LINUX JOURNAL™
Since 1994: The Original Magazine of the Linux Community

A Look at the History behind systemd

MARCH 2015 | ISSUE 251 | www.linuxjournal.com

SYSTEM ADMINISTRATION

Build Lightweight Virtual Containers

PLUS

Get a Fully Capable Android Tablet for $20

Using Puppet's Hiera and Encrypting Credentials

Libreboot for a Free Software Laptop

WATCH: ISSUE OVERVIEW

LINUX JOURNAL ARCHIVE DVD

1994–2014

NOW AVAILABLE

www.linuxjournal.com/dvd

SILICON MECHANICS

SERVERS • STORAGE • HPC

Are you tired of dealing with proprietary storage?

zStax®
ZFS Unified Storage
Powered by NexentaStor

zStax StorCore ZFS Unified Storage from Silicon Mechanics is truly software-defined storage.

From modest data storage needs to a multi-tiered production storage environment, **zStax StorCore** ZFS unified storage appliances have the right mix of performance, capacity, and reliability to fit your needs.

zStax StorCore 64

The **zStax StorCore 64** utilizes the latest in dual-processor Intel® Xeon® platforms and fast SAS SSDs for caching. The zStax StorCore 64 platform is perfect for:

- small-medium office file servers
- streaming video hosts
- small data archives

zStax StorCore 104

The **zStax StorCore 104** is the flagship of the zStax product line. With its highly available configurations and scalable architecture, the zStax StorCore 104 platform is ideal for:

- backend storage for virtualized environments
- mission critical database applications
- always available active archives

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

CONTENTS

MARCH 2015
ISSUE 251

SYSTEM ADMINISTRATION

FEATURES

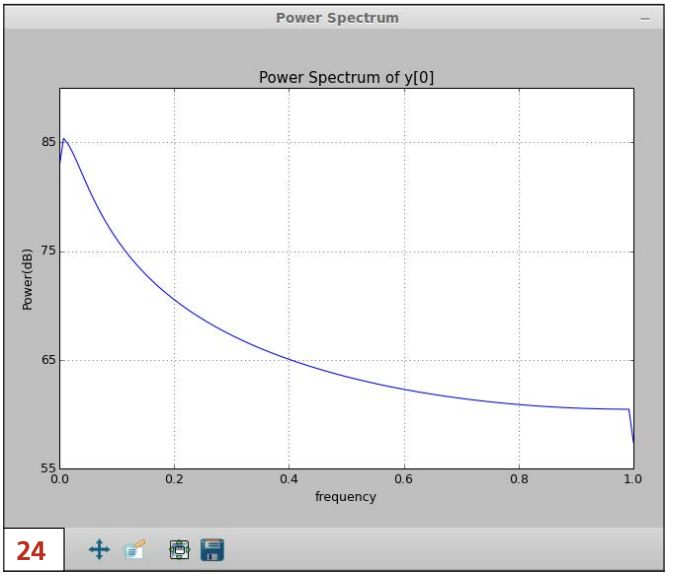
COLUMNS

IN EVERY ISSUE

ON THE COVER

- **A Look at the History behind systemd, p. 68**
- **Build Lightweight Virtual Containers, p. 82**
- **Using Puppet's Hiera and Encrypting Credentials, p. 58**
- **Libreboot for a Free Software Laptop, p. 38**
- **Plus: Get a Fully Capable Android Tablet for $20, p. 44**

22



24

44

LINUX JOURNAL (ISSN 1075-3583) is published monthly by Belltown Media, Inc., 2121 Sage Road, Ste. 395, Houston, TX 77056 USA. Subscription rate is $29.50/year. Subscriptions start with the next issue.

LINUX JOURNAL

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

Contributing Editors

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

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

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

Bases loaded?
Who's in *your* Lineup?

MinnowBoard Max

Netgate **Rookie**

Ht: 1 in. L: 4 in. W: 3 in. Wt. 5oz.

LEAGUE STATISTICS

CPU: Intel® Dual Core Atom™ E3825 1.33GHz
RAM: 2GB
Storage: MicroSD Slot
I/O: (1)Gb Ethernet, SATA2, (1)USB 3.0, (1)USB 2.0, micro HDMI, expansion headers
Integrated Intel HD Graphics, 1920x1080 max
Open Source hardware. Pick your OS: Windows®8.1, Android 4.4, FreeBSD 10, Linux , Yocto™Compatible

C2558

Netgate **MVP**

Ht: 1.5 in. L: 6.8 in. W: 7 in. Wt. 17 oz.

LEAGUE STATISTICS

CPU: Intel Quad Core Atom C2558 "Rangeley" 2.4GHz
RAM: 8GB
Storage: 4GB eMMC onboard
I/O: (6)Gb Intel Ethernet, SATA2, mSATA, (2) mPCI sockets (one with microSIM), USB
Fanless, low-power, NFV edge device
Ships with CentOS 7

APU 4

Netgate **Southpaw**

Ht: 1 in. L: 6.3 in. W: 6.5 in. Wt. 16 oz.

LEAGUE STATISTICS

CPU: AMD® G Series Dual Core "Bobcat" T40E APU 1Ghz
RAM: 4 GB
Storage: mSATA slot, SD card slot
I/O: (3)Gb Ethernet, USB, (1)SATA, DB9 serial, (2) miniPCIe sockets (one with micro sim)
Runs many flavors of BSD and Linux

C2358

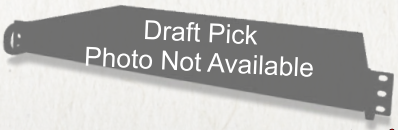
Netgate **Slugger**

Ht: 1.5 in. L: 6.8 in. W: 7 in. Wt. 17 oz.

LEAGUE STATISTICS

CPU: Intel Dual Core Atom C2358 "Rangeley" 1.7GHz
RAM: 4GB
Storage: 4GB eMMC onboard
I/O: (4)Gb Intel Ethernet, SATA2, mSATA, (2) mPCI sockets (one with microSIM), USB
Fanless, low-power internet gateway system
Ships with CentOS 7

C2758



Netgate **Heavy Hitter**

Size: 1U Rackmount Wt. TBA

LEAGUE STATISTICS

CPU: Intel Eight Core Atom C2758 "Rangeley" 2.4GHz
RAM: 8GB RAM
Storage: 64GB eMMC onboard. Yes, 64 GB!
I/O: (6)Gb Intel Ethernet, SATA2, mSATA, (2) mPCI sockets (one with microSIM), USB
Flexible low-power networking platform
Arrives with CentOS 7 and a low noise fan!

BeagleBone Black

Netgate **The Mascot**

Ht: <1in. L: 3.5in. W: 2.5in. Wt. 2oz.

LEAGUE STATISTICS

CPU: AM3358 1Ghz ARM®Cortex®-A8
RAM: 512MB
Storage: 4GB eMMC, microSD slot
I/O: (1)10/100 Ethernet, USB, HDMI, (2)46-pin headers, 3D graphics accel.
100% Open Source. Runs FreeBSD, Android, Angstrom, Debian, Nintendo, Beaglenmt, etc!

Netgate®

7212 McNeil Drive Suite 204 Austin, TX 78729 +1.512.646.4100

SHAWN POWERS

Putting Out Fires and Designing Fire-Proof Buildings

System administration is a very general term. It's our job to fix problems, repair systems and remind people to try power cycling their troubled desktops. We are also responsible for creating systems that don't develop problems, need fewer repairs and run without being power cycled. In an ideal world, system administrators would work themselves out of a job in short order. Thankfully (or unfortunately?), that's not how it goes. We always have problems to fix, and there's always a better way to do what we're doing. Thus, system administration is a vibrant and ever-changing field. This month, we learn how to be better at our jobs, even if the measure of "success" is constantly fluctuating.

Dave Taylor starts off this issue with a continuation of his script-based card game. Designing games with Dave is a great way to become better shell scripters, and so in a very real sense, we can justify playing games at work. Kyle Rankin follows Dave with a nerdier sort of game: trying to replace the proprietary BIOS on a ThinkPad with Libreboot. Coreboot is an open-source BIOS replacement, and Libreboot goes a step further by stripping out all the proprietary code. If you think having a free BIOS with built-in GRUB sounds interesting, you'll want to check out Kyle's column this month.

VIDEO:
Shawn Powers runs through the latest issue.

My personal contribution to the System Administration issue is something I find to be more useful than I ever expected. Android tablets are convenient for things like Wi-Fi sniffing, but they are often unwieldy to carry around. My solution is to convert a cheap pre-paid cell phone into a tiny, pocket-size tablet. If you already have an Android phone, it might be redundant, but for me, a $20 tablet was too hard to pass up. In my column, I give you all the details.

Puppet is an incredible tool for managing the system configurations of multiple nodes. Scott Lackey describes a great tool we can use to store site-specific data more efficiently (and securely). Hiera is a key/value lookup tool that integrates directly with Puppet and makes a great tool even better. If you want to have a clear separation between your sensitive data and the Puppet system that uses it, or if you want to save time by reusing common data, Hiera is a tool any Puppet admin will want to check out.

Jonas Gorauskas gives us a history of systemd. Whether you love the new initialization system, or think it's a terrible implementation of a horrible idea, systemd is here to stay—at least for a while. If you've ever been curious how we got from simple init scripts to SysV and beyond, you'll want to read Jonas' article. Once you understand systemd, Charles Fisher follows up with a great tutorial on using the new init system to create powerful and lightweight virtual containers utilizing systemd for initialization. For stubborn SysV lovers like myself, it's great to read some information on the advantages systemd might offer.

Doc Searls closes out our issue with a new look at the 15-year-old Cluetrain Manifesto. If you're a fan of the Locke, Levine, Weinberger and Searls project, you'll want to read what's happening with New Clues today.

If it weren't for the modern technological world we live in, system administration wouldn't even exist! Thankfully (or again, unfortunately?), our world is getting more and more technological every day. The need for system administrators and their tools are more in demand than ever before, and this issue of *Linux Journal* was written to educate, inform and even entertain those of us in the digital trenches. We hope you enjoy this issue as much as we 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

Digital Format

An interesting thing happened. I dropped reading *LJ* a while back due to hating to stare into a monitor. But last week I finally purchased an Amazon tablet and re-subscribed to *LJ* because of the *LJ* app. It's now easy to read on a nice screen. Even though I still enjoy printed magazines, I do respect the environment and agree that chopping down green for this is not good. So good choice on an environmentally-friendly mag.

—Peter K.

Thanks Peter! Paper magazines have a dear place in my heart as well, but I can't deny the digital format has some advantages too. I'm glad you're back in the fold, welcome home!—Shawn Powers

Vagrant Simplified

Regarding Shawn Powers' "Vagrant Simplified" in the January 2015 issue: great article. I tried Vagrant a few months back, and I couldn't get the light bulb to turn on. Thus, I put it aside. Shawn's article supplied the understanding I was missing. Many thanks.

—Tim Parks

That's exactly what I was hoping for! I'm glad it worked, and I'm glad Vagrant is demystified for a few more people. Thank you for the kind words.—Shawn Powers

Suggestion for Dave Taylor

A while ago I wrote a C++ program for downloading stock and option information from Yahoo in Windows. I remember it took a lot of code to parse the information, most particularly the option information.

Since then, I have graduated to Linux, and I am currently running on Xubuntu. A friend piqued my interest in option trading that caused me to revisit coding a Linux version of option tracking. I have

not yet gotten around to coding any GUI stuff, and I wanted something quickly, so I just wrote a few C++ programs using Geany to run on the terminal.

I really didn't want to write hundreds of lines of code to parse the data and looked around for some XML parsers. None looked easy enough for me to use, but then I looked at the source page, which I downloaded, and tried `grep`, which led me to develop the following few, or one, line(s) of code that I thought you might be interested as a source for some future articles:

```
wget -O /tmp/_option.html
 ➥http://finance.yahoo.com/q/op?s=SPY&date=1429228800

grep 'option_entry\|:volume' /tmp/_option.html | sed -n
 ➥'s/\r//;s/[^>]*//;s/>//;s/<\/div>//;p' | sed
 ➥'s/<\/strong>//;s/[^>]*>//;s/<\/a>//;s/%//' > /tmp/_option.txt

grep -A8 SPY141226P00230000 _option.txt
```

These three lines extracted the following information:

```
SPY141226P00230000
24.27
22.87
24.09
0.00
0.00
10
21
65.14
```

Of course, with a slight change to:

```
wget -q -O /tmp/_option.html
 ➥http://finance.yahoo.com/q/op?s=SPY&date=1429228800
 ➥&& sleep 1 && grep
 ➥'option_entry\|:volume' /tmp/_option.html | sed -n
 ➥'s/\r//;s/[^>]*//;s/>//;s/<\/div>//;p' | sed
 ➥'s/<\/strong>//;s/[^>]*>//;s/<\/a>//;s/%//'
 ➥| grep -A8 SPY141226P00230000
```

I could say that we can use "one line" of code to parse the Yahoo finance page for the option information!

This is C++ in that I use system to execute. Maybe a popen function might be better as an alternative, but I didn't think of it at the time.

In summary, I thought this was pretty cool, and you may have already done something similar, but as I said, I thought it might give you some ideas for future articles. I enjoy and find your articles educational, which are usually one of the first I read after the titles that catch my eye.

—Roger

Dave Taylor replies: *Thanks for your note and code snippet, Roger. It is rather amazing what you can do with* `sed`*, although when it gets that complex, you might consider having the script in a separate file and using the* `-f FILE` *option to* `sed` *to retain your sanity as you debug it. The problem with all of these crude HTML parsers, of course, is that if they make the slightest tweak on the page, your code's broken. I know; it happens to me all the time.*

Response to Letter in the January 2015 Issue Regarding zbackup

Regarding Chris Wills' letter in the January 2015 issue [this letter is from David Barton, author of the article "Ideal Backups with zbackup" in the November 2014 issue]: currently I use rsnapshot to back up the zbackup stores with hourly, daily, weekly rotations. Because the zbackup store changes very slightly each time, it is very space-effective. Due to the IO load caused by large numbers of files, very large numbers of servers may want to look at options that don't require linking all the files, such as rotating thin provisioned snapshots, Btrfs/ZFS snapshots or rotating onto removable storage media like tape. I don't think the snapshots need to be replicated, since it is a guard against malicious file corruption—for example, an administrator inserting random bits into the files.

Also, for readers who are interested in using zbackup to back up very large directory structures, there is a pre-release of software on **https://github.com/davidbartonau/zbackup-tar** that backs up directories about 10x faster on a non-SSD.

—David Barton

Kyle Rankin's Dr Hjkl on the Command Line

Regarding Kyle Rankin's article "Dr Hjkl on the Command Line" in the December 2014 issue of *LJ*: it seems that Mr Rankin wants to use vi keystrokes to manipulate the shell command line, so why is he explaining Emacs mode keystrokes? In the shell, all he needs to type is `set -o vi` and use vi mode from then on. Hit Esc to enter command mode, then hop to the previous word with b, next word with w, change the current word with cw and so on. Even "hjkl" are active, for moving the cursor to the previous/next letter/shell command.

The mechanism is called GNU readline; it supports both Emacs (default) and vi mode (put `set editing-mode vi` in ~/.inputrc),

and most command-line tools like the shell or the MySQL client or the GDB debugger will behave accordingly because they're using the library.
—Mike

Dr Hjkl on the Command Line, II

In the December 2014 issue, seriously, Kyle: `set -o vi`.
—Xaveer

Dr Hjkl on the Command Line, III

In Kyle Rankin's December 2014 column, he describes being comfortable with the vi command set, as I am too. He then goes on to describe bash command-line editing capabilities, using lots of Ctrl and Alt keys. If you've ever been sucked into editor wars, I'm sure it occurred to you that those key sequences seem awfully Emacs-like.

And, in fact, that's exactly what they are. Bash starts out with its command-line editing in Emacs mode. However, bash also has a perfectly functional vi mode that may seem more familiar to you. Just do `set -o vi` to turn on vi mode.

Now, having said that, I have to admit that I leave my bash sessions in Emacs mode almost exclusively. The vi mode, like the vi editor, is modal, and that modality is somewhat non-intuitive in command-line editing. However, for a vi fan, it's certainly worth exploring.
—Tim Roberts

Kyle Rankin replies: *I remember when I first got really interested in vi that I changed the command line to vi mode. I realized pretty quickly though that I didn't like having modes on the command line and switched it back. In general, I try to keep my environments set to their defaults, so you won't find me with custom bashrc files that set a lot of aliases or anything like that. It's just too much of a pain to ship custom settings like that throughout all my home and work systems, so instead, I try to make the most with the defaults I get.*

Digital Format

Happy New Year to you and your team. I had stop subscribing to this magazine some time ago and came back because of the digital format. Why? Because I am a seaman who is away for six months at a time. Without the digital format, it's hard to keep reading such a fine magazine. Seriously, I do not understand 60% of what is written, but if one keeps reading, surely one's knowledge will gradually improve. Keep up the good work.
—KokYY

Awesome! Yes, please keep reading. Then after a couple months go back and see if any of the older stuff makes sense. (Don't worry if it doesn't all make sense, however; sometimes the articles make my head spin too!)—Shawn Powers

They Said It

I just wanted to put in a good word for the They Said It column. The quotes are not always memorable (although they often are), but they always put me in a good frame of mind for enjoying the rest of the issue.
—Steven Janke

Thank you Steven. I enjoy looking for good quotes every month. The hardest part is making sure I don't repeat any (unless they're really good ones!)—Shawn Powers

Kyle Rankin's EC2 Security Groups

In the "Secure Server Deployments

in Hostile Territory" article in the January 2015 issue, Mr Rankin says: "... it's important to know that Security Groups are assigned only when an instance is created—you can't add or remove Security Groups from an instance after you create it."

