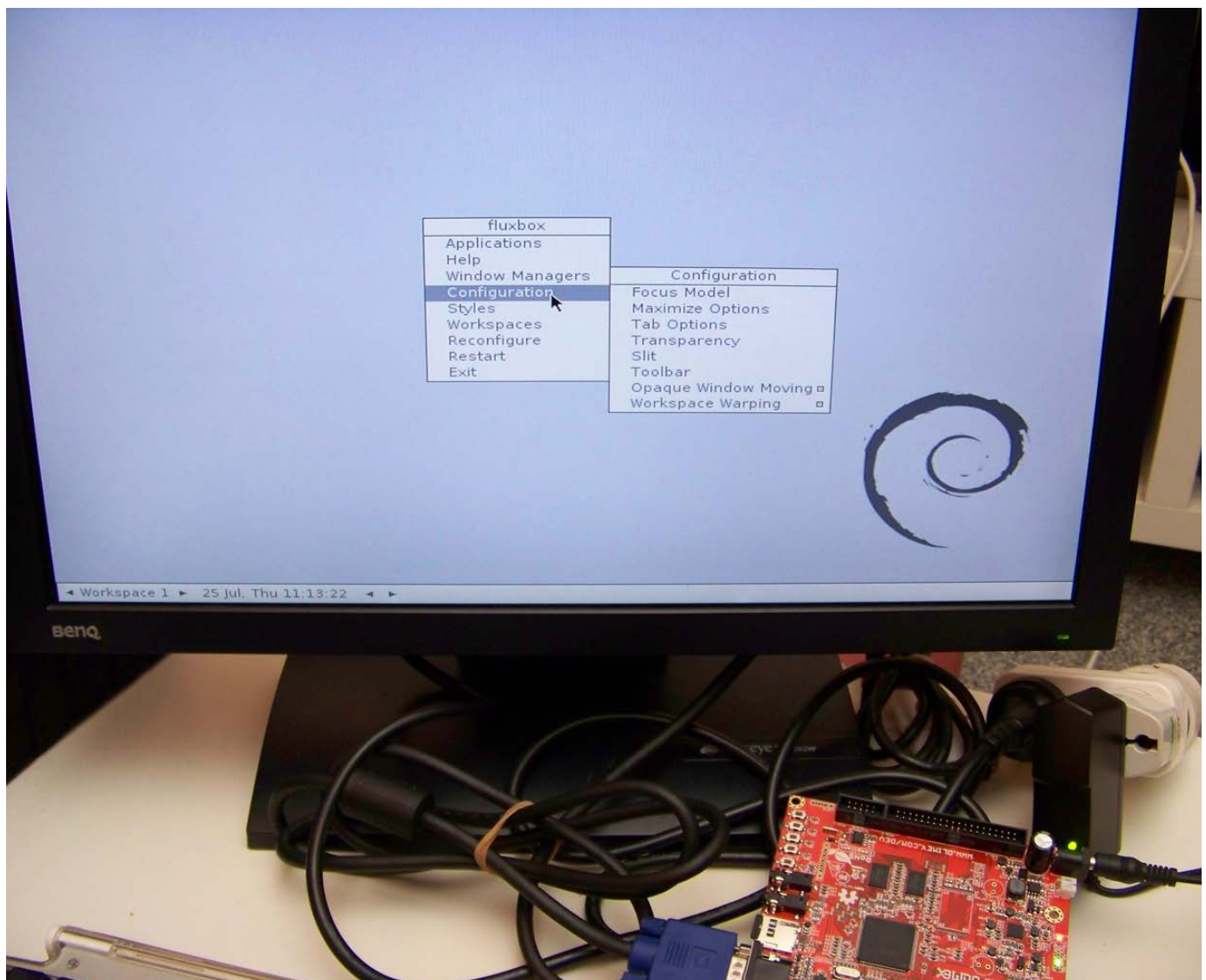


Figure 8. A13Micro Running Fluxbox

A HANDY U-BOOT TRICK

Deploy and test new builds quickly with no more than rebooting the board.

BHARATH BHUSHAN LOHRAY

Embedded developers working on kernels or bare-metal programs often go through several development cycles. Each time the developer modifies the code, the code has to be compiled, the ELF (Executable and Linkable Format)/kernel image has to be copied onto the SD card, and the card then has to be transferred from the PC to the development board and rebooted. In my experience as a developer, I found the last two steps to be a major bottleneck. Even copying files to the fastest SD cards is slower than copying files between hard drives and sometimes between computers across the network.

Moreover, by frequently inserting and removing the SD card from the slot, one incurs the risk of damaging the fragile connectors on the development boards. Believe me! I lost a BeagleBoard by accidentally applying too much force while holding the board and pulling out the SD card. The pressure caused the I²C bus to fail. Because the power management chip was controlled by I²C, nothing other than the serial terminal worked after that. Setting aside the cost of the board, a board failure at a critical time during a project is catastrophic if you do not have a backup board.

After losing the BeagleBoard, I hit upon the idea to load my bare-metal code over the LAN via bootp and TFTP and leave the board untouched. This not only reduced the risk of mechanically damaging my board, but it also improved on my turn-around times. I no longer needed to copy files to the SD card and move it around.

In this article, I present a brief introduction to U-Boot and then describe the necessary configurations to set up a development environment using DHCP and TFTP. The setup I present here will let you deploy and test new builds quickly with no more than rebooting the board. I use the BeagleBone Black (<http://beagleboard.org/Products/BeagleBone%20Black>) as the target platform and Ubuntu as the development platform for my examples in this article. You may, however, use the methods presented here to work with any board that uses U-Boot or Barebox as its stage-2 bootloader.

U-Boot

U-Boot is a popular bootloader used by many development platforms. It supports multiple architectures including ARM, MIPS, AVR32, Nios, Microblaze, 68K and x86. U-Boot

U-Boot is a pretty advanced bootloader that is capable of loading the kernel and ramdisk image from the NAND, SD card, USB drive and even the Ethernet via bootp, DHCP and TFTP.

Listing 1. The Serial Console Output from the Stage-1 Bootloader

```
U-Boot SPL 2013.04-rc1-14237-g90639fe-dirty (Apr 13 2013 - 13:57:11)
musb-hdrc: ConfigData=0xde (UTMI-8, dyn FIFOs, HB-ISO Rx,
  ↳HB-ISO Tx, SoftConn)
musb-hdrc: MHDRC RTL version 2.0
musb-hdrc: setup fifo_mode 4
musb-hdrc: 28/31 max ep, 16384/16384 memory
USB Peripheral mode controller at 47401000 using PIO, IRQ 0
musb-hdrc: ConfigData=0xde (UTMI-8, dyn FIFOs, HB-ISO Rx,
  ↳HB-ISO Tx, SoftConn)
musb-hdrc: MHDRC RTL version 2.0
musb-hdrc: setup fifo_mode 4
musb-hdrc: 28/31 max ep, 16384/16384 memory
USB Host mode controller at 47401800 using PIO, IRQ 0
OMAP SD/MMC: 0
mmc_send_cmd : timeout: No status update
reading u-boot.img
reading u-boot.img
```

has support for several filesystems as well, including FAT32, ext2, ext3, ext4 and Cramfs built in to it. It also has a shell where it interactively can take input from users, and it supports scripting. It is distributed under the GPLv2 license. U-Boot is a stage-2 bootloader.