In a VPC, which has been EC2's default for a good while now, this is not true. You easily can change the Security Group(s) associated with an instance.

—Gx

Kyle Rankin replies: *Thanks for the e-mail. It could be that I'm just showing my age in how long I've been working with EC2, since none of my accounts default into VPCs, and I'm still in the wild west of "EC2 Classic". That said, I do think having to think in terms of the limitations of the classic EC2 Security Group model helps build more robust security since you can take less for granted.*

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/enewsletters**.

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://ww.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

Nicolas Dichtel and **Thierry Herbelot** pointed out that the directories in the **/proc** filesystem used a linked list to identify their files. But, this would be slow when /proc directories started having lots of files, which, for example, might happen when the system needed lots of network sockets.

Nicolas and Thierry posted a patch to change the /proc implementation to use multiple linked lists instead of just one. Each subdirectory would have its own linked list, keyed to a hash of the directory's name. According to their benchmarks, the patch shaved 1/5 of the time needed to churn through all the entries of a given subdirectory.

Stephen Hemminger liked the speedup, but suggested that there already were implementations, like the **hlist macro**, that might simplify their hash table code.

Eric W. Biederman also liked the speedup and kicked himself for overlooking the /proc issue when doing other scalability work. But, he felt that the whole linked list concept was not the right approach. Especially, he felt that /proc/net/dev/snmp6 was the real target of Nicolas and Thierry's patch, and if no one actually needed the files in that directory (except people requiring extreme backward compatibility), it would be even more efficient to do away with them completely.

This, however, already had come up in an earlier thread, when **David S. Miller** had said that "It potentially breaks tools, it's a non-starter, sorry." So, reworking the user interface would not be allowed, which left the linked list speedup that Nicolas and Thierry proposed. But, Nicolas said he'd look into an **rbtree** implementation instead of a plain linked list, because rbtrees would potentially scale better.

Minchan Kim noticed that putting memory pressure on **qemu-kvm** under Linux 3.14 would cause a kernel stack overflow and crash the system. He dug into the

code and tried to reduce his own stack usage, but he wasn't able to cut back enough to prevent the crash. And in any case, he said, trying to reduce everyone's stack usage was not very scalable. He proposed expanding the kernel stack from 8K to 16K, although he acknowledged that there possibly were good reasons not to do this that he wasn't aware of.

Dave Chinner remarked that "8k stacks were never large enough to fit the Linux IO architecture on x86-64, but nobody outside filesystem and IO developers has been willing to accept that argument as valid, despite regular stack overruns and filesystems having to add workaround after workaround to prevent stack overruns."

He added, "We're basically at the point where we have to push every XFS operation that requires block allocation off to another thread to get enough stack space for normal operation", and said

"XFS has always been the stack usage canary and this issue is basically a repeat of the 4k stack on i386 kernel debacle."

Borislav Petkov pointed out that if they increased the kernel stack from 8K to 16K, there undoubtedly would come a time when 16K wouldn't be enough either. He wondered if there ever would be a limit, or if the kernel stack ultimately would grow to one megabyte and beyond.

Steven Rostedt said, "If [Minchan's patch] goes in, it should be a config option, or perhaps selected by those filesystems that need it. I hate to have 16K stacks on a box that doesn't have that much memory, but also just uses ext2."

Meanwhile, **H. Peter Anvin** said, "8K additional per thread is a huge hit. XFS has indeed always been a canary, or trouble spot, I suspect because it originally came from another kernel where this was not an optimization target."

At around this point, **Linus Torvalds** remarked that something like Minchan's fix probably would be necessary at some point, although the development cycle was already at -rc7, making it too late for that particular kernel version. Linus also pointed out that there was plenty of room to reduce stack usage in the stack trace Minchan had posted in his original e-mail. Linus remarked, "From a quick glance at the frame usage, some of it seems to be gcc being rather bad at stack allocation, but lots of it is just nasty spilling around the disgusting call-sites with tons or arguments. A *lot* of the stack slots are marked as '%sfp' (which is gcc-ese for 'spill frame pointer', afaik)."

There was a technical discussion about various ways to reduce stack usage in general (and some further consideration of ways in which GCC might be somewhat to blame), but with Linus willing to accept a patch to implement a larger stack, it seems like something along the lines of Minchan's patch will soon be part of the kernel. At one point, Linus summed up his position on the issue, saying, "Minchan's call trace and this thread has actually convinced me that yes, we really do need to make x86-64 have a 16kB stack. [...] The 8kB stack has been somewhat restrictive and painful for a while, and I'm ok with admitting that it is just getting *too* damn painful."

—ZACK BROWN

Android Candy: Bluetooth Auto Connect

I love my latest Android device (see this issue's Open-Source Classroom column for details), but for some reason, it won't automatically connect to my Bluetooth headset. When I turn on my headset, I want it to connect to my Android device so I can start using it right away. In order to make it connect, I have to go into the settings app, then Bluetooth, and then tap the device to connect. Thankfully, there's an application that makes life a lot easier.

Bluetooth Auto Connect is a program that runs in the background. It doesn't constantly poll for newly turned on Bluetooth devices, because that would waste battery power. It has several other ways to initiate the connection though. My favorite is the "connect when powered on" option. Because I always have to turn the phone on in order to start my audiobook (or music), it's not an inconvenience to turn the screen on in order to connect Bluetooth. As soon as the power button is pressed, it connects to my headset, and by the time I open the media player application, it's ready to rock!

Sometimes it's the simplest applications that are the most useful. Bluetooth Auto Connect is one of those. Check it out in the Google Play Store today: https://play.google.com/store/apps/details?id=org.myklos.btautoconnect.

—SHAWN POWERS

They Said It

Do something every day that you don't want to do; this is the golden rule for acquiring the habit of doing your duty without pain.
—Mark Twain

It's okay if you mess up. You should give yourself a break.
—Billy Joel

Let me tell you the secret that has led me to my goal. My strength lies solely in my tenacity.
—Louis Pasteur

If you limit your choices only to what seems possible or reasonable, you disconnect yourself from what you truly want, and all that is left is a compromise.
—Robert Fritz

The highest result of education is tolerance.
—Helen Keller

The Free Software Foundation, GNU Project and MIT's SIPB invite you to

LIBRE PLANET 2015

A conference for everyone who loves free software

Keynotes

Karen Sandler
Executive Director, Software Freedom Conservancy

Benjamin Mako Hill
Board of Directors, Free Software Foundation

Sessions by

+ Hackers
+ Activists
+ Beginners
+ Users
+ Writers
+ Artists

March 21 & 22 at MIT, Cambridge, MA

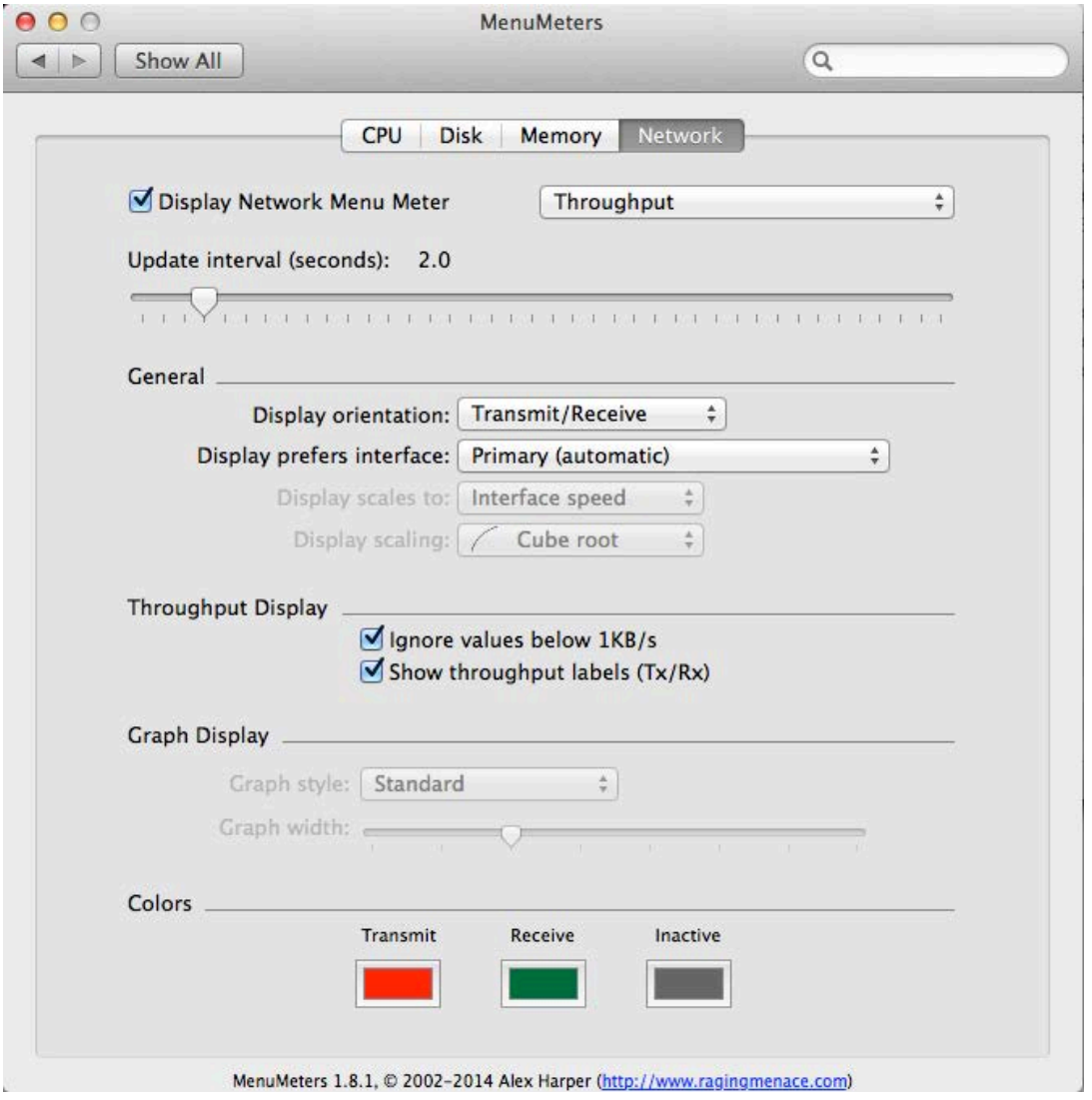
Students and FSF members attend gratis!

libreplanet.org/2015

Non-Linux FOSS: MenuMeters

It sounds like a "back in my day" story, but I really do miss the days when laptops had LED activity lights for hard drives and Wi-Fi. Sure, some still have

Menu Bar (screenshot from http://ragingmenace.com)



Customizing MenuMeters

them, but for the most part, the latest trend is to have no way of knowing if your application is pegging the CPU at 100%, or if it just locked up.

The hardware on Apple-branded laptops is amazing. Even if you hate the operating system, the solid aluminum cases are just awesome. Like most other brands of laptops, however, they lack any activity lights. A perfect fix for OS X is the open-source MenuMeters application. It puts all sorts of monitoring ability right in your menu bar. MenuMeters supports CPU activity, network activity and even memory usage. With a wide range of display options, you can customize MenuMeters to be as informative or subtle as you like.

MenuMeters is licensed under the GPL and is available to download at **http://www.ragingmenace.com**.

—SHAWN POWERS

Tighten Up SSH

SSH is a Swiss Army knife and Hogwart's magic wand all rolled into one simple command-line tool. As often as we use it, we sometimes forget that even our encrypted friend can be secured more than it is by default. For a full list of options to turn on and off, simply type `man sshd_config` to read the man page for the configuration file.

As an example, one of the first things I do is disable root login via SSH. If you open /etc/ssh/sshd_config as root, search for a line mentioning `PermitRootLogin` and change it to `no`. If you can't find a line with that option, just add it to the end. It will end up looking like:

```
PermitRootLogin no
```

Plenty of other security options are available as well. Disabling the old SSH version 1 protocol is as simple as changing (or adding):

```
Protocol 2, 1
```

Change it to:

```
Protocol 2
```

Then only the far more secure version 2 protocol will be able to connect. Every server situation has different security needs. Reading through the man page might reveal some options you never even considered before. (Note that the sshd dæmon will need to be restarted for the changes to be applied. Or, if in doubt, just reboot the computer.)**—SHAWN POWERS**

Solving ODEs on Linux

Many problems in science and engineering are modeled through ordinary differential equations (ODEs, **http://en.wikipedia.org/wiki/Ordinary_differential_equation**). An ODE is an equation that contains a function of one independent variable and its derivatives. This means that practically any system that changes over time can be modeled with an ODE, from celestial mechanics to chemistry reaction rates to ecology and population modeling.

Because of this ubiquity, many tools have been developed through the years to help solve and analyze ODEs. In this article, I take a look at one of the tools available on Linux: Model Builder (**http://model-builder.sourceforge.net**). The project is hosted on SourceForge, so you always can build it from source, but most distributions should have a package available. On Debian-based distros, you can install it with the command:

```
sudo apt-get install model-builder
```

It also installs several Python modules to support the tasks it can handle. If you do decide to build from source, you will need to handle these dependencies yourself.

Included with the source is a directory of examples. You can use them as a starting point and to gain some ideas of what you can do with Model Builder. Documentation is a bit sparse, so you may need to get your hands a little dirty to take the most advantage of what is possible with Model Builder.

To start Model Builder, you either can click on its menu item in your desktop environment or run the command `PyMB` from a terminal window. When the main window pops up, you are presented with a template where you can define the problem you are analyzing (Figure 1). The main pane, titled Differential Equations, is where you can define the set of ordinary differential equations that you are trying to solve. The general form of these equations is dy/dt = f(y,t).

If your system depends on different levels of differentiating the dependent variable, you always can rewrite it as a system of ODEs. When you give Model Builder your system, you need to write out only the right-hand side of the above equation. This equation can

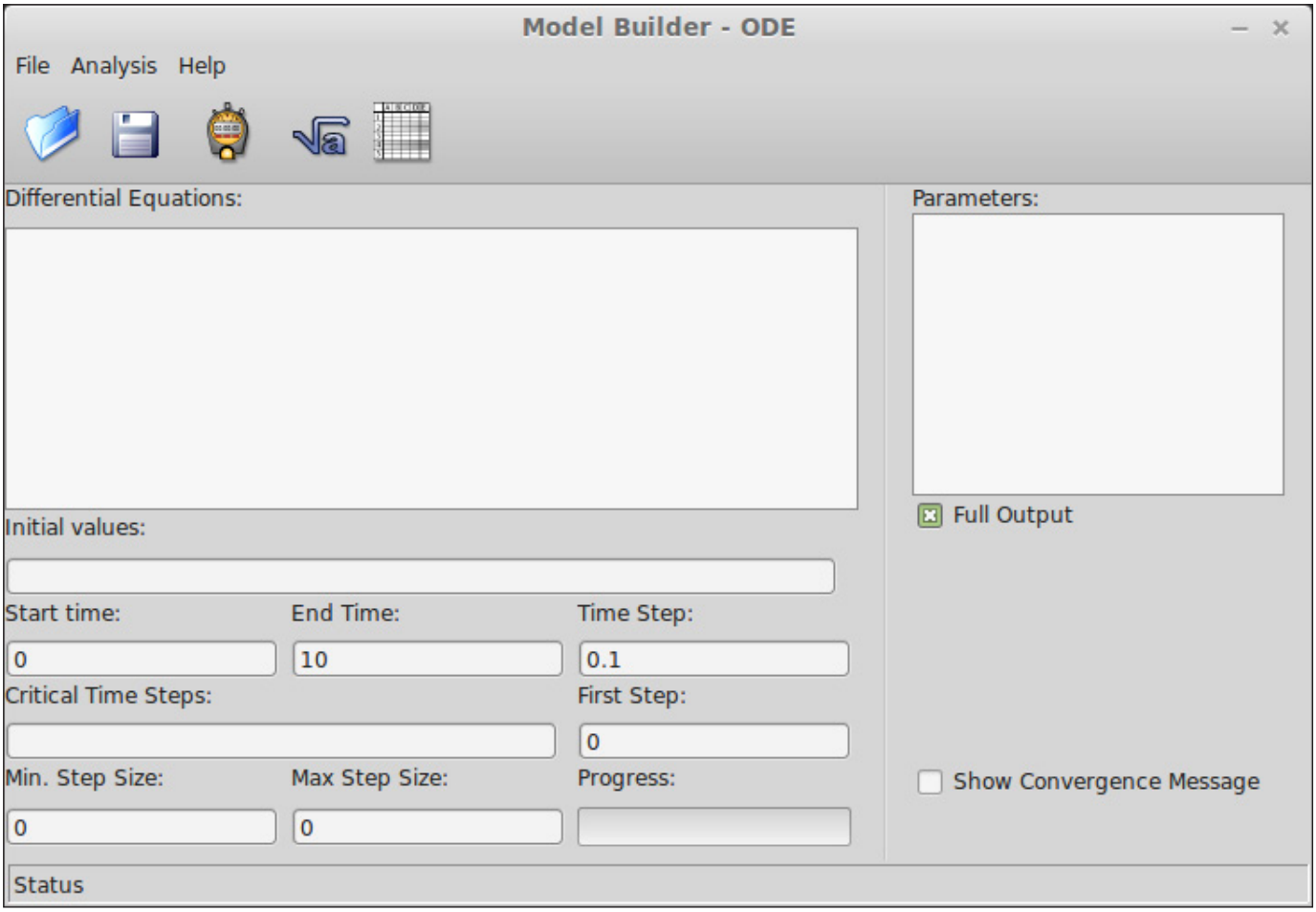


Figure 1. When Model Builder starts, you can set several parameters and the equations you want to analyze.

contain essentially any function or expression that NumPy understands, since Model Builder uses Python to do the heavy lifting.