The U-Boot project also includes the x-loader. The x-loader is a small stage-1 bootloader for ARM. Most modern chips have the ability to read a FAT32 filesystem built in to the ROM. The x-loader loads the U-Boot into memory and transfers control to it. U-Boot is a pretty advanced

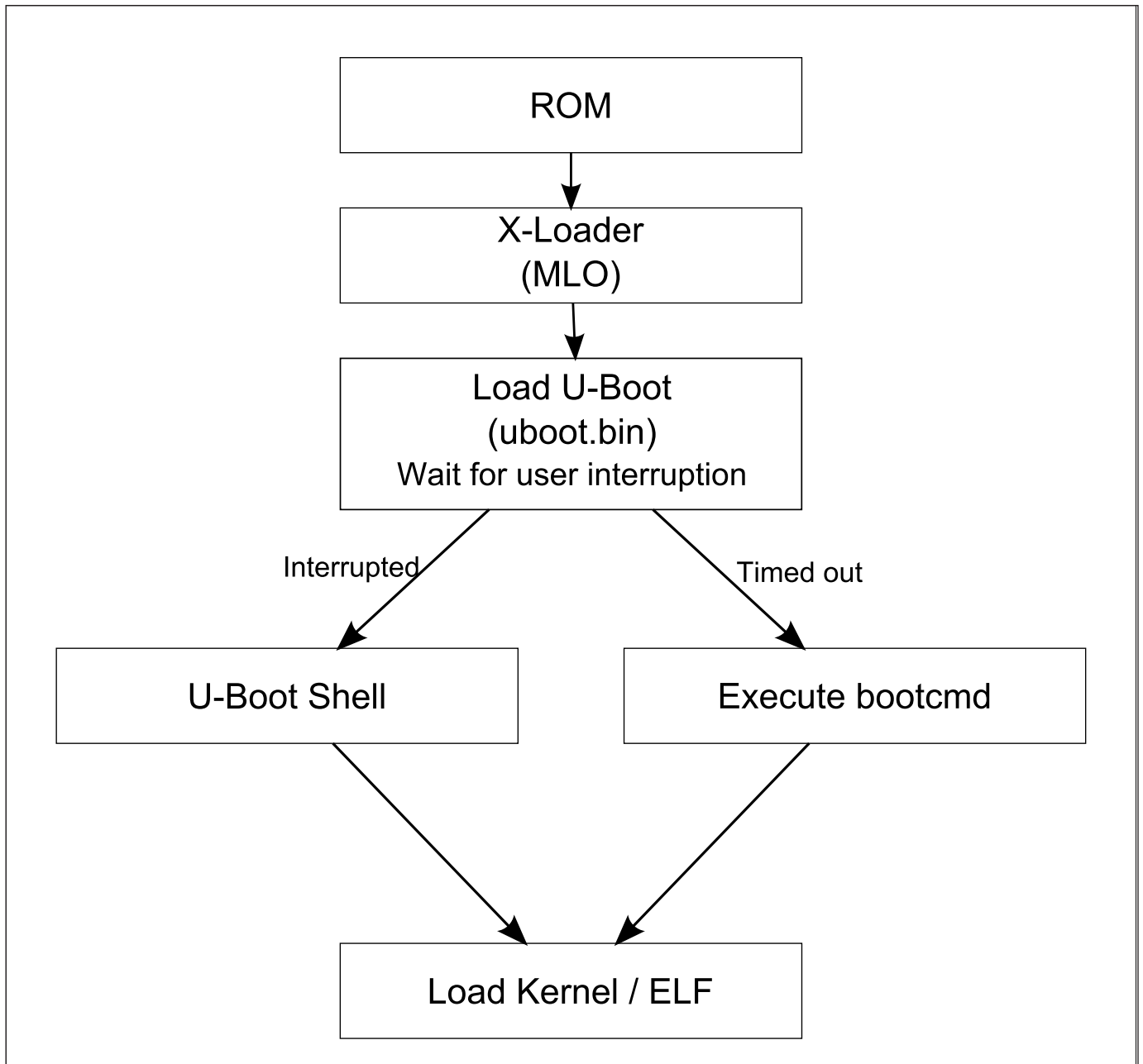


Figure 1. Boot Sequence

bootloader that is capable of loading the kernel and ramdisk image from the NAND, SD card, USB drive and even the Ethernet via bootp, DHCP and TFTP.

Figure 1 shows the default boot sequence of the BeagleBone Black. This sequence is more or less applicable to most embedded systems. The x-loader and U-Boot executables

Listing 2. The Serial Console Output from the Stage-2 Bootloader

```
U-Boot 2013.04-rc1-14237-g90639fe-dirty (Apr 13 2013 - 13:57:11)
```

```
I2C:   ready
```

```
DRAM:  512 MiB
```

```
WARNING: Caches not enabled
```

```
NAND:  No NAND device found!!!
```

```
0 MiB
```

```
MMC:   OMAP SD/MMC: 0, OMAP SD/MMC: 1
```

```
*** Warning - readenv() failed, using default environment
```

```
musb-hdrc: ConfigData=0xde (UTMI-8, dyn FIFOs, HB-ISO Rx,  
↳HB-ISO Tx, SoftConn)
```

```
musb-hdrc: MHDRC RTL version 2.0
```

```
musb-hdrc: setup fifo_mode 4
```

```
musb-hdrc: 28/31 max ep, 16384/16384 memory
```

```
USB Peripheral mode controller at 47401000 using PIO, IRQ 0
```

```
musb-hdrc: ConfigData=0xde (UTMI-8, dyn FIFOs, HB-ISO Rx,  
↳HB-ISO Tx, SoftConn)
```

```
musb-hdrc: MHDRC RTL version 2.0
```

```
musb-hdrc: setup fifo_mode 4
```

```
musb-hdrc: 28/31 max ep, 16384/16384 memory
```

```
USB Host mode controller at 47401800 using PIO, IRQ 0
```

```
Net:   <ethaddr> not set. Validating first E-fuse MAC
```

```
cpsw, usb_ether
```

```
Hit any key to stop autoboot:  0
```

are stored in the files called MLO and uboot.img, respectively. These files are stored in a FAT32 partition. The serial port outputs of the BeagleBone are

shown in Listings 1–3. The x-loader is responsible for the output shown in Listing 1. Once the execution is handed over to U-Boot, it offers you