Because Model Builder is designed to handle systems of equations, you need to define the y portion as elements of a list. So the y variable for the first equation is labeled as y[0]; the y variable for the second equation is labeled y[1] and so on. These are called the state variables.

The pane to the right of the equation window is where you can place any parameters that you need, one per line. They can be used in the equation window, where they are labeled as p[0], p[1] and so on. If you want to use time in either the parameters or equations that you have defined, you just need to use the t variable.

Because Python is used in the back end, you even can use lambda

functions to define more complex structures. You may want to take a look at the documentation available on the NumPy site to see what options are available (**http://www.numpy.org**).

Below these two panes is where you define the rest of the options for your problem. In the Initial values box, you can enter the initial values for each state variable at the time t=0. They need to be separated with a space and put in the order of the equations given in the equation pane.

Below the Initial values, you can enter the start time, the end time and the time step to use in the solution. The critical time steps box is usually left empty, so let's

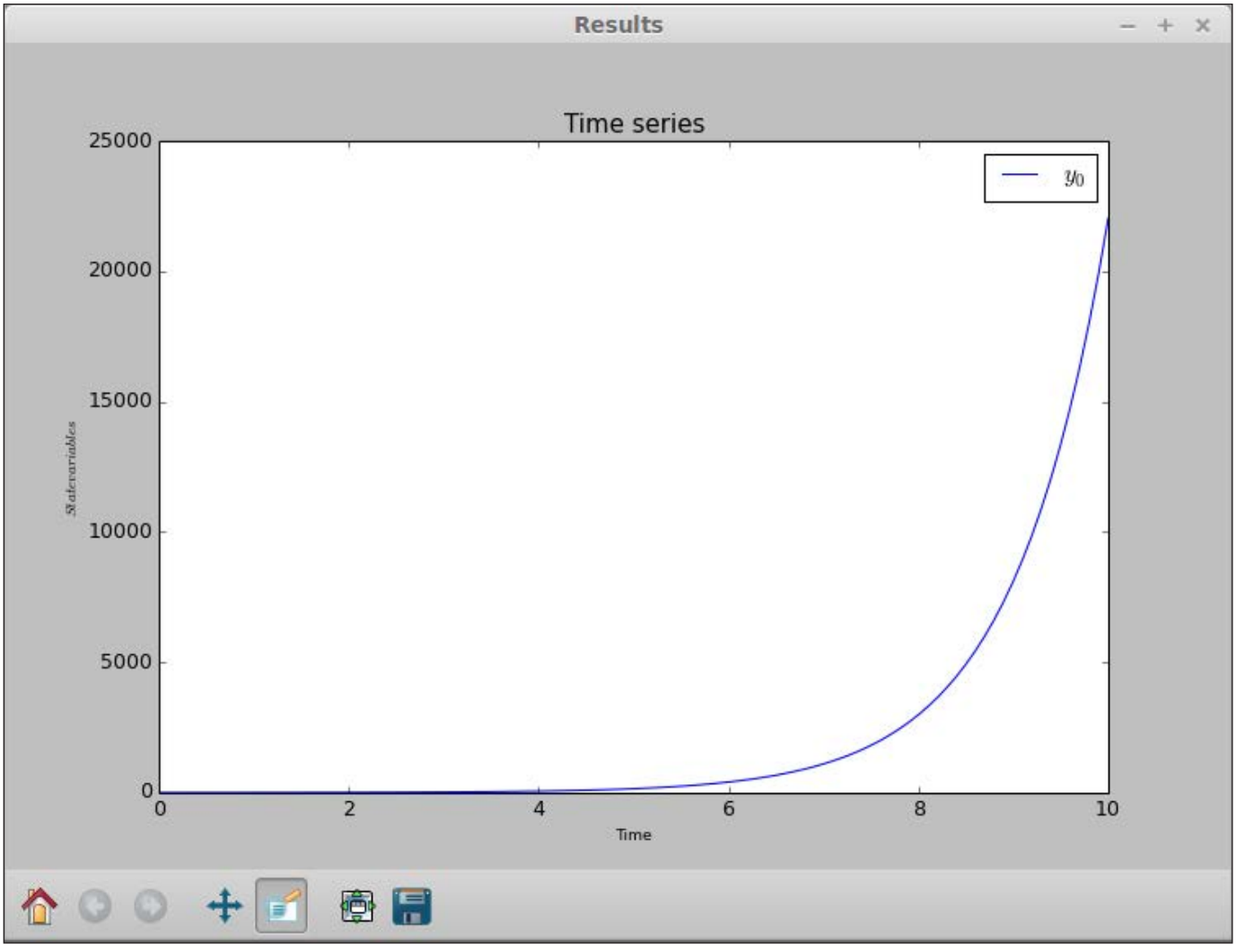


Figure 2. Once you finish defining the problem and run the integration, a result window pops up with a graph of the integration.

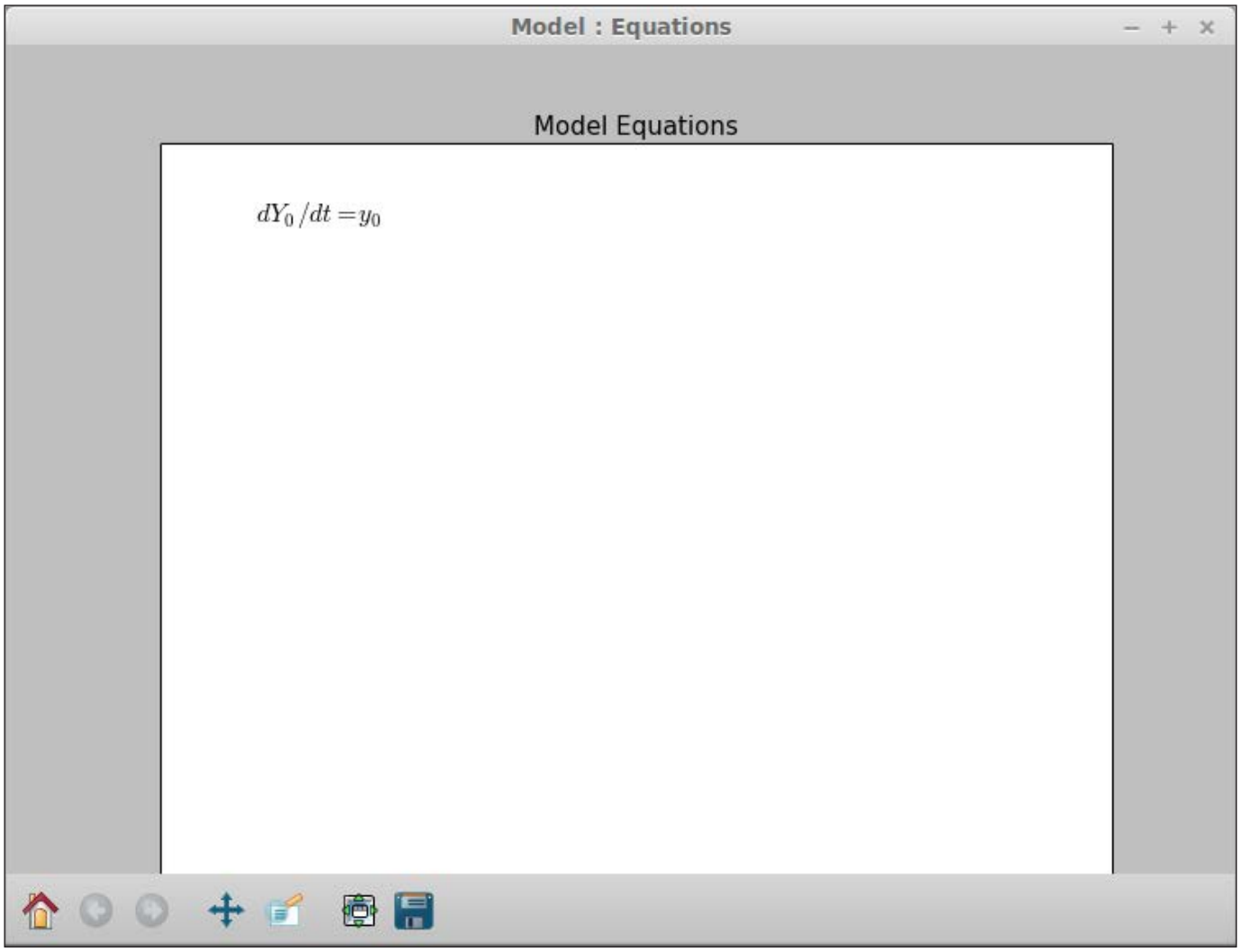
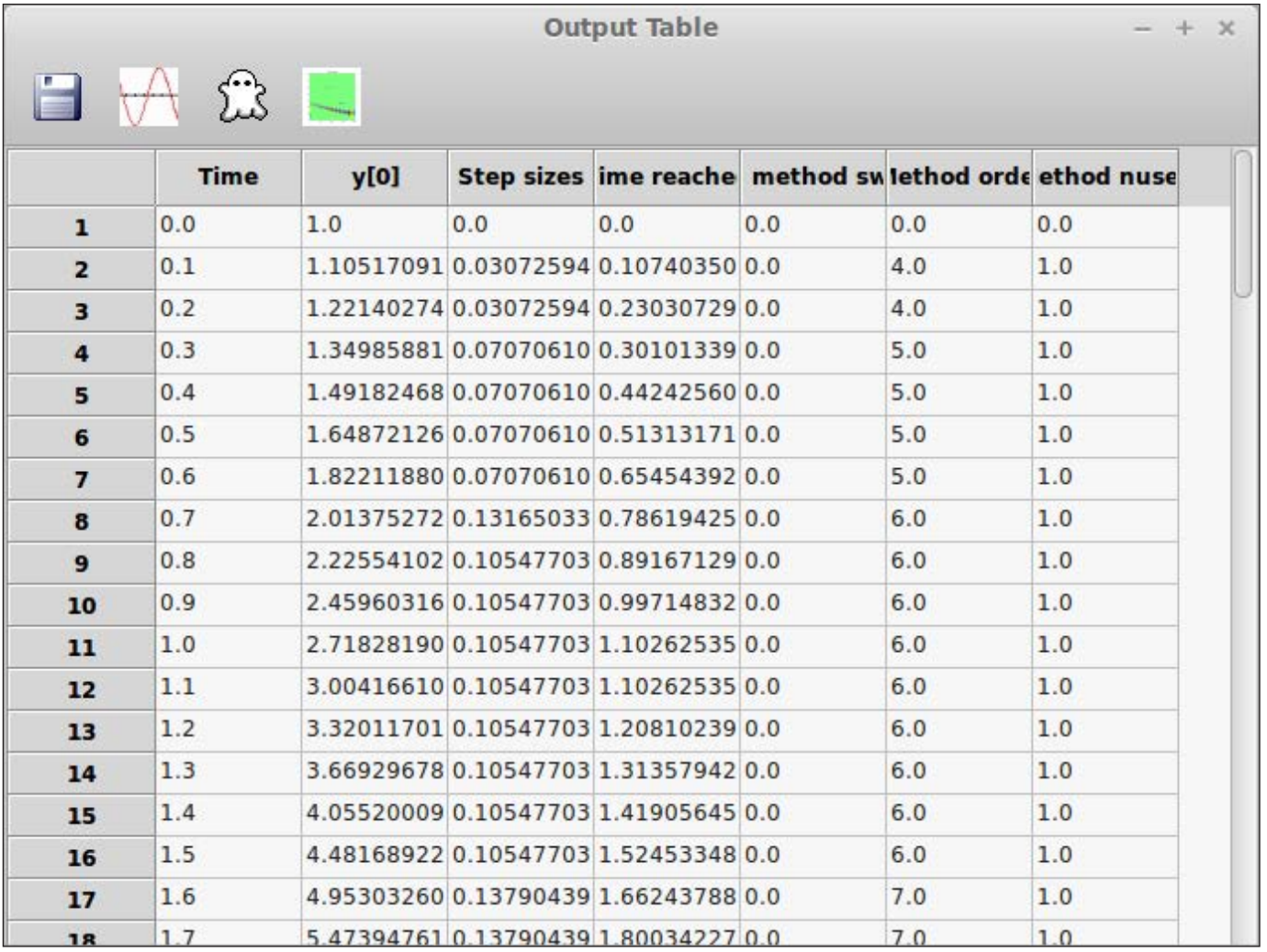


Figure 3. You always can get a typeset display of your equations to verify what they should look like.

leave it alone here. The first step box is the size of the first step. Usually, you should leave this as 0 to allow for automatic determination. The minimum and maximum step size boxes set these variables that are used in the variable step size algorithm. Typically, you should leave these as 0 as well to allow for automatic determination. The full output check box will print out more useful information about the integration in the results spreadsheet.

Once everything is entered, all you need to do is click the Start icon, and the integration will be calculated. If this is a system that you will want to work with over time, you can click on the menu



	Time	y[0]	Step sizes	ime reache	method sw	lethod orde	ethod nuse
1	0.0	1.0	0.0	0.0	0.0	0.0	0.0
2	0.1	1.10517091	0.03072594	0.10740350	0.0	4.0	1.0
3	0.2	1.22140274	0.03072594	0.23030729	0.0	4.0	1.0
4	0.3	1.34985881	0.07070610	0.30101339	0.0	5.0	1.0
5	0.4	1.49182468	0.07070610	0.44242560	0.0	5.0	1.0
6	0.5	1.64872126	0.07070610	0.51313171	0.0	5.0	1.0
7	0.6	1.82211880	0.07070610	0.65454392	0.0	5.0	1.0
8	0.7	2.01375272	0.13165033	0.78619425	0.0	6.0	1.0
9	0.8	2.22554102	0.10547703	0.89167129	0.0	6.0	1.0
10	0.9	2.45960316	0.10547703	0.99714832	0.0	6.0	1.0
11	1.0	2.71828190	0.10547703	1.10262535	0.0	6.0	1.0
12	1.1	3.00416610	0.10547703	1.10262535	0.0	6.0	1.0
13	1.2	3.32011701	0.10547703	1.20810239	0.0	6.0	1.0
14	1.3	3.66929678	0.10547703	1.31357942	0.0	6.0	1.0
15	1.4	4.05520009	0.10547703	1.41905645	0.0	6.0	1.0
16	1.5	4.48168922	0.10547703	1.52453348	0.0	6.0	1.0
17	1.6	4.95303260	0.13790439	1.66243788	0.0	7.0	1.0
18	1.7	5.47394761	0.13790439	1.80034227	0.0	7.0	1.0

Figure 4. You can pull up all of the results of your integration and do further analysis.

item File→Save to save the model to a file. This file format is an XML file, so you could edit it with a text editor if you want. When you are ready to do more work with it, you can load it by clicking on File→Open.

Once the calculations are done, which may be fast for simple problems, a results window will pop up (Figure 2). matplotlib handles this graph window, so you can manipulate it just like any other matplotlib window. This includes panning, zooming or changing the plot window. You also can save the resulting plot as an image file in one of several different formats.

Going back to the main window, let's look at some other available tools. Clicking on the Show equations icon pops up a window

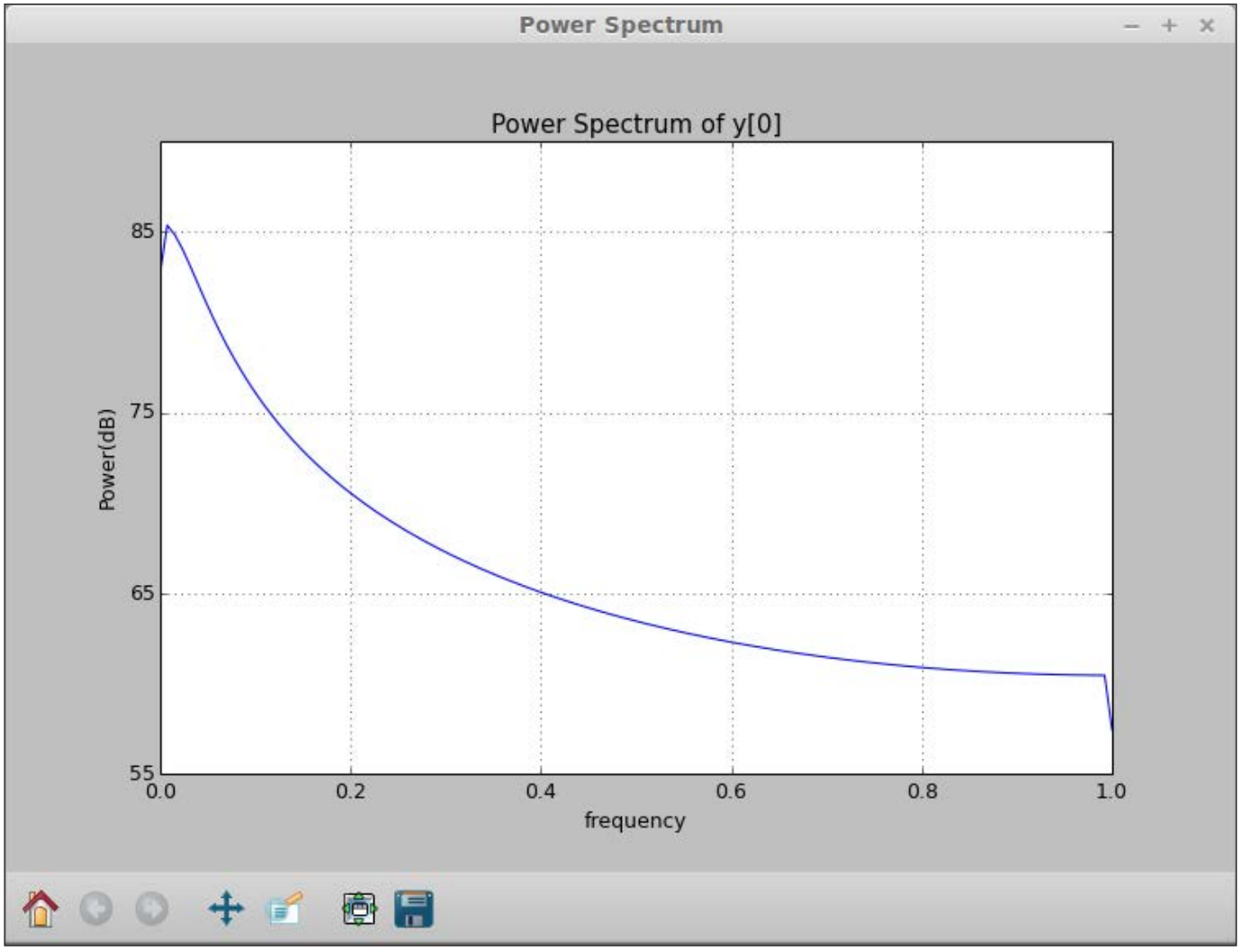


Figure 5. You can generate a power spectrum of any column of your results.

where you can see the equations typeset (Figure 3). Beside this icon is the Results icon. Clicking on that pops up a spreadsheet of all of the results from your integration (Figure 4). The columns of data include the time, the value of y[0] and the step sizes, among other things. You can select a couple columns by holding down the Ctrl key and clicking on the column headers. Then, click on the plot button to plot them in a new window. You can get a power spectrum for any one column by selecting one of interest and clicking on the Spectrum icon. This pops up two new windows, the first a power spectrum of the column (Figure 5) and the second a spectrogram of the column (Figure 6).