Listing 3. The Serial Console Output from the Stage-2 Bootloader and Kernel

```

gpio: pin 53 (gpio 53) value is 1
Card did not respond to voltage select!
.
.
.
gpio: pin 54 (gpio 54) value is 1
SD/MMC found on device 1
reading uEnv.txt
58 bytes read in 4 ms (13.7 KiB/s)
Loaded environment from uEnv.txt
Importing environment from mmc ...
Running uenvcmd ...
Booting the bone from emmc...
gpio: pin 55 (gpio 55) value is 1
4215264 bytes read in 778 ms (5.2 MiB/s)
gpio: pin 56 (gpio 56) value is 1
22780 bytes read in 40 ms (555.7 KiB/s)
Booting from mmc ...
## Booting kernel from Legacy Image at 80007fc0 ...
Image Name:   Angstrom/3.8.6/beaglebone
Image Type:   ARM Linux Kernel Image (uncompressed)
Data Size:    4215200 Bytes = 4 MiB
Load Address: 80008000
Entry Point:  80008000
Verifying Checksum ... OK
## Flattened Device Tree blob at 80f80000

Booting using the fdt blob at 0x80f80000
XIP Kernel Image ... OK
OK
Using Device Tree in place at 80f80000, end 80f888fb

Starting kernel ...

Uncompressing Linux... done, booting the kernel.
[ 0.106033] pinctrl-single 44e10800.pinmux: prop pinctrl-0
index 0 invalid phandle
.
.
.
[ 9.638448] net eth0: phy 4a101000.mdio:01 not found on slave 1

.---0---.
|      |           .-.      o o
|  |  |-----|-----|-----|-----|-----|-----|-----|-----|
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|-----|-----|-----|-----|-----|-----|-----|-----|
      -' |
      ----'

The Angstrom Distribution beaglebone tty0

Angstrom v2012.12 - Kernel 3.8.6

beaglebone login:

```

a few seconds to interrupt the boot sequence, as shown in Listing 2. If you choose not to interrupt, U-Boot executes an environment variable

called `bootcmd`. `bootcmd` holds the search sequence for a file called `uImage`. This is the kernel image. The kernel image is loaded into the

Listing 4. Well Formatted Content of the Variable bootcmd

```
01 gpio set 53;
02 i2c mw 0x24 1 0x3e;
03 run findfdt;
04 mmc dev 0;
05 if mmc rescan ;
06 then
07     echo micro SD card found;
08     setenv mmcdev 0;
09 else
10     echo No micro SD card found, setting mmcdev to 1;
11     setenv mmcdev 1;
12 fi;
13 setenv bootpart ${mmcdev}:2;
14 mmc dev ${mmcdev};
15 if mmc rescan;
16 then
17     gpio set 54;
18     echo SD/MMC found on device ${mmcdev};
19     if run loadbootenv;
20     then
21         echo Loaded environment from ${bootenv};
22         run importbootenv;
23     fi;
24     if test -n $uenvcmd;
25     then
26         echo Running uenvcmd ...;
27         run uenvcmd;
28     fi;
29     gpio set 55;
30     if run loaduimage;
31     then
32         gpio set 56;
33         run loadfdt;
34         run mmcboot;
35     fi;
36 fi;
```


memory, and the execution finally is transferred to the kernel, as shown in Listing 3.

The search sequence defined in the `bootcmd` variable and the filename (`ulmage`) are hard-coded in the U-Boot source code (see Listing 9). Listing 4 shows the formatted content of the environment variable `bootcmd`. The interesting parts of `bootcmd` are

U-Boot shell. At the shell, you can see a list of supported commands by typing `help` or `?`. You can list all defined environment variables with the `env print` command. These environment variables are a powerful tool for scripting. To resume the boot sequence, you either can issue the `boot` command or `run bootcmd`. A good way to understand what the

The `uEnv.txt` file is a method for users to insert scripts into the environment.

lines 19–28. This part of the script checks for the existence of a file called `uEnv.txt`. If the file is found, the file is loaded into the memory (line 19). Then, it is imported to the environment ready to be read or executed (line 22). After this, the script checks to see if the variable `uenvcmd` is defined (line 24). If it is defined, the script in the variable is executed. The `uEnv.txt` file is a method for users to insert scripts into the environment. Here, we'll use this to override the default search sequence and load the kernel image or an ELF file from the TFTP server.

For better insight into the workings of U-Boot, I recommend interrupting the execution and dropping to the

`bootcmd` is doing is to execute each command one at a time from the U-Boot shell and see its effect. You may replace the `if...then...else...fi` blocks by executing the conditional statement without the `if` part and checking its output by typing `echo $?`.

DHCP

The DHCP (Dynamic Host Configuration Protocol) is a protocol to provide hosts with the necessary information to access the network on demand. This includes the IP address for the host, the DNS servers, the gateway server, the time servers, the TFTP server and so on. The DHCP server also can provide the name of the file containing the kernel

image that the host must get from the TFTP server to continue booting. The DHCP server can be set up to provide a configuration either for the entire network or on a per-host basis. Configuring the filename (Listing 5) for the entire network is not a good idea, as one kernel image or ELF file will execute only on the architecture for which it was built. For instance, the vmlinuz image built for an x86_64 will not work on a system with an ARM-based processor.

The Ubuntu apt repository offers two DHCP servers: `isc-dhcp-server` and `dhcpcd`. I prefer to use `isc-dhcp-server`.

The `isc-dhcpd-server` from the Ubuntu repository is pretty advanced and implements all the necessary features. I recommend using Webmin to configure it. Webmin is a Web-based configuration tool that supports configuring several Linux-based services and daemons. I

Listing 5. The Host Configuration Section for a DHCP Server

```
subnet 192.168.0.0 netmask 255.255.0.0 {
    next-server 192.168.146.1;
    option domain-name-servers 192.168.146.1;
    option routers 192.168.146.1;
    range 192.168.145.1 192.168.145.254;

    # The BeagleBone Black 1
    host BBB-1 {
        next-server 192.168.146.1;
        filename "/BI/uImage";
        hardware ethernet C8:A0:30:B0:88:EB;
        fixed-address 192.168.146.4;
    }
}
```

recommend installing Webmin from the apt repository. See the Webmin documentation for instructions for adding the Webmin apt repository to Ubuntu (<http://www.webmin.com/deb.html>).

Once you have your DHCP server

IMPORTANT NOTE:

Be extremely careful while using the DHCP server. A network must not have more than a single DHCP server. A second DHCP server will cause serious problems on the network. Other users will lose network access. If you are on a corporate or a university network, you will generate a high-priority incident inviting the IT department to come looking for you.

installed, you need to configure a subnet and select a pool of IP addresses to be dished out to hosts on the network on request. After this, add the lines corresponding to the host from Listing 5 into your `/etc/dhcp/dhcpd.conf` file, or do the equivalent from Webmin's intuitive interface. In Listing 5, `C8:A0:30:B0:88:EB` corresponds to the BeagleBone's Ethernet address. The `next-server` is the address of the TFTP server from which to fetch the kernel image of ELF. The `/BI/uImage` filename is the name of the kernel image. Rename the image to whatever you use.

TFTP

TFTP (Trivial File Transfer Protocol) is a lightweight file-transfer protocol. It does not support authentication methods. Anyone can connect and download any file by name from the server or upload any file to the server. You can, however, protect your server to some extent by setting firewall rules to deny IP addresses out of a particular range. You also can make the TFTP home directory read-only to the world. This should prevent any malicious uploads to the server. The Ubuntu apt repository has two different TFTP servers: `atftp` and

`tftp-hpa`. I recommend `tftp-hpa`, as development of `atftp` has seized since 2004.

`tftpd-hpa` is more or less ready to run just after installation. The default file store is usually `/var/lib/tftpboot/`, and the configuration files for `tftp-can` may be found in `/etc/default/tftpd-hpa`. You can change the location of the default file store to any other location of your choice by changing the `TFTP_DIRECTORY` option. The TFTP installation creates a user and a group called `tftp`. The `tftp` server runs as this user. I recommend adding yourself to the `tftp` group and changing permissions on the `tftp` data directory to `775`. This will let you read and write to the `tftp` data directory without switching to root each time. Moreover, if files in the `tftp` data directory are owned by root, the `tftp` server will not be able to read and serve them over the network. You can test your server by placing a file there and attempting to get it using the `tftp` client:

```
$ tftp 192.168.146.1 -c get uImage[COMMAND]
```

Some common problems you may face include errors due to permission. Make sure that the files are readable by the `tftp` user or whichever user the

Listing 6. An Example of the uenvcmd Variable for DHCP Booting

```

echo Booting the BeagleBone Black from LAN (DHCP)...
dhcp ${kloadaddr}
tftpboot ${fdtaddr} /BI/${fdtfile}
setenv bootargs console=${console} ${optargs} root=${mmcroot}
↳rootfstype=${mmcrootfstype} optargs=quiet
bootm ${kloadaddr} - ${fdtaddr}

```

Listing 7. An Example of uenvcmd Variable for TFTP Booting

```

echo Booting the BeagleBone Black from LAN (TFTP)...
env set ipaddr 192.168.146.10
env set serverip 192.168.146.1
tftpboot ${kloadaddr} /BI/${bootfile}
tftpboot ${fdtaddr} /BI/${fdtfile}
setenv bootargs console=${console} ${optargs} root=${mmcroot}
↳rootfstype=${mmcrootfstype} optargs=quiet
bootm ${kloadaddr} - ${fdtaddr}

```

tftpd runs as. Additionally, directories must have execute permission, or tftp will not be able to descend and read the content of that directory, and you'll see a "Permission denied" error when you attempt to get the file.

U-Boot Scripting

Now that you have your DHCP and TFTP servers working, let's write a U-Boot script that will fetch the kernel image and boot it. I'm going to present two ways of doing this: using DHCP and using only TFTP. As I

mentioned before, running a poorly configured DHCP server will cause a network-wide disruption of services. However, if you know what you are doing and have prior experience with setting up network services, this is the simplest way to boot the board.

A DHCP boot can be initiated simply by adding or modifying the uenvcmd variable in the uEnv.txt file, as shown in Listing 6. uEnv.txt is found in the FAT32 partition of the BeagleBone Black. This partition is available to be mounted when the

BeagleBone Black is connected to your computer via USB cable.

For a TFTP-only boot, you manually specify an IP address for the development board and the TFTP server. This is a much safer process, and you incur very little risk of interfering with other users on the network. As in the case of configuring to boot with DHCP, you must modify the `uenvcmd` variable in the `uEnv.txt` file. The script shown in Listing 7 is an example of how to set up your BeagleBone Black to get a kernel image from the TFTP server and pass on the execution to it.

Both Listing 6 and 7 are formatted to give a clear understanding of the process. The actual `uEnv.txt` file should look something like the script shown in Listing 8. For more information about U-Boot scripting, refer to the U-Boot FAQ (<http://www.denx.de/wiki/DULG/Faq>) and U-Boot Manual

(<http://www.denx.de/wiki/DULG/Manual>). The various commands in the `uenvcmd` variable must be on the same line separated by a semicolon. You may notice that I place my script in `uenvcmdx` instead of `uenvcmd`. This is because `test -n` throws an error to the console based on the content of the variable it is testing. Certain variable contents, especially long complicated scripts, cause the `test -n` to fail with an error message to the console. Therefore, I put a simple command to run `uenvcmdx` in `uenvcmd`. If you find that your script from the `uEnv.txt` is not being executed, look for an error on the serial console like this:

```
test - minimal test like /bin/sh

Usage:
test [args..]
```

Listing 8. An Example of `uEnv.txt` for TFTP Booting

```
optargs=quiet
uenvcmdx=echo Booting the bone from emmc...; env set ipaddr
↳192.168.146.10; env set serverip 192.168.146.1; tftpboot
↳${kloadaddr} /BI/${bootfile}; tftpboot ${fdtaddr}
↳/BI/${fdtfile}; setenv bootargs console=${console}
↳${optargs} root=${mmcroot} rootfstype=${mmcrootfstype}
↳optargs=quiet; bootm ${kloadaddr} - ${fdtaddr}
uenvcmd=run uenvcmdx
```

On some development boards like the BeagleBoard xM (<http://beagleboard.org/Products/BeagleBoard-xM>), the Ethernet port is implemented on the USB bus. Therefore, it is necessary to start the USB subsystem before attempting any network-based boot. If your development board does not hold a Flash memory on board, it may not have a MAC address either. In this case, you will have to set a MAC address before you can issue any network requests. You can do that by setting the environment variable

ethaddr along with the rest of the uEnv.txt script.