The last tool available is a wavelet

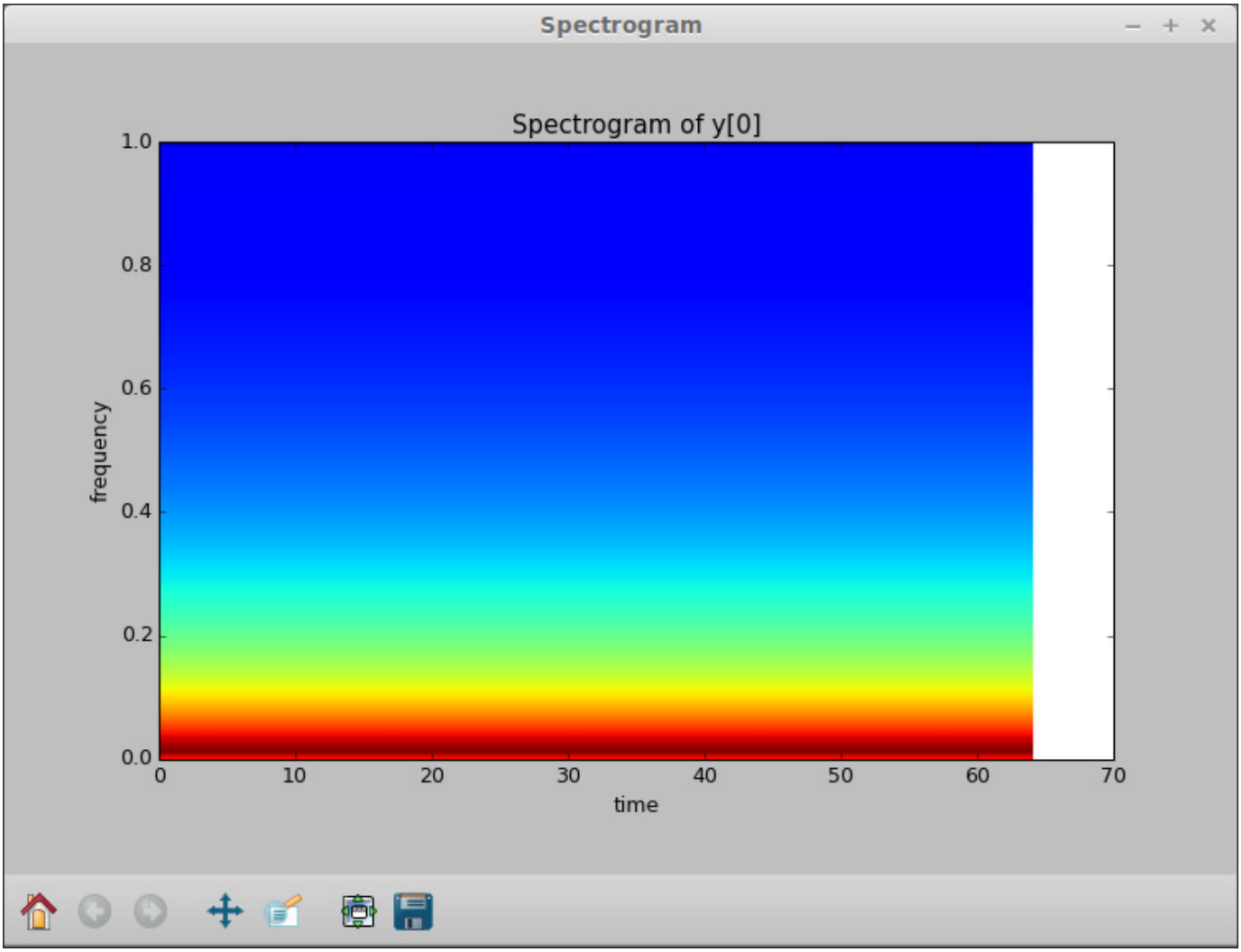


Figure 6. You also can generate a spectrogram of your results.

transform. When you select a column, you can apply a continuous wavelet transform to the data. When you are done with Model Builder, you can save this data into a comma-separated values (CSV) file from the spreadsheet window. Then, you can import it into other tools, like R, to do even further analysis.

Now that you have seen the options available in Model Builder, hopefully you will consider it when looking at ODE problems. It provides a pretty simple interface to the tools available in Python to solve ODEs numerically. Although other more powerful tools are available, Model Builder fits into the niche of experimenting quickly with different equations and playing with ideas.

—JOEY BERNARD

12th Annual

2015 HPC for Wall Street – Cloud Technology

April 6, 2015 (Monday)

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

Free Conference Registration for Qualified End Users. Go Online - www.flaggmgmt.com/linux

Plan to Attend:

2015 HPC for Wall Street will deliver top-notch content and connections.

Low-cost conference at $295. save $100. Full program, including lunch.

Free Conference Registration for qualified end users. Register online as end user.

Cloud Technology, Big Data, Low Latency, Networks, Data Centers, APIs, Scalability, cost savings for the global financial markets.

Leading Wall Street IT directors and vendor technology experts will speak on the program.

Speakers will cover 2015 Cloud, HPC and the latest programs to increase speed, put-through, and reduce costs.

Full conference program includes industry luncheon, general sessions, drill down sessions, exhibits, post show receptions.

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

2015 Sponsors

IBM, Cisco, redhat, hp, Arista, lenovo, Oracle, Scalable Informatics, Linux New Media The Pulse of Open Source, NYC NoSQL, Linux Journal, datanami, Traders Magazine, HPCwire, Securities Technology Monitor, Money management executive, EnterpriseTech, Integration developer news, WSTA Wall Street Technology Association

Show Hours: Mon, April 6	8:00 - 4:00
Conference Hours: Mon, April 6	8:30 - 4:50

Visit: www.flaggmgmt.com/linux

Dave Weber
Global Financial Services Segment Leader, Lenovo

Ken Barnes
SVP Corp Dev, Options Information Technology

Bernard S Donefer
Associate Director, Baruch College

Mike Blalock
Global Sales Director, Intel

Paul Jameson
Managing Director, Global Fin Services, Cisco Systems

Dave Malik
Senior Director, Advanced Services, Cisco Systems

Dino Vitale
Dir, Morgan Stanley Quality Assurance & Production Mgmt

Harvey Stein
Head of Credit Risk Modeling, Bloomberg

Fadi Gebara
Sr Manager, IBM Research

Terry Keene
CEO, iSys

Rob Krugman
VP Digital Strategy, Broadridge Fin Sols

Lee Fisher
VP Marketing, Redline Trading Solutions

Jeremy Eder
Perf Engineering, Red Hat

Matt Smith
Sol Architect, Red Hat

David B. Weiss
Sr Analyst, Aite

Rick Aiere
Architect Specialty, AIG

Shagun Bali
Analyst, TABB Group

Jeffrey Scheel
Senior Technical Staff, IBM Linux Tech Center

Ed Turkel
Mgr WW HPC Mkting, Hewlett-Packard

Charles Milo
Enterprise Technical Specialist, Intel

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

Davor Frank
Sr Solutions Architect, Solarflare

Phil Albinus
Editor, Traders Magazine, SourceMedia

Nmap—Not Just for Evil!

If SSH is the Swiss Army knife of the system administration world, Nmap is a box of dynamite. It's really easy to misuse dynamite and blow your foot off, but it's also a very powerful tool that can do jobs that are impossible without it.

When most people think of Nmap, they think of scanning servers, looking for open ports to attack. Through the years, however, that same ability is incredibly useful when you're in charge of the server or computer in question. Whether you're trying to figure out what kind of server is using a specific IP address in your network or trying to lock down a new NAS device, scanning networks is incredibly useful.

Figure 1 shows a network scan of my QNAP NAS. The only thing I use the unit for is NFS and SMB file sharing, but as you can tell, it has a ton of ports wide open. Without Nmap, it would be difficult to figure out what the machine was running.

Another incredibly useful way to use Nmap is to scan a network. You don't even have to have root access for that, and it's as simple as specifying the network block you want to scan. For example, typing:

```
nmap 192.168.1.0/24
```

will scan the entire range of 254 possible IP addresses on my local network and let me know which are pingable, along with which ports are open. If you've just plugged in a new piece of hardware, but don't know what IP address it grabbed via DHCP, Nmap is priceless. For example, the above command revealed this on my network:

```
Nmap scan report for
➥TIVO-8480001903CCDDB.brainofshawn.com (192.168.1.220)
Host is up (0.0083s latency).
Not shown: 995 filtered ports
PORT     STATE  SERVICE
80/tcp   open   http
443/tcp  open   https
2190/tcp open   tivoconnect
2191/tcp open   tvbus
9080/tcp closed glrpc
```

```
spowers@docboy:~$ sudo nmap -sS -O garfield

Starting Nmap 6.40 ( http://nmap.org ) at 2015-02-02 12:37 EST
Nmap scan report for garfield (192.168.1.10)
Host is up (0.00030s latency).
rDNS record for 192.168.1.10: garfield.brainofshawn.com
Not shown: 981 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
389/tcp   open  ldap
443/tcp   open  https
445/tcp   open  microsoft-ds
548/tcp   open  afp
616/tcp   open  sco-sysmgr
631/tcp   open  ipp
636/tcp   open  ldapssl
873/tcp   open  rsync
2049/tcp  open  nfs
3689/tcp  open  rendezvous
8080/tcp  open  http-proxy
8200/tcp  open  trivnet1
9091/tcp  open  xmltec-xmlmail
49152/tcp open  unknown
49153/tcp open  unknown
MAC Address: 00:08:9B:CF:8A:B5 (ICP Electronics)
Device type: general purpose
Running: Linux 2.6.X|3.X
OS CPE: cpe:/o:linux:linux_kernel:2.6 cpe:/o:linux:linux_kernel:3
OS details: Linux 2.6.32 - 3.9
Network Distance: 1 hop

OS detection performed. Please report any incorrect results at http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 3.71 seconds
spowers@docboy:~$
```

Figure 1. Network Scan

This not only tells me the address of my new Tivo unit, but it also shows me what ports it has open. Thanks to its reliability, usability and borderline black hat abilities, Nmap gets this month's Editors' Choice award. It's not a new program, but if you're a Linux user, you should be using it!

—SHAWN POWERS

DAVE TAYLOR

Let's Play Cards with Acey-Deucey, Part II

Dave adds the necessary code to turn a demo into a playable game, complete with some rule variants.

In my last article, I started developing a simple card game called Acey-Deucey, in which you deal two cards face up, then bet on whether the next card is going to be between those two in rank value. In other words, if a 5 of diamonds and a jack of spades were flipped up, the bet would be whether the next card was going to be between a 6 and a 10.

I also dug into the math too, if you missed it, because this is a great game for understanding odds and probability. Remember, any given card has a 1 in 52 chance of appearing, and because two cards already have been exposed, that means any given card actually has 1:50 odds.

For the example above, there are four 6s, four 7s, 8s, 9s and 10s, meaning that there are (4*5):50 -> 20:50, 2:5 or a 40% chance that the next card flipped up will indeed be between the two exposed cards. Make that 5 of diamonds an ace of diamonds, and the odds get crazy good: 80%. I'd take those odds!

The math will factor into the script because you actually can have the game suggest what to do based on the odds. The greater the spread, the better the odds—easy enough.

I ended my last article with the game being able to shuffle and deal three cards: two exposed and one hidden. Running the program with

More important, it also means that the game can identify situations where there's no point in betting, like when a 7 of diamonds and 8 of clubs are dealt out.

just that code results in this:

```
$ sh acey-deucey.sh
I've dealt:
   Ace of Hearts
   Queen of Diamonds
$
```

There's not much to do yet, because there's no game logic, so let's add some.

Turning the Code into a Playable Game

To start, let's initialize and deal out the cards. With the highly mnemonic function names already assigned, it's quite readable:

```
initializeDeck

shuffleDeck

dealCards

echo "Do you think the next card will be between? (y/n/q) "

read answer
```

This is good for a start, but as I mentioned earlier with the math discussion, it can be a bit more helpful, particularly knowing that the `dealCards` function ensures that the two cards displayed are in order of increasing rank, which means that this is a darn helpful addition:

```
splitValue=$(( $rank2 - $rank1 ))
```

More important, it also means that the game can identify situations where there's no point in betting, like when a 7 of diamonds and 8 of clubs are dealt out. There are no cards that can be between them. This is added with a simple test:

```
if [ $splitValue -le 1 ] ; then
  echo "No point in betting when you can't win!"
  continue
fi
```

The third card already has been "dealt" within the function `dealCards`, its rank calculated (as `$rank3`) and its display name set (as `$cardname3`). So, the test to see if the new card is or isn't between the two existing ranks is the next section of the

So you can pick three cards randomly out of the deck, you can calculate their ranks and display names, and you can prompt the user to guess whether the next card will or won't be between the two, then test to see if they were right.

code required, and it too is easy:

```
if [ $rank3 -gt $rank1 -a $rank3 -lt $rank2 ] ; then # winner!
  winner=1
else
  winner=0
fi
```

So you can pick three cards randomly out of the deck, you can calculate their ranks and display names, and you can prompt the user to guess whether the next card will or won't be between the two, then test to see if they were right.

What's left? Scoring. And, that's done with the `$won` variable, which is incremented in a conditional statement that appears immediately after the test to see if the third card is a `$winner` or not:

```
if [ $winner -eq 1 -a "$answer" = "y" ] ; then
  echo "You bet that it would be between the two and it is.
        You WIN!"
  won=$(( $won + 1 ))
elif [ $winner -eq 0 -a "$answer" = "n" ] ; then
  echo "You bet that it would not be between the two and
        it isn't. You WIN!"
  won=$(( $won + 1 ))
else
  echo "Bad betting strategy. You lose."
fi
```

You'll notice that in this implementation of Acey-Deucey, I'm allowing the player to win if he or she bet the card won't be between the two, and it turns out that it isn't. This is probably too generous, because all you need to do is pick the more likely scenario, which is to say any situation where the spread is six cards or less (like at the very beginning of this article).

Still, it's not Vegas or Atlantic City, it's just a shell script, right? So I'll be nice. If you'd rather not offer that option, simply change the message in the first `elif` conditional code block and skip incrementing the `$won` variable.

All that's left to do is to wrap the entire code block in a big loop that'll run forever, and use that standard technique of shell script programmers worldwide:

```
while [ /bin/true ] ; do
```

You probably wondered why /bin/true existed in Linux, didn't you? So that's the first line of the main code block, and let's increment the `$games` variable in the last line of the block:

```
games=$(( games + 1 ))
```

But there's one more fragment needed, and that's the test to see if the user guessed that the third card would or would not be between the two displayed cards, or if the user quit the game. In the latter situation, it's time to display some stats. That's easy enough, and it turns out that you can just leave `$answer` alone for the yes/no answer:

```
if [ "$answer" = "q" ] ; then
  echo "You played $games games and won $won times."
  exit 0
fi
```

In fact, you'll never quit the game by falling out of the while loop, but that makes sense since the conditional test of /bin/true is, well, always true.

Stitch all these fragments together and you have a game, by George!

```
$ sh acey-deucey.sh
I've dealt:
   6 of Hearts
   9 of Clubs
The spread is 3. Do you think the next card will
 be between them? (y/n/q) n
I picked: 9 of Hearts
You bet that it would not be between the two
 and it isn't. You WIN!
I've dealt:
   Ace of Hearts
   7 of Spades
The spread is 6. Do you think the next card will
 be between them? (y/n/q) y
I picked: 3 of Spades
You bet that it would be between the two
 and it is. You WIN!
I've dealt:
   7 of Spades
   10 of Spades
The spread is 3. Do you think the next card will
 be between them? (y/n/q) q
You played 2 games and won 2 times.
$
```

A perfect score. Nice. Las Vegas, here I come!■

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 just completed a 10th anniversary revision to the book, coming very soon from O'Reilly and NoStarch Press). He can be found on Twitter as @DaveTaylor and more generally at his tech site http://www.AskDaveTaylor.com.