An alternative but cumbersome way to change the default boot sequence is to modify the U-Boot source code. Modifying the source code gives you greater versatility for booting your development board. When you interrupt the U-Boot boot sequence, drop to the U-Boot shell and issue the `env print` command, you'll see a lot of environment variables that are defined by default. These environment variables are defined as macros in the source

Listing 9. Part of the u-Boot/include/configs/am335x_evm.h File Responsible for the Default Script in the bootcmd Variable

```
#define CONFIG_BOOTCOMMAND \
    "mmc dev ${mmcdev}; if mmc rescan; then " \
    "echo SD/MMC found on device ${mmcdev};" \
    "if run loadbootenv; then " \
    "echo Loaded environment from ${bootenv};" \
    "run importbootenv;" \
    "fi;" \
    "if test -n $uenvcmd; then " \
    "echo Running uenvcmd ...;" \
    "run uenvcmd;" \
    "fi;" \
    "if run loaduimage; then " \
    "run mmcboot;" \
    "fi;" \
    "fi;" \
```


The Personal Cloud

The Personal Cloud gives you access and control over your own data and lets you do more with it than your vendors can. Who wouldn't want that? The question is, how do we get there from here? T.ROB

Personal.

What do you say when someone asks “How much money do you make?”, or “How old are you?”, or “How much do you weigh?”, or “What is your address?” The quintessential response “that’s personal” works because of an underlying assumption that we are within our rights to withhold that information and, more important, that privacy is the default and natural state of affairs. There is no government procedure on reaching the age of majority requiring you to check boxes on an interminably long list, enumerating all of the personal data attributes that you choose to keep private. As a sovereign person, they are all considered private. It is your choice whether to reveal your income, age, weight or any other piece of

personal information. The ubiquity of the phrase “that’s personal” speaks to a broadly shared understanding that we get to choose what information about ourselves we reveal and to whom we reveal it.

Cloud.

There is no industry-standard definition of the term “cloud”, but we should at least all be able to agree on a few attributes. Many of these are optional depending on who you talk to, but the one essential element of the cloud is abstraction. An observer sees the cloud as a single logical thing with which to interact and need not be concerned with the individual elements that make up the cloud.

In addition, a cloud will have one or more of the following elements:

- Elasticity: computing resources can

be allocated dynamically to match variable load.

- Redundancy: the use of multiple physical servers increases the availability of the cloud service.
- Resiliency: the cloud service can survive interruption or loss of some physical components without loss of function.
- Runtime resolution: resource identities and addresses are resolved dynamically.
- Ubiquity: some clouds are public while others are private. However they are scoped, a cloud service is expected to be accessible from all points within that scope.

By incorporating these elements, cloud architecture aims to deliver computing services that are as reliable and available as power or water. The ultimate expression of the cloud is that the computing infrastructure completely disappears except for the user interface.

Personal Cloud.

Imagine a cloud database filled with information about you and a vendor inquiring of that cloud “How much money do you make? How old are

you? How much do you weigh? What is your address?” Whether the cloud is personal or not depends on who gets to decide the answers to these questions. It is not enough that the response back is “that’s personal” if someone else can override your preferences and make the disclosure anyway. It is only a Personal Cloud if you have ultimate control over whether to disclose that information.

A Personal Cloud starts with the assumption that privacy is the default and natural state of affairs and then incorporates that design philosophy into a cloud computing architecture owned and operated by an individual.

Resist the temptation to read more into that statement than there is. For example, there is no requirement that a Personal Cloud must be self-hosted. Today there are commercial services for password management, storage, backups, VoIP, chat and more, all of which fit the definition of Personal Cloud. In all these cases, the custodians of the data have no access to it. Instead, they provide the framework in which the data owner sets policies that express the relationship between the data and any third parties authorized to access it. If it is a cloud architecture, respects the owner’s privacy, and the owner is the ultimate authority over authorization

decisions, it is a Personal Cloud.

Not a Buzzword

Much of the buzz about cloud computing is just that—buzz. But Personal Cloud is more. There is a parallel here with the way we purchase motive power. In San Francisco, halfway down the hill on Mason Street, there are giant motors transferring motive power to cables, which in turn run below the street to distribute that motive power along the cable car routes. When the cable car requires motive power, it gets it directly from the cable. Many years ago, factories worked the same way. Large motors distributed motive power to belts and pulleys that distributed it throughout the factory.

But that method didn't scale very well. There is no tap from your house to an underground cable that provides power for washers and dryers, refrigerators and so on. Nor is there a giant motor in the back yard and a system of pulleys and belts running from it to the house. Instead, we self-host hundreds of tiny motors, invisibly built in to disk drives, DVD players, appliances, clocks and almost everything capable of movement. The magic of this is that we don't think of these things as motors that wash clothes, tell time or spin digital

media. The motors have receded into the background, and we rarely think about them at all, unless they break.

The same thing is happening now with computing. There is more computing capacity in the average modern phone than there was in the Lunar Lander. It's everywhere around you, and more is on the way. All of your devices that have an embedded motor soon will have embedded computing power, if they do not already. Things that currently have no motor or computing power—switches, outlets and bulbs, for example—soon will become smart. More important, all of these sensors, devices and computing platforms are increasingly interconnected and integrated with if-then-else rules engines that correlate events to generate complex behaviors in formerly dumb devices. Sensors, actuators and computing power are quietly but inexorably being woven into the fabric of your life.

Is it reasonable to assume that with all this computing power available to individuals that we will fail to apply it in commercial settings for our own benefit, in much the same way that our vendors have applied it on the supply side? The value proposition of E-Commerce, Supply Chain Management and Customer Relationship Management always

applied to individual users. We never delivered it to that market because of cost, but that barrier continues to fall. These applications radically transformed commerce on a global scale when they were first built out by large enterprises. We have the opportunity to transform commerce again, with even greater impact, by building out business functionality on the consumer side. Or we could continue to focus on making better versions of *Angry Birds* and moving light switches from the wall to the phone.

Today's architecture, in which vendors have all the data and computing power, is a dying relic, left over from a world where computing was specialized and so expensive that only large businesses could afford it. It is the cable car wheelhouse or the big motor out back behind the factory. But in a world of cheap and abundant computing power, individuals are the natural points of integration for their own data. Now we have the choice to continue on with the legacy model or to move to a new model in which the individual's computing capacity improves the experience of both the individual and the vendor. Having all that computing power on the consumer side and not building out new business integration seems to me to be akin to ripping the motors out of

all the disk drives, DVD players, clocks, refrigerators and other appliances, and equipping them with belts and pulleys. You could do it, but it doesn't scale and you won't like the experience.

Impersonal Clouds

Let's revisit that cloud database filled with information about you and a vendor inquiring of that cloud "How much money do you make? How old are you? What is your address?" If that cloud is owned by a credit bureau or data broker, the responses will be the actual values. There is no default assumption of privacy. You have no say in how much data about you these commercial entities hold, whether it is accurate or with whom they share it. The entire business model of data brokers is to provide answers for questions to which, if you had been asked in person, you would have responded by saying "that's personal".

The impersonal cloud also hoards your data, preventing you from deriving any benefit from it. If you participate in a grocery loyalty program, somewhere out there is a database containing line-item records all your purchases. That's a rather intimate level of detail about you and your immediate household. If someone on the street asked you detailed questions about some of

those line items, you undoubtedly would say “that’s personal”. If that same person then offered to pay you a dollar for details of which personal hygiene products you use and how often you buy them, it would seem creepy. But that is exactly the kind of information we give out in exchange for a dollar off of a jug of milk.

I’ve been told that the average person has no use for grocery line-item data. If that is true (and I don’t believe that it is), it’s only because we do not yet have access to that data and, therefore, have not written applications to use it. We don’t know now what we’ll do with that data any more than we could have looked at HTTP in 1983 and predicted Web 2.0. What we do know now though is that all the data is held by vendors, and wherever it is that Personal Cloud can take us, we can’t get there from here. Getting access to our own data is the first step. Personal Cloud is Cloud 2.0.

Strategies for Personal Cloud Development

Twenty years from now, we will know what the Personal Cloud equivalents of Google, Wikipedia, YouTube, Facebook and Instagram are, and they will seem obvious in hindsight. But if you want to get started in Personal Cloud today, where do

you place your bets? Although the specific implementations have yet to be revealed—that’s your job—we do know some general directions with a high degree of confidence. First and foremost among these is that the Personal Cloud wants to be invisible. Once the user sets up the connection and preferences, the fact that the apps are backed by a Personal Cloud should not normally be evident to the user. The exceptions exist mainly where the Personal Cloud provides enhanced functionality, such as prompting the user for permission to release information. Other than those exceptions, all the user should see is the exposed functionality.

Some additional Personal Cloud directions are listed below. The strategy for deriving these is to imagine a future state where privacy is baked into the architecture and where sharing is enabled by policy. This is the opposite of today’s model where the architecture is built for sharing and privacy is provided by policy. Also, assume a future state where individuals have access to all the transaction data their vendors currently collect, and to the data from all their devices, and that all of this is accessible to the user in aggregate through their Personal Cloud.

Software — Extend Supply Chain

management systems all the way to the consumer so that the merchant is the next-to-last rather than the last stop in the chain. The opportunity here is to create the API of Me and My Things and to assist large Enterprise to understand the value in integrating with it. Considerable momentum already exists in Vendor Relationship Management applications, the consumer counterpart to Customer Relationship Management. In short, re-imagine business software as owned and operated by and for the individual.

There are a number of initiatives and groups working to provide infrastructure and resources to further these concepts. The unsurprisingly named Privacy By Design framework “advances the view that the future of privacy cannot be assured solely by compliance with legislation and regulatory frameworks; rather, privacy assurance must become an organization’s default mode of operation”. The Personal Data Ecosystem Consortium’s mission is to connect entrepreneurs and startups focused on user-centric personal data, advocate for individuals’ rights to tools and access to their own data, and to help existing businesses dependent on the old personal data ecosystem transform to become profitable in the new one. The Respect Network

extends the basic connectivity of the Internet with a community and context for trusted identity and interactions between people, businesses and devices. There is a growing collection of information, references and index of projects related specifically to Personal Cloud at Personal-Clouds.org. The folks at Kynetx have built and implemented CloudOS and an event network that are up and running today and built on the Personal Cloud philosophy. This is a non-inclusive list but will jump-start your investigation into Personal Clouds.

Data aggregation and integration — Much of the value of data lies in the ability to correlate and analyze across many datasets to find valuable relationships. For example, consumers can participate in cradle-to-cradle tracking of recyclable resources by integrating data from their purchasing, waste and recycle streams. Add Health data to the mix for insight into how dietary purchasing habits affect wellness. Integrate power and water consumption for a total household greenness rating. But, which vendor will collect and manage all this information for you?

Mint.com aggregates all your financial accounts into a single view but probably won’t be adding your

health records, power and grocery loyalty information any time soon, nor would you want it to. If Mint.com began asking for your health and grocery data, you probably would tell it “that’s personal”. The Personal Cloud version of an aggregator operates more like FileThis.com. FileThis periodically fetches your bank statements, e-bills and purchase receipts, then loads them into a data store that you control. It never stores the information it collects on your behalf nor does it attempt to provide any analysis or presentation of that data. You supply the credentials and the storage, it supplies the integration.

Apart from a few early examples, this field is wide open. Every loyalty program, interactive health service, government-hosted citizen database, social-media service, utility provider, identity provider, bookmark sync service, contact list or other service that holds your information is a candidate either to aggregate that data into a single view or pull it into the individual’s Personal Cloud. If those services hope to remain competitive, they will recognize the value in empowering individuals to access their own data and begin to cooperate with integrators and publish APIs.

Hosted apps and plugins — hosting

providers cannot process your data if they cannot see it. If you are to get any value out of the data you choose to keep private, you will need something that you control to correlate, analyze and visualize it. One approach, exemplified by OwnCloud, is a framework made up of database hosting, a plugin architecture and APIs. Out of the box, the system supplies cloud storage, calendaring, contact lists, music streaming and on-line photo galleries. But the real power is in the plugin architecture and APIs. For example, there is no personal health vault in the basic version of OwnCloud, but that easily could be added.

Alternatively, some Personal Cloud applications will run standalone. Consider the FileThis service mentioned previously. Among the other accounts to which it integrates, it can download your bank statements. But this requires you to give it your on-line banking credentials, which is a bit problematic. That issue would go away if you could run its application locally. The standalone app model provides all the benefits of the vendor’s integration know-how without exposing your banking account credentials.

Retro-fit — There are a lot of “dumb” appliances out there with plenty of useful life in them. Similarly,

there are very few existing houses wired for data or that have their own Personal Cloud servers. If we imagine a future world where all those devices are smart, how do we get from here to there? One way might be to take the Crutchfield approach and apply it to the Internet of Things.

Crutchfield built an unlikely business by betting that ordinary people could and would install high-end stereos into their own cars if they had access to tools and instructions. Crutchfield provided custom wiring harnesses that eliminated most of the complexity, access to tools and comprehensive instructions from an exhaustive database of vehicle makes and models. The bet paid off and it grew into a formidable player in the consumer audio business.

A similar model would work for Internet-enabling dumb devices. Assuming that you had the appropriate wiring harness, a Wi-Fi-enabled Arduino or Digispark could be dropped into a washer, dryer, refrigerator or other appliance in minutes. This could jump-start the Internet of Things and most of the research could be crowdsourced. There are business opportunities in performing the installations and in providing the code that sends and receives events from the devices

and turns those into more complex behaviors. Of course, getting a data fabric into the home is a prerequisite, so the installation of home-automation servers, cabling, power-over-Ethernet and Wi-Fi access points also will be a growth opportunity.