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

KYLE RANKIN

Libreboot on an X60, Part I: the Setup

Find out what Libreboot is and why you should dust off that old ThinkPad and give it a fresh BIOS.

Recently I wrote a review for the *Linux Journal* Web site on the Purism Librem 15 laptop (**http://www.linuxjournal.com/content/purism-librem-15-review**). The goal of this laptop is to provide a piece of modern hardware that can run 100% free software not just for the OS, but also all device drivers and firmware up to and including the BIOS. At the time I'm writing this, the last major sticking point along those lines for the project is the Intel Management Engine: a proprietary piece of firmware that is required to boot up modern systems. In that review, I wrote the following:

> It turns out it's rather difficult to have a fully free software laptop. Even if you can pick hardware that can use free software drivers, there's still that pesky BIOS. While coreboot and libreboot are great free software BIOS implementations, to get it on many laptops requires hardware BIOS chip flashing with pomona clips—the kind of thing I wasn't ready to brick a laptop to try. Like other privacy advocates, I turned to the old ThinkPad X60 laptop series. While it's old, underpowered and has a low-res screen by today's standards, the keyboard is great and more important, you could flash its BIOS with coreboot or libreboot from within Linux itself—no hardware hacking required. So that's what I did.

Although the Purism 15 laptop seems to be a viable choice for those who want a free software laptop, at the time of this writing, the crowdfunding campaign is still in process, and even after it completes,

I've been able to find used ThinkPad X60 laptops on auction sites as cheap as $30, so if you are willing to live with some of the limitaions of hardware that old, it is an inexpensive route to a decent machine that runs only free software.

it will take some time until they ship. Plus, a new laptop like that doesn't come cheap, and many people who may want a laptop that runs 100% free software may not have $1,600+ to spend on it. I've been able to find used ThinkPad X60 laptops on auction sites as cheap as $30, so if you are willing to live with some of the limitations of hardware that old, it is an inexpensive route to a decent machine that runs only free software.

The first time I attempted to flash an X60 with coreboot, it was one of the more difficult things I'd done with Linux to the point that I wasn't ever planning on writing it up in *Linux Journal*. More recently, I tried again, only this time with Libreboot—a coreboot BIOS distribution that has all of the proprietary software removed. The process was greatly simplified and automated to the point where I feel relatively comfortable recommending others try it (with a few caveats I'll explain later).

In my next couple articles, I'm going to walk through the journey that brought me to the X60 running Libreboot that I'm using to type this column. In this first part, I discuss the setup, including what Libreboot is, what hardware it currently supports and some of the risks around flashing your BIOS. If I haven't scared you off by the end of this article, in future articles, I'll cover how to download Libreboot and verify its integrity, how to flash the BIOS itself in detail with sample script output and how to modify the default GRUB bootloader. If you can't wait until next month, a lot of my process is based on the excellent guide provided at **https://github.com/bibanon/Coreboot-ThinkPads/wiki/ThinkPad-X60**.

Free as in BIOS

To understand Libreboot, it helps to understand coreboot first. Coreboot is an open-source BIOS replacement. With coreboot, you can replace a proprietary BIOS with open-source

Libreboot is a custom distribution of coreboot that removes all proprietary software from the BIOS.

software on supported hardware with a minimal amount of proprietary firmware included to support things like video hardware in the BIOS or the Intel Management Engine on newer hardware. Coreboot doesn't currently support all hardware out there, although the list continues to grow, and you might be surprised to know that Chromebooks ship with coreboot by default. To install coreboot on much of the supported hardware, you must use external hardware including a connector like an 8-pin Pomona clip to reflash the BIOS chip. That's pretty intense for a lot of people, but fortunately, some hardware including the X60, X60s, X60 tablet and T60 can be flashed completely in software.

When I first attempted to flash an X60 with coreboot a few months ago, the process involved disassembling the laptop to inspect the underside of the motherboard with a magnifying glass so I could determine which of two BIOS chip types I had. I used that information to hand-patch the flashrom software with custom code and compiled a special version just to unlock my BIOS. Then I downloaded, configured and compiled a custom coreboot BIOS image for my laptop and went through a two-phase flash. In the end, I got it working; however, I needed to strip out and include the proprietary video firmware from my proprietary BIOS to get any video at boot time—useful when you want to select between hard drive and USB boot.

Libreboot is a custom distribution of coreboot that removes all proprietary software from the BIOS. Instead of proprietary BIOS boot selector, for instance, Libreboot boots straight into its own GRUB menu that you can use to load your own underlying OS. In addition, Libreboot has automated a lot of the difficult processes around installing coreboot and provides custom scripts and pre-build ROMs for its officially supported hardware.

But, why would you want a free software BIOS? For those who fully support the Free Software Foundation and the principles of free software, you don't need any further justification. Although I have traditionally taken a more pragmatic approach to the free vs. open-source software debate, I've recently been more motivated to seek out free software whenever I can find it as I

explain in my Librem 15 review:

> In the past, I didn't care all that much if I had to use a binary blob to get a wireless card or video card working as long as it worked, and I definitely never cared that my BIOS was proprietary software.
>
> Then the Snowden leaks happened. The sheer depth and breadth of the loss of privacy motivated me to step up my game in terms of overall security and focus on privacy. In the past it would seem rather paranoid to think that there might be some sort of NSA-sanctioned spyware in a binary blob, firmware, or the BIOS. After the Snowden leaks and the subsequent disclosures about the ANT catalog, these things stopped seeming so far-fetched. I found myself leaning more toward the Stallman camp. One of the only ways to be truly sure that you don't have a backdoor on your system is to be able to see the source code for all of it from the browser plugins to the kernel drivers all the way to the BIOS.

Supported Hardware

Due to the fact that Libreboot avoids any proprietary firmware in the BIOS, its hardware support is somewhat limited. Among other reasons, this is due to the fact that modern Intel hardware requires the proprietary Intel Management Engine firmware even to boot. Although you may be able to get Libreboot to work on other hardware, at this point, only a few laptops are listed on its hardware compatibility list (**http://libreboot.org/docs/hcl/index.html#supported_list**) as officially supported:

- Lenovo ThinkPad X60/X60s
- Lenovo ThinkPad X60 Tablet
- Lenovo ThinkPad T60
- Apple MacBook1,1
- Apple MacBook2,1

You may find one major thing in common with all the laptops on this list: they are old. In most cases, we are talking about 32-bit Intel Core Duo processors or 64-bit Core 2 Duos in some cases (and the T60's CPU can be replaced with a 64-bit CPU apparently). That said, the X60 is a decent piece of hardware with a solid keyboard and decent battery life, even if the CPU is slow and the screen resolution is low by today's standards.

Even on this list of supported

hardware there are some exceptions. Although all X60s are supported, only T60s that use Intel GPUs are supported, and those with ATI GPUs are not. The Libreboot hardware compatibility page has more information to help you figure out what's supported and what isn't. The page also lists recommended Wi-Fi chipsets that are known to work well with Libreboot and Linux in general, as they don't require any proprietary binary blobs to function.

Risky Business

If it doesn't already go without saying, reflashing the BIOS on your laptop with custom software is *risky*! Although I've had success so far flashing a couple different X60s, I did temporarily brick one laptop when I got fancy and tried an initial flash with one of my own custom ROMs instead of one provided by Libreboot. For the most part, the process is straightforward and automated, but as you'll see in my follow-up article that describes each step, many of the automated scripts call other software that output some pretty scary warnings and errors during the process that you are supposed to ignore.

There are two primary ways you can brick your laptop during the process. First, you could have a bad flash during the initial bootstrapping flash phase. If that happens but you were using one of the Libreboot-supplied ROMs, all you should have to do is shut off the machine, unplug the CMOS battery for a few seconds, reconnect it and power on your machine to get back to the original BIOS.

If you flash during the initial bootstrapping phase with a custom ROM like I tried one time, lose power during the process, attempt this on incompatible hardware or otherwise encounter a worst-case scenario, you could end up with a completely unbootable machine. Because you can't boot back to your OS, you can't attempt to reflash, so you are stuck with a bricked laptop unless you buy hardware that can flash your BIOS chip, such as a BusPirate or a Raspberry Pi running custom software. That said, if you have that hardware, wire it properly and you remembered to back up your original BIOS first, you should be able to restore your laptop to normal.

Although so far I've been successful when I've stuck strictly to the directions, there is still a possibility you will brick your laptop, so if you are particularly attached to your laptop and can't risk it being out of service while you acquire hardware flashing tools, you may

want to reconsider going down this road. Again, you can get used X60s relatively cheap on-line if you shop around, so if you are concerned, you may want to try this first with a sacrificial machine.

Conclusion

Well, if I haven't scared you off yet, I hope you check out my next column in this series where I jump right into step-by-step instructions on how to flash an X60 with Libreboot. Although the process isn't quite as simple as updating a traditional proprietary BIOS and requires a number of unusual steps, most of the hard work already has been done for you, and in the end you'll have a trusted machine without any proprietary firmware.■

Kyle Rankin is a Sr. Systems Administrator in the San Francisco Bay Area and the author of a number of books, including *The Official Ubuntu Server Book*, *Knoppix Hacks* and *Ubuntu Hacks*. He is currently the president of the North Bay Linux Users' Group.

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

SHAWN POWERS

The Teeny Tiny $20 Tablet

What's better than a pocket-sized Android tablet? One for $20.

For reasons other than "which do you like better", my cell phone is an Apple iPhone. Mainly it's because the rest of my family members use Apple products, and I want to be able to fit into their environment. With three teenage daughters, it's nice to run "Find my iPhone" and see why they're running late. That leaves me with two problems. First, there's the ridicule and teasing from my geeky friends. (You know who you are!) The second problem is that I really love Android apps for much of what I do on a day-to-day basis. My Nexus 7 is too unwieldy to carry around all the time, so I really need a tiny little Android tablet I can keep in my pocket. If the roles were flipped, I could just buy an iPod Touch and be done with it. It turns out things aren't so simple in the Android world.

I was able to find the Samsung Galaxy Player in several sizes, but not only did they cost hundreds of dollars, they also were discontinued before I ever could order one. It seems my demographic is tiny enough that it can't support a line of devices. Thankfully, my demographic is also pretty nerdy, so with a little research and hard work, I got a better solution altogether—for $20.

My Prepaid Non-Phone

The short version is that I bought a prepaid Android phone and never activated it. That version of the story leaves out some really important and really cool details, however. My end result is a pocket-sized Android device that I can use for listening to audiobooks via Bluetooth headset, make and receive calls, play games, and sorta use for a GPS device while driving. The best part is that my uber-micro-tablet really did cost me only $20.

If you're lazy, you can just buy a prepaid phone off the shelf and never activate it. Most (but not all) will allow you to cancel the

If you do some research and don't mind a little hard work, however, you can get a cheap Android device that does everything you want without any nag screens or limitations.

activation screen and use the device without cell service. If you do some research and don't mind a little hard work, however, you can get a cheap Android device that does everything you want without any nag screens or limitations. I describe my process here, and if it sounds like something interesting, you can do the same.

My Requirements

I wanted my new anti-iPod to be every bit as useful as the Samsung Galaxy Player would have been. Here's what I expected:

- A small but nice quality screen: I didn't want a cheap plastic screen that would haze over with tiny scratches. Preferably, I wanted Gorilla Glass.
- Wi-Fi: this seems obvious, but with cheap prepaid phones, you never can tell. It's always safest to make sure!
- At least a dual-core CPU: I didn't want a powerhouse, but I wanted to be able to *do* things with the device. I wanted at least 2GB of RAM as well, but I ended up settling for 1GB.
- MicroSD expansion slot: this is vitally important, because prepaid phones generally come with absurdly small amounts of internal storage.
- Bluetooth: the main purpose of this device will be to listen to audiobooks. For that, it needs to work with my knockoff-brand version of the Logitech HB-730.
- *Must be rootable*: this is as important as the MicroSD slot. With the advent of Android 4.4, you need to have a rooted device in order for applications to be able to write to the SD card. I personally think it's about the dumbest "feature" a new version of Android could offer, but at least with root access, it can be fixed.

- Must be affordable. I already have a phone (the iPhone), so I have to be able to convince my wife that it's not wasteful to buy a prepaid phone I never plan to activate. Happy wife = happy life!

My New Non-Phone

I spent a very, very long time researching what phone to purchase. Since what I was proposing goes against everything the prepaid vendors stand for, it's not like I could check their Web sites to see if the phones were rootable or if they'd work without activation. I considered several models:

1. Motorola Moto G from Verizon and Boost Mobile: the Moto G is a pretty decent-looking phone, and it has a beautiful screen. Unfortunately, although it has 8GB of onboard storage, it lacks a MicroSD expansion slot. It's also around $80, which is reasonable considering how nice of a device it is, but without that SD slot, it's more than I was willing to pay.

2. LG Volt from Boost Mobile: this phone is probably what I'd buy if I were going to buy another device right now. It checks all the boxes above, and it has really great battery life along with really great cameras. You currently can pick up this device for around $60, and for the hardware you get, I'm guessing Boost is losing some money on every sale.

3. LG Realm from Boost Mobile: the Realm is what I ended up buying (Figure 1). The specs are a step down from the Volt, but I was able to get the device for $19.99 from Best Buy at the end of 2014 in the "last chance to get Black Friday Sale Prices" sale. That sale probably still is running; it seems that's how Black Friday sales work nowadays.

Phone models change all the time. Rather than make decisions based on my findings from a few months ago, I urge you to look for the latest and greatest (or cheapest!) prepaid options out there, and make sure they meet your list of requirements. I can't stress enough how important it is for the phone to be rootable though, so do at least that much research before buying one.

The Rooting

Sometimes the hardest part of the process is to get out of the "ACTIVATE ME NOW" screen. With enough button pressing, I was able to put the

Figure 1. Oddly, a replacement battery for this phone costs more than the phone itself. At $19.99, you can't go wrong!

activation screen in the background. Every time the phone booted, however, it had the same annoying screen trying to force me to activate. Therefore, the very first thing I recommend doing is rooting the phone.

Usually, that's as simple as visiting **http://towelroot.com** from the phone's browser and installing the tr.apk file. As long as your phone is supported, it's literally 2–3 clicks, and your phone is rooted. Then install SuperSU from the Google Play store, and your phone is ready to hack.

It's important to note that rooting a phone is not the same as installing a third-party ROM. Although it's dead simple to root most phones, installing something like Cyanogenmod is far more difficult, and often it's not possible even if the phone is rooted. Thankfully, once the phone is rooted, the existing ROM can be made to function a little nicer. Getting rid of the nag screens is the first obstacle in that journey.

Stop the Nags!

Once your phone is rooted, it's time to start looking for the applications that are doing all the nagging. Unfortunately, this will take some googling, some guessing and a little bit of luck. The process is itself pretty straightforward:

1. Download a root-enabled file manager app like Root Browser or something similar.

2. Figure out what app(s) are responsible for the activation nag screens. Basically, google your phone's model along with "disable activation screen" or something like that.

3. Rename the apk files to add .bak at the end. In my case, I had to rename /system/app/LGDMSClient.apk to /system/app/LGDMSClient.apk.bak.

4. Reboot the phone and see if it worked. If it did, celebrate. If not, do some more googling, or just educated guessing, and try again. There is some danger here, but as long as you're not deleting files, just renaming them, most bad guesses can be reversed.

Other Anti-iPod Tweaks

Once your phone is working, and you're able to reboot it without the frustrating nag screens, head over to the settings app. Here is where you can disable all cellular data radios. Since you're not going to activate the phone with cell service, it will save some serious battery power if you disable the radios entirely. Depending on your model, you may have to disable 4G/LTE and 3G separately.

The one frustration I have is that try as I might, I've not been able to remove the cellular radio icon from the top of the phone (Figure 2). There are some apps in the Google Play store to remove icons, but they remove the Wi-Fi icon too, and that doesn't help me at all. Oh well, it's a small price to pay. Plus, it's a great way for me to keep track of

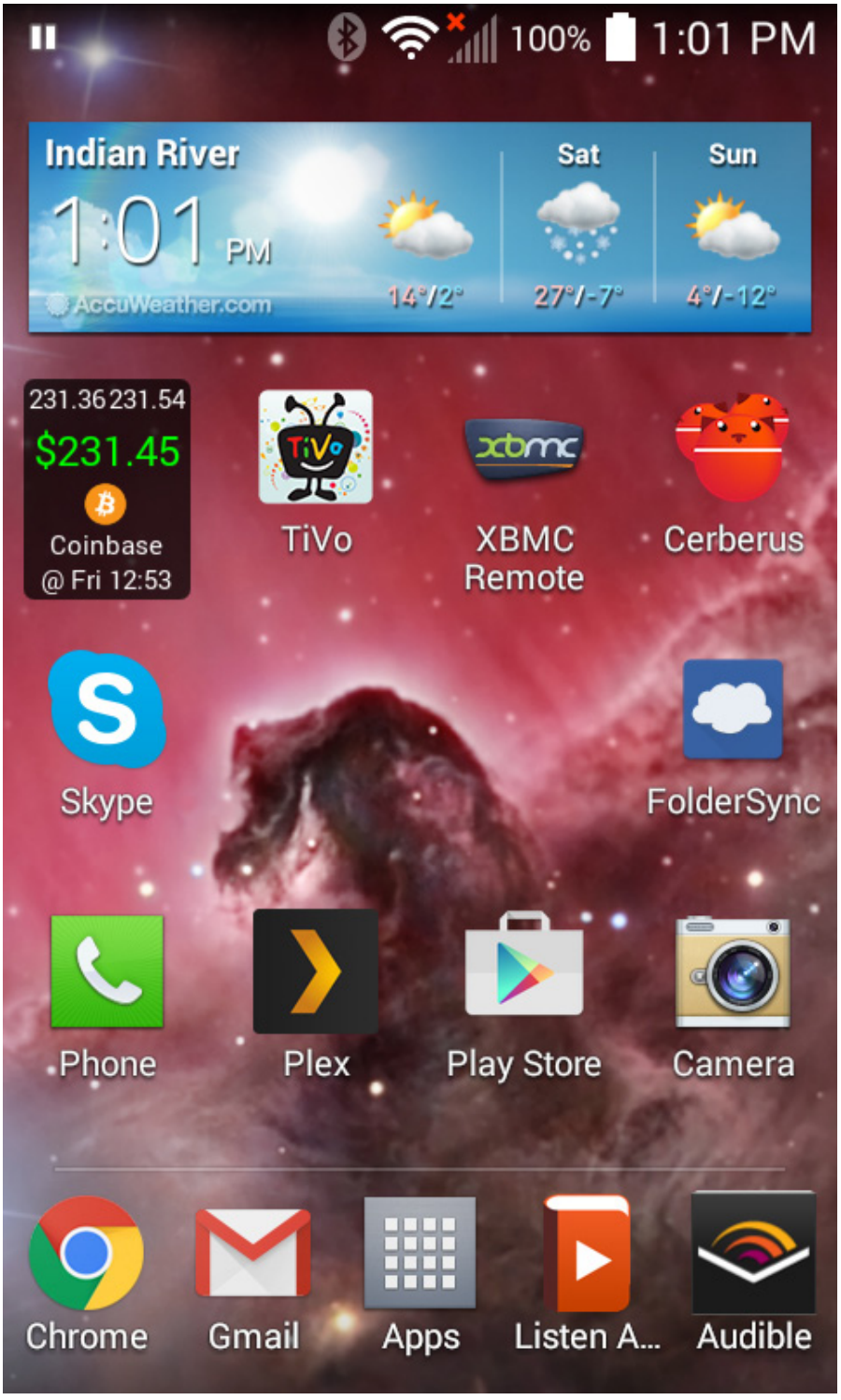


Figure 2. Although the cellular radio icon bothers me, it's the only frustration I haven't been able to eliminate!

Sprint coverage in my area. If it ever gets strong enough, I'll probably invest in a Karma router (http://www.yourkarma.com), which would make my new mini-tablet even more useful on the road.

One last recommendation I have is to download one of the "SD Fix"

apps from the Google Play store. With the advent of Android 4.4, the SD card isn't writable by apps like FolderSync, and as such, it makes managing audiobooks or MP3 files really difficult. With a rooted phone, it's another two-click solution to make your SD card functional again. I still can't believe Google crippled Android 4.4 like that. Thankfully, root access and Linux can save the day.

Here's My Number, Call Me Maybe

If the cell radio icon bothers me, just imagine how much it bothers me to have a phone with microphone and speaker, but no phone service. I know, I said I wasn't looking for a phone, but I have OCD, so that unused hardware really annoys me. Enter SIP.

Some phones come with a firmware that supports Android SIP calling out of the box. Most prepaid phones, however, disable that feature because they want you to use their service. It makes sense. Thankfully, you can download a third-party app called CSipSimple and add complete Wi-Fi-based SIP calling to your phone. It even integrates with the native dialer application, so you use it like a regular phone! I'm still using Google Voice service through **http://www.simonics.com**, but because that ability was supposed to stop almost a year ago, I wouldn't count on it working forever. Any SIP provider will work with CSipSimple, however, so even if the free Google Voice option through Simonics stops working, you can get the prepaid phone working without paying the cellular provider.

GPS!

It's hard to buy an Android device that doesn't come with GPS built in. Although the lack of cellular radio means you can't do real-time map downloading on the road (unless you have mobile Wi-Fi or a hotspot on your actual cell phone), it's simple to use your new mini-tablet as a GPS device with a little bit of planning. Google Maps allows you to download map data for specific areas locally to the device. This doesn't work great for long trips, because grabbing all those "map sections" is tedious, but for short trips to unknown locations it works well.

There are also several off-line GPS apps available in the Google Play store, and although most cost money, they're cheaper than buying a standalone GPS unit at the store. If I'm being completely honest, I still use a Garmin GPS for long trips, but that's probably because I'm old and don't always trust technology to work as expected.

VAULT

LINUX STORAGE AND FILESYSTEMS CONFERENCE

MARCH 11 - 12, 2015

REVERE HOTEL - BOSTON, MA

Vault will bring together the leading developers in file systems and storage in the Linux kernel with related projects to forge a path to continued innovation and education.

WHY YOU SHOULD ATTEND

- 40 technical sessions on the hottest trends and technologies in storage and file systems.
- Keynote presentations by top technical leaders from Facebook, NetApp, Red Hat, and SanDisk.
- A roundtable discussion with the track leaders from the invitation-only Linux Storage Filesystem and MM Summit.

REGISTER TODAY

http://go.linuxfoundation.org/vault2015

LINUX FOUNDATION

What Else?

For me, having a tiny Android device that lets me sync audiobooks with FolderSync and play with the Listen app is all I could want and more. That doesn't mean I couldn't think of more things to do with a cheap Android device, however. I actually bought another LG Realm when they were on sale, and I have lots of plans for it—things like:

- A Plex player for watching movies.
- A really cheap IP camera (birdcam!) using the IPWebcam app.
- Music player for the bathroom counter, connected to speakers.
- Tiny gaming device for bored children who visit.
- Cheap Skype/Hangout device to give someone I want to keep in contact with.
- Surprisingly affordable alarm clock for my nightstand.
- Universal XBMC/Kodi remote for all our televisions.
- Wi-Fi testing tool (WiFi Analyzer).

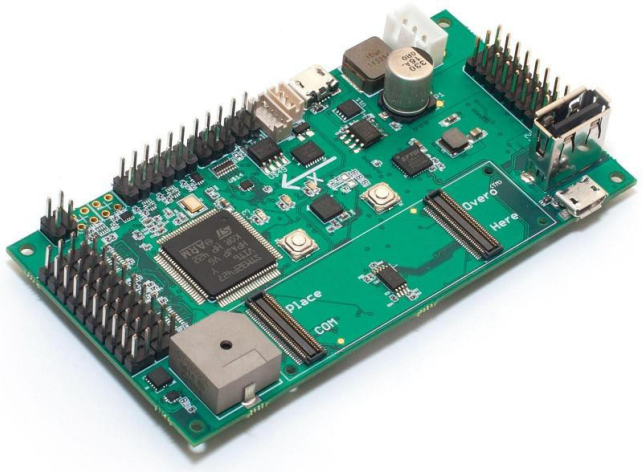
Really, I'd do anything you can do with any other Android device, but without the guilt of spending a ton of money. Having a $20 Android device capable of doing so many things really makes it easy to come up with fun projects. It's also nice to be able to mount one in the car and just leave it there. With FolderSync, it will download any new media when you're parked in the garage, and you'll never have to take your phone out of your pocket!

I'm really quite happy Samsung discontinued its Galaxy Player devices. I worry that if they were available, I might have purchased one. I'm so much happier with my $20 tablet than I would have been with a $300 media player. If you have a need for another Android device in your life, but don't want to spend a fortune, I urge you to check out the prepaid phone options out there. It's surprising what $20 will buy! ■

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.

Gumstix Inc.'s Geppetto



Gumstix Inc. is so proud of the embedded systems it designed with its home-grown Geppetto design tool that it wants the wider world to enjoy similar benefits. Gumstix calls the new Geppetto 2.0 the most advanced version of the company's on-line build-to-order tool for designing custom-embedded Linux systems. This new iteration of Geppetto introduces Tux-approved recommended mappings for buses, ensuring optimal compatibility between customer-created hardware and standard Linux images. In addition, version 2.0 offers an expanded module selection, improved dimensioning, faster UI and video tutorials. As part of the Geppetto announcement, Gumstix also announced the Geppetto-designed AeroCoreTM 2 Micro-Aerial Vehicle Control Board and the Geppetto-designed Pepper DVI-D single-board computer.
http://www.gumstix.com

Investintech.com's Able2Extract PDF Converter

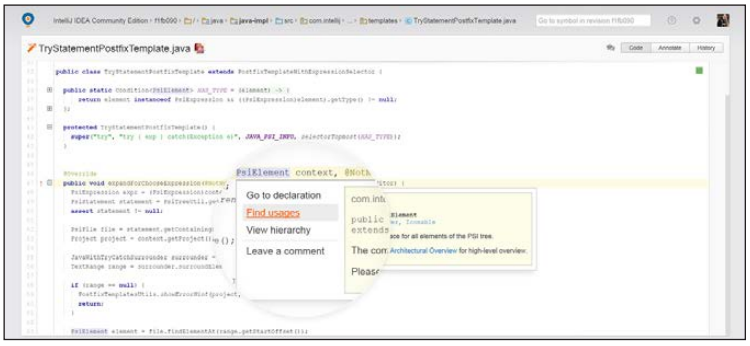
It's not a stretch to call Investintech.com's Able2Extract 9 PDF Converter the "Swiss army knife" of PDF converters. Not only is Able2Extract able to convert PDFs to a wide range of formats accurately, but it also features the unique ability to work across Linux (Ubuntu and Fedora), Mac OS X and Windows platforms. Investintech.com notes the ability of Able2Extract to maintain intact all aspects—images, colors, formatting and fonts—regardless of file format. Supported formats include converting PDF to OpenOffice.org, MS-Office, AutoCAD, Excel and commonly used image formats. The upgrade version 9 adds secure PDF creation, improved custom PDF to Excel conversion and an improved GUI and overall user experience.
http://www.investintech.com

Linutop XS

Until the era of the Linutop computer, the word minuscule has not been a common descriptor for a full-fledged PC. That word nevertheless hits the nail squarely on the head to describe the new Linutop XS, a truly tiny Linux computer designed to reduce TCO from shipping to deployment, operation and maintenance. As Linutop's smallest and most energy-efficient computer to date, the Linutop XS weighs a mere 3.3 ounces (92 g), measures about the size of a typical playing card and operates on only 5 Volts and 3 Watts. Linutop says that the Linutop XS comes loaded with Debian Weezy and ready-to-use software, including Libre Office and Linutop Kiosk, making it an ideal system for a wide range of applications in business, government, education and the home.

http://www.linutop.com

JetBrains' Upsource

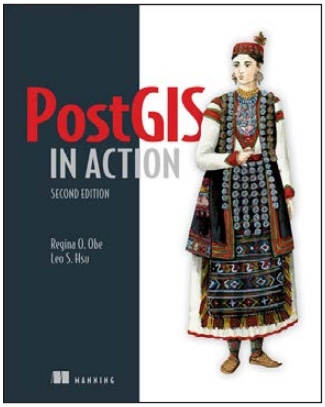


The idea for JetBrains' new team collaboration tool for developers, called Upsource, originally came from the intention to make a totally different tool, IntelliJ IDEA, available from both the desktop and the Web. The final result is Upsource 1.0, a new Web-based team collaboration tool that helps developers read, browse and review code maintained in Git, Mercurial, Subversion and/or Perforce repositories. Both a repository browser and a code-review tool, Upsource 1.0 provides instant read access to code developed throughout an organization and helps improve code quality by enabling easy code review. JetBrains adds that, thanks to platform sharing with the IntelliJ IDEA IDE for Java, Java teams enjoy an additional advantage. Upsource boasts in-depth knowledge of Java code and is able to execute server-side static code analysis on Java projects, as well as provide code-aware navigation and smart search for code usages.

http://www.jetbrains.com/upsource

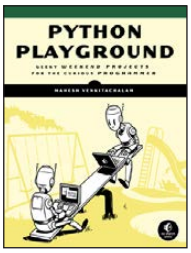
Corsair Flash Voyager Slider Series X1 and X2

The new Flash Voyager Slider X1 and X2 families of USB 3.0 Flash drives expand Corsair's already formidable arsenal of memory products. Combining the speed of USB 3.0 with the functionality of a cap-less USB drive, the Flash Voyager Slider X1 and Slider X2 series share a sleek, glossy design that allows the USB cap to slide back conveniently into the drive housing, says Corsair. The company added that the Slider X1 is available in 16GB, 32GB, 64GB, 128GB and 256GB capacities and, thanks to its USB 3.0 interface, is able to reach read speeds of up to 130MB/s. Meanwhile, Slider X2 knocks the performance up a level with read speeds of 200MB/s in capacities of 16GB, 32GB, 64GB and 128GB. Both Corsair drive families are compatible with Linux, Windows and Mac OS X, and they also are fully USB 2.0-backward compatible.
http://www.corsair.com



Regina O. Obe and Leo S. Hsu's *PostGIS in Action*, 2nd ed. (Manning Publications Co.)

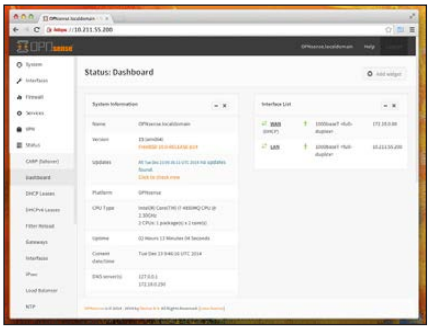
Hybrid GIS and Linux geeks know that the open-source PostGIS gives support for geographic objects to PostgreSQL, allowing the relational database to serve as the back end for ArcGIS, GRASS GIS and other geospatial programs. The new 2nd edition of *PostGIS in Action* from Regina O. Obe and Leo S. Hsu teaches readers of all levels to write spatial queries that solve real-world problems. Obe and Hsu start by getting readers' feet wet with a background in vector-, raster- and topology-based GIS, followed by a tutorial in analyzing, viewing and mapping data. Readers learn how to optimize queries for maximum speed, simplify geometries for greater efficiency, analyze rasters, vectorize rasters, better manage data utilizing topologies and create custom functions. The book covers PostGIS 2.0 and 2.1, PostgreSQL 9.1, 9.2 and 9.3 features and shows how to integrate PostGIS with other GIS tools.
http://manning.com



Mahesh Venkitachalam's *Python Playground* (No Starch Press)

Putting the subtitle *Geeky Weekend Projects for the Curious Programmer* onto a book is a sure way to charm one's way onto these geek-friendly *Linux Journal* pages. The main title of said book is *Python Playground*, a new book from Mahesh Venkitachalam and irreverent publisher No Starch Press. No Starch describes the book as "a collection of fun programming projects that will inspire you to new heights as a Pythonista". Readers will learn to use Python for all kinds of playful purposes—for example, to manipulate images, build simulations and interact with hardware using Arduino and Raspberry Pi. As readers work through each project, they power up their programming skills and learn how to leverage external libraries for specialized tasks, how to break problems into smaller, solvable pieces and how to translate an algorithm into code. The fun projects include an autostereogram generator, an ASCII art maker, a Conway's Game of Life simulator, a ray casting volume renderer and an Arduino rig.
http://www.nostarch.com

Deciso OPNsense Firewall



Deciso B.V. is a Netherlands-based manufacturer of networking equipment that developed and recently released OPNsense, a new, open-source firewall that reportedly "combines the best of open-source and closed-source firewalls". Deciso adds that OPNsense brings the rich feature set of commercial offerings with the benefits of open and verifiable sources, combined with a simple, two-clause BSD license. The latter permits companies to create a branded firewall based on OPNsense, extend its features, or even create a fork and build upon the same codebase. Key features of OPNsense include load balancing, high availability and captive portal. The easy-to-use Bootstrap-based GUI makes configuring and managing the firewall a comfortable task for administrators. The kicker, boasts Decisio, is that all sources and build tools are freely available without special clauses and without licensing costs. The company also puts a great deal of value on the community surrounding OPNsense, which it says will give users, developers and businesses a friendly, stable and transparent environment.
http://www.opnsense.org and http://www.deciso.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.

USING HIERA WITH PUPPET

A GUIDE TO USING HIERA WITH PUPPET, SEPARATING CODE FROM DATA AND ENCRYPTING PASSWORDS AND CERTIFICATES.

SCOTT LACKEY

With Hiera, you can externalize your systems' configuration data and easily understand how those values are assigned to your servers. With that data separated from your Puppet code, you then can encrypt sensitive values, such as passwords and keys.

Separating code and data can be tricky. In the case of configuration management, there is significant value in being able to design a hierarchy of data—especially one with the ability to cascade through classifications of servers and assign one or several options. This is the primary value that Hiera provides—the ability to separate the code for "how to configure the /etc/ntp.conf" from the values that define "what ntp servers should each node use". In the most concise sense, Hiera lets you separate the "how" from the "what".

The idea behind separating code and data is more than just having a cleaner Puppet environment; it allows engineers to create more re-usable Puppet modules. It also puts your variables in one place so that they too can be re-used, without importing manifests across modules. Hiera's use cases include managing packages and versions or using it as a Node Classifier. One of the most compelling use cases for Hiera is for encrypting credentials and other sensitive data, which I talk about later in this article.

Puppet node data originally was managed through node inheritance, which is no longer supported, and subsequently through using a params.pp module subclass. Before Hiera, it was necessary to modify the params.pp module class locally within the module, which frequently damaged the re-usability of the module. params.pp still is used in modules today, but as of Puppet version 3, Hiera is not only the default, but also the first place checked for variable values. When a variable is defined in both Hiera and a module, Hiera takes precedence by default. As you'll see, it's easy to use a module with params.pp and store some or all of the variable data in Hiera, making it easy to migrate incrementally.