Stupid Cloud Tricks

To seed your mind with ideas for Personal Cloud development, imagine a world in which the most mundane objects and surfaces are smart and in which you can program behaviors based on interactions of devices and events in your life. The cloud also has access to all your transaction and demographic data, location and preferences. Now combine these in the most far-fetched way that you can think of.

Here's an example. My smart house would have individually addressable path lighting throughout. My dog would have an NFC or Bluetooth beacon on his collar so that the house could know exactly where he is. I would then program a behavior causing the path lighting to follow him around the house. Once he got used to that, I would train him that following the path lighting cues would lead to a reward. The last step would be to create a phone app (I'll call it "Fetch") that leads the dog from wherever he is in the house to

wherever I am.

I admit this is a very stupid cloud trick. I criticized vendors for moving wall switches to phone apps, and I just did the same thing with the dog. The point is that if you own the data and your devices talk first to you instead of the vendor, you are not constrained by the vendor's choice of integrations and can invent weird and wonderful behaviors for your stuff. (But you have to admit that a phone app that remotely controls the dog is kinda cool.) So let's try something a bit more practical.

Your power company has a rate plan that lets it shut off your water heater and air conditioner when it or a neighboring utility has a peak load shortage. But why limit this to water heaters and air conditioners? When the home is filled with smart devices, it will be possible for lights, fans, battery chargers or any powered device to take part in discretionary load abatement. For example, on receiving an abatement request, an LED bulb (or the dashboard controlling that bulb) might respond with an offer to cut 10% of its current drain during the abatement period. Because the house knows whether anyone is home, the degree of abatement can vary automatically and accordingly.

You don't have to honor the pledge and can turn the light back up, but the closer your house gets to its projected abatement, the larger the rebate you get.

With enough devices participating, discretionary load abatement will allow us to defer the need to build more power plants. But this considers only the first-tier effects. The system really gets interesting when social aspects are added. Your Personal Cloud knows how much load abatement you have provided, what your baseline usage is and the normal interior conditions in the house. Someone eventually will combine these into a competitive social app where individuals or groups can compete for energy-efficiency badges. How do you compare with your neighborhood? Your region? Households with similar demographics? Can your scout troop be the greenest in the pack?

What will your Stupid Cloud Tricks be? Feel free to dream some up and send them to ljeditor@linuxjournal.com. Or just wait a few years and create them for real.

Personal Cloud = Opportunity

These types of applications are not only possible but trivial when we

all have access to our own data and are not dependent on vendors for the integrations. Personal Cloud detractors claim that individuals have shown no interest in having access to their own data. Until PRISM, many said we have no interest in privacy either. But this is not a question of whether Personal Clouds will exist or whether individuals need or want access to their own data. Technology seeps into every niche that will support it, and this is hardly a niche. When the value of Personal Cloud apps exceeds their cost, they will flourish. It really is that simple.

The market for Personal Clouds is a superset of that for mobile phones, game consoles, NAS devices, home-entertainment servers and home-automation servers because they are all potentially participants. That is a very large market, which with regard to Personal Cloud, is as yet untouched. Those who can make the value of Personal Cloud exceed the cost (as measured in currency, skill requirement and administrative overhead) can begin to carve out their own piece of that very large pie.

Would you like your piece of that pie now? For more information on Personal Clouds and the ecosystem growing up around them, please visit the Personal Data Ecosystem

Consortium, Project VRM and the Respect Network. ■

T.Rob spent the last 20 years working on security, clustering, high availability and architecture of enterprise messaging. He intends to spend the next 20 applying these same technologies to benefit individuals through technologies, such as Personal Cloud and Internet of Things. He recently left IBM, where he was a product manager for the WebSphere messaging product family, to start loPT Consulting with the mission of putting People first in the Internet of Things. He can be found at <https://ioptcconsulting.com> or <https://t-rob.net>. (Full disclosure: T.Rob is a member of the Respect Network and the Personal Data Ecosystem Consortium, but please don't hold that against them.)

Resources

FileThis: <http://filethis.com>

Kynetx Developer Resources:
<http://developer.kynetx.com>

LastPass: <https://lastpass.com>

Mint: <http://www.mint.com>

OwnCloud: <https://owncloud.com>

PDEC: <http://pde.cc>

Personal Clouds.org:
<http://personal-clouds.org>

Privacy by Design:
<http://www.privacybydesign.ca>

Project VRM: <http://projectvrn.org>

Respect Network:
<http://respectnetwork.com>



DOC SEARLS

The First Personal Platform — for Everything

Computing started out corporate, then it got personal. Same thing happened with networking. Next up: the cloud.

Maybe the biggest thing that ever happened to Linux—at least scale-wise—is virtualization. As I recall, virtualization first materialized in a big commercial way with IBM, which started by putting many Linux instances on System z mainframes. (Once on the old Linux Show, we had a guest geek from IBM who said it was not only his idea, but also that he came up with it over lunch.) IBM didn't call those mainframes "clouds", but that's what it hosted. Now we have clouds of clouds of Linux all over the place. Nothing could be more widespread and ordinary. (Of Netcraft's ten most reliable hosting company sites for June of this year,

eight are Linux and two are FreeBSD: <http://news.netcraft.com/archives/2013/07/01/most-reliable-hosting-company-sites-in-june-2013-2.html>.)

Now think about the Internet of Things, often abbreviated IoT. It is generally assumed today that the Internet of Things will require embedded smarts. But in fact, any thing can have a cloud, whether the thing has embedded smarts or not. This insight comes to us from Phil Windley (<http://www.windley.com>), the hacker-in-chief of Kynetx (<http://kynetx.com>), a small Utah start-up with very big plans. (Disclosure: I sometimes consult them, as I do a number of other companies.) Phil is also the inventor and alpha maintainer

of CloudOS, a small and simple cloud operating system for anybody and anything, including you and every thing you own. CloudOS is open source and GPL'd. So is KRL (kinetic rule language, http://en.wikipedia.org/wiki/Kinetic_Rule_Language), the first language for programming on CloudOS (among other things), also first authored by Phil.

By abstracting intelligence away from physical things, we can unburden those things of the need to be intelligent in themselves. In fact, we can enlarge to absolute the variety of things that can have intelligence. Phil embodies this range in the word "pico", for *persistent compute object*. One of his is a pothole in front of his house (http://www.windley.com/archives/2013/04/potholes_and_picos.shtml). That pothole has brains in the cloud Phil gave to it, and that cloud is in Phil's personal cloud (http://personal-clouds.org/wiki/Main_Page). To help demonstrate how this can work in everyday life, here is a list of things I've made smart by giving each its own cloud:

- Canon 5D camera body.
- Canon 30D camera body.
- Dish Network VIP 922 set-top box, with Slingbox.
- Eurorack UB802 audio mixer.
- LaCrosse Technology BC-9009 battery charger.