To get started using Hiera with your existing Puppet 3 implementation, you won't have to make any significant changes or code migrations. You need only a hierarchy file for Hiera and a yaml file with a key/value pair. Here is an example of a Hiera hierarchy:

```
hiera.yaml:

:backends:
    - yaml
:yaml:
```

```
  :datadir: /etc/puppet/hieradata
:hierarchy:
  - "node/%{::fqdn}"
  - "environment/%{::env}/main"
  - "environment/%{::env}/%{calling_module}"
  - defaults
```

And a yaml file:

```
/etc/puppet/hieradata/environment/prod/main.yaml:
---
$nginx::credentials::basic_auth: 'password'
```

Hiera can have multiple back ends, but for now, let's start with yaml, which is the default and requires no additional software. The `:datadir:` is just the path to where the hierarchy search path should begin, and is usually a place within your Puppet configuration. The `:hierarchy:` section is where the core algorithm of how Hiera does its key/value lookups is defined. The `:hierarchy:` is something that will grow and change over time, and it may become much more complex than this example.

Within each of the paths defined in the `:hierarchy:`, you can reference any Puppet variable, even `$operatingsystem` and `$ipaddress`, if set. Using the `%{variable}` syntax will pull the value.

This example is actually a special hierarchical design that I use and recommend, which employs a fact assigned to all nodes called `@env` from within facter. This `@env` value can be set on the hosts either based on FQDN or tags in EC2 or elsewhere, but the important thing is that this is the separation of one large main.yaml file into directories named prod, dev and so on, and, therefore, the initial separation of Hiera values into categories.

The second component of this specific example is a special Hiera variable called `%{calling_module}`. This variable is unique and reserved for Hiera to indicate that the yaml filename to search will be the same as the Puppet module that is performing the Hiera lookup. Therefore, the way this hierarchy will behave when looking for a variable in Puppet is like:

```
$nginx::credentials::basic_auth
```

First, Hiera knows that it's looking in /etc/puppet/hieradata/node for a file named <hostname.domain.tld>.yaml and for a value for `nginx::credentials::basic_auth`. If either the file or the variable isn't there, the next step is to look in /etc/puppet/hieradata/environment/<prod|stage|dev>/main.yaml, which is a great way to have one yaml file with most

of your Hiera values. If you have a lot of values for the nginx example and you want to separate them for manageability, you simply can move them to the /etc/puppet/hieradata/environment/<prod|stage|dev>/nginx.yaml file. Finally, as a default, Hiera will check for the value in defaults.yaml at the top of the hieradata directory.

Your Puppet manifest for this lookup should look something like this:

```
modules/nginx/manifests/credentials.pp


class nginx::credentials (
  basic_auth = 'some_default',
){}
```

This class, when included, will pull the value from Hiera and can be used whenever included in your manifests. The value set here of `some_default` is just a placeholder; Hiera will override anything set in a parameterized class. In fact, if you have a class you are thinking about converting to pull data from Hiera, just start by moving one variable from the class definition in {} to a parameterized section in (), and Puppet will perform a Hiera lookup on that variable. You even can leave the existing definition intact, because Hiera will override it. This kind of Hiera lookup is called Automatic Parameter Lookup and is one of several ways to pull data from Hiera, but it's by far the most common in practice. You also can specify a Hiera lookup with:

```
modules/nginx/manifests/credentials.pp


class nginx::credentials (
  basic_auth = hiera('nginx::credentials::basic_auth'),
){}
```

These will both default to a priority lookup method in the Hiera data files. This means that Hiera will return the value of the first match and stop looking further. This is usually the only behavior you want, and it's a reasonable default. There are two lookup methods worth mentioning: `hiera_array` and `hiera_hash`. `hiera_array` will find all of the matching values in the files of the hierarchy and combine them in an array. In the example hierarchy, this would enable you to look up all values for a single key for both the node and the environment—for example, adding an additional DNS search path for one host's /etc/resolv.conf. To use a `hiera_array` lookup, you must define the lookup type explicitly

(instead of relying on Automatic Parameter Lookup):

modules/nginx/manifests/credentials.pp

```
class nginx::credentials (
  basic_auth = hiera_array('nginx::credentials::basic_auth'),
){}
```

A `hiera_hash` lookup works in the same way, only it gathers all matching values into a single hash and returns that hash. This is often useful for an advanced `create_resources` variable import as well as many other uses in an advanced Puppet environment.

Perhaps Hiera's most powerful feature is the ability to pull data from a variety of back-end storage technologies. Hiera back ends are too numerous to list, but they include JSON, Redis, MongoDB and even HTTP to create a URL-driven Puppet value API. Let's take a look at two useful back ends: Postgres and hiera-eyaml.

To start with the psql back end, you need to install the hiera-psql gem on your Puppet master (or each node if you're using masterless Puppet runs with Puppet apply), with a simple hiera.yaml file of:

```
:hierarchy:
* 'environment/%{env}'
* default
:backends:
* psql
:psql:
  :connection:
    :dbname: hiera
    :host: localhost
    :user: root
    :password: password
```

You can do lookups on a local Postgres installation with a single database called hiera with a single table called config with three columns: Path, Key and Value.

```
path                 key                            value

'environment/prod'   'nginx::credentials::basic_auth'   'password'
```

This is extremely useful if you want to expose your Hiera data to custom in-house applications outside Puppet, or if you want to create a DevOps Web console or reports.

Storing credentials in Puppet modules is a bad idea. If you store credentials in Puppet and your manifests on an external code repository, you're not only unable to share those manifests with developers with less-secure access, but you're obviously exposing vital security data outside the organization, and possibly in violation of various types

of compliance. So how do you encrypt sensitive data in Puppet while keeping your manifests relevant and sharable? The answer is with hiera-eyaml.

Tom Poulton created hiera-eyaml to allow engineers to do just that: encrypt only the sensitive string of data inside the actual file rather than encrypting the entire file, which also can be done with hiera-gpg (a very useful encryption gem but not covered in this article).

To get started, install the hiera-eyaml gem, and generate a keypair on the Puppet master:

```
$ eyaml createkeys
```

Then move the keys to a secure location, like /etc/puppet/secure/keys. Your hiera.yaml configuration should look something like this:

```
hiera.yaml:
---
:backends:
  - eyaml
  - yaml
:yaml:
  :datadir: /etc/puppet/hieradata
:eyaml:
  :datadir: /etc/puppet/hieradata
  :extension: 'yaml'          # <- so all files can be named .yaml
  :pkcs7_private_key: /path/to/private_key.pkcs7.pem
  :pkcs7_public_key: /path/to/public_key.pkcs7.pem
:hierarchy:
  - "node/%{::fqdn}"
  - "environment/%{::env}/main"
  - "environment/%{::env}/%{calling_module}"
* defaults
```

To encrypt values, you need only the public key, so distribute it to anyone who needs to create encrypted values:

```
$ eyaml encrypt -s 'password'
```

This will generate an encrypted block that you can add as the value in any yaml file:

```
main.yaml:
nginx::credentials::user: slackey       #cleartext example value
nginx::credentials::basic_auth : >      #encrypted example value
ENC[PKCS7,Y22exl+OvjDe+drmik2XEeD3VQtl1uZJXFFF2Nn
   /HjZFXwcXRtTlzewJLc+/gox2IfByQRhsI/AgogRfYQKocZg
   IZGeunzwhqfmEtGiqpvJJQ5wVRdzJVpTnANBA5qxeA==]
```

Editing encrypted values in place is one of the coolest features of the hiera-eyaml back end. `eyaml edit` opens a copy of the eyaml file in your editor of choice and automatically decrypts all of the values in the file. Here you can modify the values just as though they were plain text. When you exit the editor by saving the file, it automatically encrypts all of the modified values and saves the new file in place. You can see that the

unencrypted plain text is marked to allow the eyaml tool to identify each encrypted block, along with the encryption method that originally was used. This is used to make sure that the block is encrypted again only if the clear text value has changed and is encrypted using the original encryption mechanism:

```
nginx::credentials::user: user1

nginx::credentials::basic_auth : DEC(1)::PKCS7[very secret password]!
```

Blocks and strings of encrypted text can get rather onerous once you have more than a hundred entries or so. Because these yaml files are meant to be modified by humans directly, you want them to be easy to navigate. In my experience, it makes sense to keep your encrypted values in a separate file, such as a secure.yaml, with a hierarchy path of:

```
:hierarchy:
  - "node/%{::fqdn}"
 - "environment/%{::env}/secure"
  - "environment/%{::env}/main"
  - "environment/%{::env}/%{calling_module}"
```

This isn't necessary, as each value is encrypted individually and can be distributed safely to other teams. It may work well for your environment, however, because you can store the encrypted files in a separate repository, perhaps in a different Git repository. Only the private keys need to be protected on the Puppet master. I also recommend having separate keys for each environment, as this can give more granular control over who can decrypt different datafiles in Hiera, as well as even greater security separation. One way to do this is to name the keys with the possible values for the @env fact, and include that in the path of the hierarchy. You'll need to encrypt values with the correct key, and this naming convention makes it easy to tell which one is correct:

```
:pkcs7_private_key: /path/to/private_key.pkcs7.pem-%{::env}

:pkcs7_public_key: /path/to/public_key.pkcs7.pem-%{::env}
```

When using Hiera values within Puppet templates, either encrypted or not, you must be careful to pull them into the class that contains the templates instead of calling the values from within the template across classes—for example, in the template mytest.erb in a module called mymodule:

```
mytest.erb:

...

username: user1

passwd: <%= scope.lookupvar('nginx::credentials::basic_auth') %>

➥#don't do this

...
```

LinuxFest Northwest

April 25th & 26th

Bellingham, WA

- All things Open Source
- 40+ Exhibitors
- 80+ Presentations
- 1500+ Attendees
- Prizes and after party
- FREE admission & parking
- Bring the whole family!

Hosted By

Bellingham TECHNICAL COLLEGE

linuxfestnorthwest.org

Puppet may not have loaded a value into `nginx::credentials::basic_auth` yet because of the order of operations. Also, if you are using the `%calling_module` Hiera variable, the calling module in this case would be mymodule, and not nginx, so it would not find the value in the nginx.yaml file, as one might expect.

To avoid these and other issues, it's best to import the values into the mymodule class and assign local values:

```
mymodule.pp:
class mymodule {
  include nginx::credentials
  $basic_auth = "${nginx::credentials::basic_auth}"
  file { '/etc/credentials/boto_cloudwatch.cfg':
     content =>  template ("mymodule/mytest.erb"),
  }
```

And then reference the local value from the template:

```
mytest.erb:
...
username: user1
passwd: <%= @basic_auth %>
```

You're now ready to start introducing encrypted Hiera values gradually into your Puppet environment. Maybe after you separate data from your Puppet code, you can contribute some of your modules to the PuppetForge for others to use!■

Scott Lackey is a 17-year engineering veteran and Sr. DevOps Engineer for Salesforce.com. He's passionate about helping companies migrate to the cloud and mentoring prospective DevOps engineers. He lives in Los Angeles with his dachshund Zelda. Reach him at sudosudash@gmail.com.

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

Resources

Docs—Hiera 1 Overview: **https://docs.puppetlabs.com/hiera/1**

"First Look: Installing and Using Hiera": **http://puppetlabs.com/blog/first-look-installing-and-using-hiera**

TomPoulton/hiera-eyaml: **https://github.com/TomPoulton/hiera-eyaml**

dalen/hiera-psql: **https://github.com/dalen/hiera-psql**

"Encrypting sensitive data in Puppet": **http://www.theguardian.com/info/developer-blog/2014/feb/14/encrypting-sensitive-data-in-puppet**

INITIALIZING AND MANAGING SERVICES IN LINUX: PAST, PRESENT AND FUTURE

systemd is the new init system used by many of the top Linux distributions, but do you know the history behind it and how we got here? Learn about the history of init systems in Linux and their UNIX legacy. Gain a better perspective about how Linux manages services and other support processes.

Jonas Gorauskas

One of the most crucial pieces of any UNIX-like operating system is the init dæmon process. In Linux, this process is started by the kernel, and it's the first userspace process to spawn and the last one to die during shutdown.

During the history of UNIX and Linux, many init systems have gained popularity and then faded away. In this article, I focus on the history of the init system as it relates to Linux, and I talk about the role of init in a modern Linux system. I also relate some of the history of the System V Init (SysV) scheme, which was the de facto standard for many Linux distributions for a long time. Then I cover a couple more modern approaches to system initialization, such as Upstart and systemd. Finally, I pay some attention to how things work in systemd, as this seems to be the popular choice at the moment for several of the largest distributions.

The Role of Init

Init is short for initializer, and it's both a startup manager and a session manager for Linux and other UNIXes. It's a startup manager, because it plays a crucial role in the startup of Linux. It's the process that creates or initializes userspace and, ultimately, all userspace processes. It also may be considered a session manager, because it takes care of many aspects of userspace and its processes once the system is up and running.

The call to start this process is, in fact, hard-coded in the Linux kernel. Download the latest kernel sources and look for a function called kernel_init in the file init/main.c. Among the files that the Linux kernel will try to execute is /sbin/init. If Linux cannot find one of these processes, it throws a kernel panic and halts.

The kernel gives the init process an ID of 1 or PID 1. All other userspace processes are forked from init, and therefore, PID 1 claims ancestral rights to all other userspace processes. PID 1 also automatically will become the direct parent process of any userspace process that is orphaned.

A Little Bit of History

Now that I have set the stage for the article and given you a very basic understanding of what init is and does, I'd like to digress into a little bit of UNIX history.

There has been a lot of diversity in the initialization schemes for UNIX-like operating systems over time. Two of the most important init schemes that had a historical impact on how different Linux distributions do things are the rc scheme used in the 4.4 BSD

A Linux distribution implementing a SysV scheme can be in one of many distinct states in which a predetermined number of processes may be running.

and the SysV scheme used in SunOS and Solaris.

The 4.4 BSD init system is pretty simple and monolithic. When booting, the kernel runs /sbin/init, which would spawn a shell to run the /etc/rc script. The /etc/rc script contained commands to check the integrity of hard drives and mount them, start other processes, and start the networking subsystem. This scheme was contained completely within a few scripts: namely /etc/rc, /etc/rc.local and /etc/netstart. This scheme also had no specific shutdown procedure. Init would receive a SIGTERM signal and send a SIGHUP and/or a SIGTERM to its children, and after all processes exited, it would drop to single-user mode and shut down.

Today, the systems that have inherited the rc initialization scheme are Free-BSD, Net-BSD and the Slackware Linux distribution. These modern systems have improved quite a bit on the original 4.4 BSD scheme and are much more modular than the original.

Most other Linux distributions have, historically, been adepts of the SysV scheme, which originally was implemented in AT&T UNIX and derivative systems like Solaris.

System V Init

A Linux distribution implementing a SysV scheme can be in one of many distinct states in which a predetermined number of processes may be running. These states are called runlevels and to get to a certain runlevel means that the system is in a certain operational stage.

The meaning for each runlevel may vary based on your distribution of Linux. For example, there are a few distributions (such as Ubuntu) that use runlevel 2 to mean multi-user graphical mode with networking enabled. Others (like Fedora) use runlevel 5 to mean the same thing.

In a SysV Linux machine, the kernel runs /sbin/init as usual, which in turn will load parameters and execute

directives defined in /etc/inittab. This file defines the default runlevel for the whole system, describes what happens when Ctrl-Alt-Del is pressed, loads keymap files, defines which terminals to spawn gettys for, spawns terminal login processes, runs the /etc/init.d/rcS script, and it also influences the order of execution of other runlevel scripts.

The /etc/init.d/rcS script will put the system in a single-user mode in order to finish probing hardware, mount disks, set hostname, set up networking and so on. Take a look at /etc/rcS.d/ in a Debian 7 system for all the gory details. Next, /sbin/init will switch itself to the default runlevel to start all the system services. The default runlevel value is defined in the `initdefault` line of /etc/inittab.

This actually translates into a call to the /etc/init.d/rc script with the parameter of 2 for the runlevel value. The rc script will then execute all of the K* (for Kill) and S* (for Start) scripts in the /etc/rc2.d/ directory. These are actually links to the real scripts in /etc/init.d/. The names of the links follow the format `S##<service-name>` or `K##<service-name>`, where the ## token is the two-digit number used to determine the order in which the script should run. Order is alphabetical, and Kill scripts execute before Start scripts. The last thing to happen is to run the /etc/rc.local script, which is where you can add custom system commands that you want to execute at startup.

A system that uses the SysV scheme usually comes with the service program used to manage the services while the system is running. You can check on the status of a service, or all services, and start or stop a service, respectively, using the `service` utility:

- `$ service <service> status`
- `$ service status -all`
- `# service <service> start|stop`

To manage the assignment of services to a particular runlevel, you can use a tool called `sysv-rc-conf`, which manages the setup of all links in the respective rc directories. You also can switch the runlevel of the system at any time when you use the command `telinit` as a privileged user. For example, `telinit 6` will reboot a SysV system.

The SysV scheme still is in use today in Debian 7 (Wheezy) systems. However, the Debian developers will be changing the init system in version 8 to systemd. I cover systemd in more

The SysV scheme has been great, but it started to show its age around the time when Linux on the desktop gained a little more momentum.

detail below, but first, let's look at why we need a new init system.

The Problem with System V Init

The SysV scheme has been great, but it started to show its age around the time when Linux on the desktop gained a little more momentum. When the SysV scheme originally was designed, computers where nothing like they are today. SysV was not designed to handle certain things well:

- USB devices.
- External storage volumes.
- Bluetooth devices.
- The cloud.

The SysV scheme was designed for a world that was static and slow moving. This init scheme originally was responsible only for bringing the system into a normal running state after power on or gracefully shutting down services prior to shutdown. As a result, the design was strictly synchronous, blocking future tasks until the current one had completed.

This left the system unable to handle various events that were not related to the startup or shutdown of the system. Things that we take for granted today were really cumbersome to handle elegantly during the heyday of SysV init:

- There was no real process supervision—for example, dæmons were not automatically restarted when they crashed.
- There was no real dependency checking. The order of script naming determined the order in which they were loaded.
- The addition or removal of USB drives and other portable storage/network devices while the machine was running was cumbersome and oftentimes required a reboot.