- Sangean PR-D5 radio.
- Delkin Sensor Scope.
- Ful backpack.
- InFocus Model LP-130 projector.
- Teac model HD-100 HD Radio receiver.
- Sirius Sportster satellite radio receiver.
- Garmin Legend HCx GPS.
- A 30" x 30" tablecloth that looks like the QR code shown in Figure 1 (https://squaretag.com/app.html#!/app/a41x178/squareTag_scanned&tagName=YUV6WT&token=E0BFBEA6-E8CD-11E2-BD96-D358A022FB09).

The doors to those things' clouds are the QR codes I've hung or stuck on them—and, in the case of the last one, the item itself is a QR code. There can be other doors as well (for example, URLs), but QR codes are handy because: a) as with Ethernet, the patent owners (Denso, in Japan) decided they would rather create more value than they capture, so they let anybody do anything they want with QR codes; and b) they're easy to scan with a smartphone.



Figure 1.
Doc's 30" x 30"
tablecloth
looks like this
QR code.

Until now, QR codes have had the misfortune of being exploited mostly by marketers, which is why they appear as “robot barf” on promotional jive all over the place. But now it’s time for the hackers to take over, which is what the ones working for Kynetx have done. They created a service (http://www.windley.com/archives/2013/01/introducing_squaretag.shtml) called SquareTag (<https://squaretag.com>), which hosts things’ clouds. SquareTag’s business model, for now, is selling hang-tags and stickers with QR codes on them. I’m using some of the tags and stickers for the things in the list above. (Note that SquareTag isn’t a silo. I can take my things’ clouds, plus my own personal cloud, and put them wherever I want. That means I can self-host them, put them in Dropbox, or stick them in some other cloud service.)

Through a feature called “safe and mine” you can present a message to any Samaritan who scans the QR code of a thing you’ve lost. For example, “This bag belongs to (your name). Text me at (your number).” From any computing device, you can write or change that message.

But here’s the biggest thing: *Every thing’s cloud is a platform for relationship*—between you (as the owner) and whoever else you welcome aboard: notably the companies that make and sell the things you’ve bought.

For example, I have messages waiting for the makers of many of the items above. Here are a few:

- Canon — *I’m in the market for a 5D Mark III when the price for a new one falls below \$2500.*
- LaCrosse — *I’ll be glad to testify my love for the charger in any promo you want to run. It’s the best charger I’ve ever used. The display also tends to flicker and fade. Is there an easy fix for that?*
- Sangean — *Gave the radio a 5-star review on Amazon, but I won’t buy another one like it unless it does HD too.*
- Garmin — *Love the sensitivity and the UI. What I want are more than 10,000 waypoints in memory and the ability to produce KML files.*
- Dish Network — *DishAnywhere is a great system. That’s mostly how I watch the VIP 922. What I’d like is to have access to all the menu items remotely through the browser UI and on the tablet app. Please notify me when that feature is ready. Thanks.*

I also can advertise to the world, should I wish, some or all of what I

say about those products in my cloud. Or, I can restrict what I say just to the companies I invite into a relationship, such as the ones above.

Likewise, any company (such as Canon, LaCrosse, Sangean and Garmin) can give every product it sells a unique cloud of its own, with its own QR code, and transfer ownership of that cloud to the customer along with the product itself. If the customer welcomes a relationship with the company, and the company agrees to the customer's terms of engagement (such as, "respect the privacy of this communication channel in the following ways"), the whole "own cycle" of a product becomes a much richer experience for both the customer and the company. The QR code then becomes what's called a "TalkTag"—meaning that its purpose is to serve as a way for the customer to signal his or her interest in talking to the company. For example, I can program the cloud of my Dish Network set-top box to make a scan of its QR code send a message to the company saying I'd like a call from an agent to help me work through a problem. I've talked to call-center people about this possibility and they love it.

What they love especially is that it's now possible to have a standard way for customers to relate to companies. The problem today is that every company's CRM (customer relationship

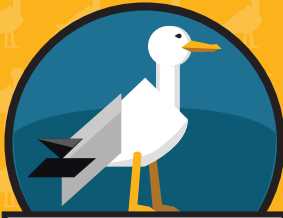
Advertiser Index

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

ADVERTISER	URL	PAGE #
All Things Open	http://www.allthingsopen.org	41
AnDevCon Fall	http://www.andevcon.com	2
Big Data TechCon	http://www.bigdatatechcon.com/SanFrancisco2013/index.html	47
Drupal Camp Atlanta	http://www2.mediacurrent.com/I/10072/2013-09-10/fw6gp	120
Emac, Inc.	http://www.emacinc.com	11
EmperorLinux	http://www.emperorlinux.com	31
iXsystems, Inc.	http://www.ixsystems.com	7
Manage Engine	http://www.manageengine.com	45
OVH	http://www.ovh.com	35
Seattle GNU/Linux Conference	http://seagl.org	119
Silicon Mechanics	http://www.siliconmechanics.com	3
USENIX LISA	https://www.usenix.org/conference/LISA2013	73

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>.



SeaGL

Seattle GNU/Linux Conference

Seattle needed a new grassroots Free Software summit, so we created one!

Join speakers and participants from
around the world for the inaugural
Seattle GNU/Linux Conference.

October 11th and 12th, 2013,
Seattle Central Community College campus.

→ Visit SeaGL.org for more information.

#DcATL

WWW.DRUPALCAMPATLANTA.COM



WWW.DRUPALCAMPATLANTA.COM

Drupalcamp

ATLANTA

October 27th 2012

Register Today for Drupalcamp Atlanta!

Our mission is to educate, train and evangelize
Drupal within our geographic region.



400+ expected at the 4th Annual Drupalcamp Atlanta on Saturday, October 27th. Drupalcamp Atlanta is catered towards everyone—curious beginners, designers, developers, and business owners are all welcome.



Drupal is an award-winning, open-source content management system that is powered by a LAMP stack. Learn more about Drupal at Drupalcamp Atlanta!



JOSH CLARK
KEYNOTE SPEAKER

DESIGNER SPECIALIZING
IN MOBILE DESIGN
STRATEGY AND USER
EXPERIENCE.

Authored Best iPhone Apps
(O'Reilly, 2009)
Tapworthy: Designing Great iPhone Apps
(O'Reilly, 2010)

*Agenda includes keynote
by Josh Clark, four
separate session-tracks,
and an after-hours party.*

- DRUPAL FOR BEGINNERS
- THEMING, DESIGN, AND USABILITY
- DEVELOPMENT AND PERFORMANCE
- DRUPAL FOR BUSINESS AND SERVICES



www.drupalcampatlanta.com