- There were no facilities to discover and scan for new storage devices without locking the system, especially when a disk might not even power on until it was scanned.

- There were no facilities to load firmware for a device, which may have needed to occur after it was detected but before it was usable.

Inevitably, around the 2005/2006 time frame, several alternative efforts tried to fix all the issues with the SysV scheme. But the effort that looked most promising during that time was the Upstart init project sponsored by Canonical.

Upstart

To be sure, Upstart init doesn't share any code with the SysV init scheme, but it's rather a superset of it, providing a good degree of backward-compatibility. The main departure from the traditional SysV way of doing things is that Upstart implements an event-driven model that allows it to respond to milestones asynchronously as they are reached. Upstart also implements the concept of jobs, which are described by the files under /etc/init/*.conf, and whose purpose is to execute a script section that spawns a process. As such, system initialization can be expressed as a consecutive set of "spawn process X when event Y occurs" rules.

Just like in the SysV scheme, the Linux kernel gives control to Upstart when it executes the Upstart implementation of /sbin/init. At this point, things may work a little differently depending on your distribution of Linux. For Red Hat Enterprise Linux (RHEL) 6 users, you'll still find a file at /etc/inittab, but the sole function of this file is to set the default runlevel for the system. If your distribution is one of the Ubuntu derivatives, /etc/inittab doesn't even exist anymore, and the default runlevel is set in a file called /etc/init/rc-sysinit.conf instead.

The Upstart version of /sbin/init will emit a single event called startup, which triggers the rest of the system initialization. There are a few jobs that specify the startup event as their start condition, the most notable of which is `mountall`, which mounts all filesystems. The `mountall` job then triggers various other events related to disk and filesystem initialization. These events, in turn, trigger the udev kernel device manager to start, and it emits the event that starts the networking subsystem.

This is when one of the most critical

jobs is triggered by Upstart. This job is called `rc-sysinit`, which has a start dependency on the filesystem and network-up events. The role of this job is to bring the system to its default runlevel. It executes the command `telinit <runlevel>` to achieve this. The `telinit` command then emits the runlevel event, which causes many other jobs to start. This includes the /etc/init/rc.conf job, which implements a compatibility layer for the SysV init scheme. It executes `/etc/init.d/rc <runlevel>` and determines if a /etc/rc#.d/ directory exists for the current runlevel, executing all scripts in it.

In Upstart-based systems, such as Ubuntu and RHEL 6, you can use the tools `sysv-rc-conf` or `chkconfig`, respectively, to manage the runlevel of different services. You also can manage jobs via the `initctl` utility. You can list all jobs and their respective start and stop events with the command `initctl show-config`. You also can check on job status, list available jobs and start/stop jobs with the following commands, respectively:

- `$ initctl status <job>`
- `$ initctl list`
- `# initctl start|stop <job>`

The Upstart scheme has been used in popular distributions of Linux, such as Fedora from versions 9 up to 14, the RHEL 6 series and Ubuntu since version 6.10 to present. But for all the flexibility that Upstart init brings to Linux, it still falls short in a few fundamental ways:

- It ignores the system state between events. For instance, a system has a power cord plugged in, then the system runs on AC power for a while, and then the user unplugs the power cord. Upstart focuses on each event above as a single discrete and unrelated unit, instead of tracking the chain of events as a whole.

- The event-driven nature of the system turns the dependency chain on its head. Instead of doing the absolute minimum amount of work needed to get the system to a working state, when an event is triggered, it executes all jobs that could possibly follow it. For example, just because networking has started, it doesn't mean that NFS also should start. As a matter of fact, the opposite is the correct order of things: when a user requests access to an NFS share, the system should validate that networking is also up and running.

The main design goals of this init scheme are, according to Lennart Poettering, lead developer of systemd, "to start less, and to start more in parallel".

- The dependency chain is still present. Although many more things happen in parallel in Upstart, the user has to port the original script sequence from SysV init to a set of event trigger action rules in the *.conf files in /etc/init. Furthermore, because of the spanning tree structure of the event system, it is a real nightmare to figure out why something happened and what event triggered it.

There is another init scheme whose purpose is to address the issues listed above.

systemd

systemd is the latest milestone on the road to init system nirvana. The main design goals of this init scheme are, according to Lennart Poettering, lead developer of systemd, "to start less, and to start more in parallel". What that means is that you execute only that which is absolutely necessary to get the system to a running state, and you run as much as possible at the same time.

To accomplish these goals, systemd aims to act against two major trouble spots of previous init schemes: the shell and parallelism. The main executable for systemd, /lib/systemd/systemd, performs all calls that originally were present in scripts, thus eliminating the need to spawn a shell environment. What about the call to /sbin/init that's hard-coded in the Linux kernel? It's still there in the form of a symbolic link to /lib/systemd/systemd.

To address parallelism, you need to remove the dependency chain between the various services or at least make it a secondary concern. If you look at the problem at its most fundamental level, the dependency between the various services boils down to one thing: having a socket available for the processes to communicate among themselves. systemd creates all sockets first and then spawns all processes in parallel. For example, services that need to write to the system log need to wait for the

/dev/log socket to become available, but as soon as it is available, these services can start. Therefore, if systemd creates the socket /dev/log first, then that's one less dependency that blocks other services. Even if there is nothing to receive messages at the other end of the socket, this strategy still works. The kernel itself will manage a buffer for the socket, and as soon as the receiving service starts, it will flush the buffer and handle all the messages. The ideas above are not new or revolutionary. They have been tried before in projects like the xinetd superserver and the launchd init scheme used in OS X.

systemd does introduce the new concepts of units and targets. A target is analogous to a runlevel in previous schemes and is composed of several units. systemd will execute units to reach a target. The instructions for each unit reside in the /lib/systemd/system/ directory. These files use a declarative format that looks like a Windows INI file. The most common type of these units is the service unit, which is used to start a service. The sshd.service file from Arch Linux looks like this:

```
[Unit]
Description=OpenSSH Daemon
Wants=sshdgenkeys.service
After=sshdgenkeys.service
After=network.target

[Service]
ExecStart=/usr/bin/sshd -D
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process
Restart=always

[Install]
WantedBy=multi-user.target
```

This format is really simple and really portable across several different distributions. There are other types of unit files that describe a system, and they are socket, device, mount, automount, swap, target, path, timer, snapshot, slice and scope. Going into all of them in detail is beyond the scope of this article; however, I want to mention one thing: target is a special type of unit file that glues the other types together into a cohesive whole. For example, here are the contents of basic.target from Arch Linux:

```
[Unit]
Description=Basic System
Documentation=man:systemd.special(7)
Requires=sysinit.target
Wants=sockets.target timers.target paths.target
 ➥slices.target
After=sysinit.target sockets.target timers.target
 ➥paths.target slices.target
JobTimeoutSec=15min
JobTimeoutAction=poweroff-force
```

You can follow the chain of dependencies if you look at what basic.target requires and wants. Those are actual unit files in the same /lib/systemd/system/ directory. The `Requires` and `Wants` directives above are how systemd defines the dependency chain among the units. The `Requires` directive denotes a hard requirement, and `Wants` denotes an optional requirement. Also keep in mind that `Requires` and `Wants` don't imply order. If the `After` directive isn't specified, systemd will start the units in parallel.

Timer units are also really interesting. They are unit files that contain a `[Timer]` section and define how the `TimeDateD` subsystem of systemd will activate a future event. In these timer units, you can create two types of timers: one that will activate after a time period based on a variable starting point, such as the systems boot, and another that activates at fixed intervals like a cron job. As a matter of fact, timer units are an alternative to cron jobs.

One last thing to mention about systemd unit files is that they provide the means to describe easily what to do when a service crashes. You can do that by using the directives `Restart` or `RestartSec` in your unit files. This feature allows systemd to take the role of process supervisor as well.

systemd refers to the init dæmon executable itself, namely /lib/systemd/systemd, but it also refers to the set of utilities and programs used to manage the system and services. Chief among these utilities is the `systemctl` program that's used to manage services. You can use it to enable, start and disable services, find the status of a given service and also list all loaded units. For example:

- `# systemctl enable sshd`
- `# systemctl start sshd`
- `# systemctl stop sshd`
- `# systemctl status sshd`
- `# systemctl list-units`

Some Linux distributions, like RHEL 7 and CentOS 7, provide a compatibility layer that translates SysV and Upstart commands into systemd commands. If you issue the command `service sshd status` in CentOS 7, you will get the following output:

```
Redirecting to /bin/systemctl status  sshd.service
sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled)
   Active: active (running) since Mon 2014-12-08 02:01:53 PST;
➥12h ago
  Process: 915 ExecStartPre=/usr/sbin/sshd-keygen (code=exited,
➥status=0/SUCCESS)
     Main PID: 937 (sshd)
       CGroup: /system.slice/sshd.service
               ...937 /usr/sbin/sshd -D
```

Notice that first line of console output above and how it indicates that the SysV-style command was redirected to the systemd-style of command. This allows the user to ease into the systemd way of doing things while still allowing the user to leverage the previous skill set.

Another really important program in the systemd toolbox is the `journalctl` utility. It allows you to view and manage the systemd logging subsystem called journald. systemd's logfile is a binary file and using `journalctl` really simplifies the user experience. Here are some interesting examples:

- Display full log: `# journalctl --all`
- Tail the log: `# journalctl -f`
- Filter log by executable:
  `# journalctl /lib/systemd/systemd`
- Display log since last boot:
  `# journalctl -b`
- Display errors from last boot:
  `# journalctl -b -p err`

I urge you to look at the documentation of the different schemes presented here to learn more.

Controversies

From my vantage point, the future is not 100% certain when it comes to init schemes for Linux. The clear leader, as I write this in late 2014, is systemd. A lot of distributions are adopting it; the latest ones are RHEL 7 and Debian 8. However, the adoption of systemd has been controversial, and these distributions have received a lot of strong feedback from their respective communities. Of note is the Debian technical committee debate that occurred in the Debian mailing list and a complaint by Linus Torvalds himself in the Linux kernel mailing list.

systemd is not just an init scheme. It unifies everything that is related to starting and managing system services into a centralized and monolithic whole: user login, cron jobs, network services, virtual TTY management and so on. The use of shell scripts to control system startup has the benefit of providing flexibility, and a lot of

members of the community want to be able to choose their favorite init scheme. This has spawned some forks of systemd and even a faction of the Linux community that is for completely boycotting systemd. Check out the site http://boycottsystemd.org.

Conclusion

The userspace initialization and management of Linux systems has had a rich and diverse history. I hope that this article has given you a new perspective for how we got to where we are today with systemd becoming the new standard. I have covered all the pros and cons of the different schemes and how those factors have influenced user choice in this space over time. I hope this article will foster further discussion, and your feedback is highly encouraged.■

Jonas Gorauskas is technically a software developer by trade but also a generalist with background in operations. In the past he has been one or more of the following: programmer, technical support analyst, technical writer, systems designer, database administrator, amateur cook and professional curmudgeon. Jonas is currently working at Intuit in Reno, Nevada, as part of the Application Operations Engineering team helping them with operations, deployment, DevOps or anything else they can think of.

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

Resources

The source code of various Linux distributions, including:

- Debian 7 and 8
- CentOS 6.5 and 7
- Slackware 14
- Fedora 20
- Ubuntu 12.4 and 14.4
- Arch Linux

The Web site of Lennart Poettering: **http://0pointer.net/blog**

The systemd Documentation: **http://freedesktop.org/wiki/Software/systemd**

Upstart Documentation: **http://upstart.ubuntu.com/cookbook**

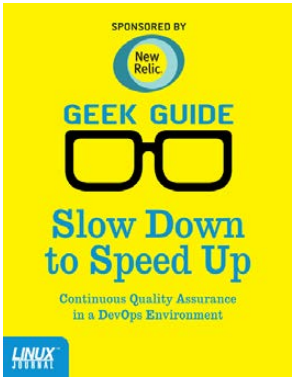
NEW!

Linux Journal eBook Series

GEEK GUIDES

FREE Download NOW!

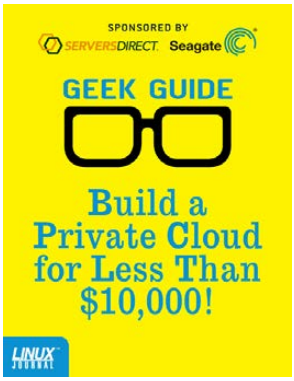
Slow Down to Speed Up:
Continuous Quality Assurance in a DevOps Environment



By Bill Childers

DevOps is one of the newest and largest movements in Information Technology in the past few years. The name DevOps is a portmanteau of "Development" and "Operations" and is meant to denote a fusion of these two functions in a company. Whether or not your business actually does combine the two functions, the lessons and tools learned from the DevOps movement and attitude can be applied throughout the entire Information Technology space. This eBook focuses on one of the key attributes of the DevOps movement: Quality Assurance. At any point, you should be able to release your product, code or configuration—so long as you continue keeping your deliverables in a deployable state. This is done by "slowing down" to include a Quality Assurance step at each point in your workflow. The sooner you catch an error or trouble condition and fix it, the faster you can get back on track. This will lower the amount of rework required and keep your team's momentum going in a forward direction, enabling your group to move on to new projects and challenges.

Build a Private Cloud for Less Than $10,000!



By Mike Diehl

This eBook presents a compelling argument as to why you should consider re-architecting your enterprise toward a private cloud. It outlines some of the design considerations that you need to be aware of before implementing your own private cloud, and it describes using the DevCloud installer in order to install OpenStack on an Ubuntu 14 server. Finally, this eBook will familiarize you with the features and day-to-day operations of an OpenStack-based private cloud architecture, all for less than $10K!

DOWNLOAD NOW AT: http://linuxjournal.com/geekguides

Infinite BusyBox with systemd

Lightweight virtual containers with PID 1.

Charles Fisher

In this article, I demonstrate a method to build one Linux system within another using the latest utilities within the systemd suite of management tools. The guest OS container design focuses upon BusyBox and Dropbear for the userspace system utilities, but I also work through methods for running more general application software so the containers are actually useful.

This tutorial was developed on Oracle Linux 7, and it likely will run unchanged on its common brethren (Red Hat, CentOS, Scientific Linux), and from here forward, I refer to this platform simply as V7. Slight changes may be necessary on other systemd platforms (such as SUSE, Debian or Ubuntu). Oracle's V7 runs only on the x86_64 platform, so that's this article's primary focus.

Required Utilities

Red Hat saw fit to remove the long-included BusyBox binary from its V7 distribution, but this easily is remedied by downloading the latest binary directly from the project's Web site. Since the /home filesystem gets a large amount of space by default when installing V7, let's put it there for now. Run the commands below as root until indicated otherwise:

```
cd /home
wget http://busybox.net/downloads/binaries/latest/busybox-x86_64
```

You also can get a binary copy of the Dropbear SSH server and client from this location:

```
wget http://landley.net/aboriginal/downloads/
➥binaries/extras/dropbearmulti-x86_64
```

For this article, I used the following versions:

- BusyBox v1.21.1.
- Dropbear SSH multi-purpose v2014.63.

These are static binaries that do not link against shared objects—nothing else is required to run them, and they are ideal for building a new UNIX-ish environment quickly.

Build a chroot

The chroot system call and the associated shell utility allow an arbitrary subdirectory somewhere on the system to be declared as the root for all child processes. The commands below populate the "chroot jail", then lock you in. Note that the call to chroot needs your change to the SHELL environment

BusyBox changes its behavior depending upon how it is called—it bundles a whole system of utility programs into one convenient package.

variable below, as you don't have bash inside the jail (and it's likely the default value of $SHELL):

```
export SHELL=/bin/sh
mkdir /home/nifty
mkdir /home/nifty/bin
cd /home/nifty/bin
cp /home/busybox-x86_64 /home/dropbearmulti-x86_64 .
chmod 755 busybox-x86_64 dropbearmulti-x86_64
./busybox-x86_64 --list | awk '{print "ln -s
 ➥busybox-x86_64 " $0}' | sh
chroot /home/nifty
export PATH=/bin
ls -l
###(try some commands)
exit
```

Take some time to explore your shell environment after you launch your chroot above before you exit. Notice that you have a /bin directory, which is populated by soft links that resolve to the BusyBox binary. BusyBox changes its behavior depending upon how it is called—it bundles a whole system of utility programs into one convenient package.

Try a few additional UNIX commands that you may know. Some that work are `vi`, `uname`, `uptime` and (of course) the shell that you are working inside. Commands that don't work include `ps`, `top` and `netstat`. They fail because they require the /proc directory (which is dynamically provided by the Linux kernel)—it has not been mounted within the jail.

Note that few native utilities will run in the chroot without moving many dependent libraries (objects). You might try copying bash or gawk into the jail, but you won't be able to run them (yet). In this regard, BusyBox is ideal, as it depends upon nothing.

Build a Minimal UNIX System and Launch It

The systemd suite includes the eponymous program that runs as PID 1 on Linux. Among many other utilities, it also includes the `nspawn` program that is used to

launch containers. Containers that are created by `nspawn` fix most of the problems with chroot jails. They provide /proc, /dev, /run and otherwise equip the child environment with a more capable runtime.

Next, you are going to configure a getty to run on the console of the container that you can use to log in. Being sure that you have exited your chroot from the previous step, run the following commands as root:

```
mkdir /home/nifty/etc
mkdir /home/nifty/root
echo 'NAME="nifty busybox container"' >
 ➥/home/nifty/etc/os-release
cd /home/nifty
ln -s bin sbin
ln -s bin usr/bin
echo 'root::0:0:root:/root:/bin/sh' >
 ➥/home/nifty/etc/passwd
echo 'console::respawn:/bin/getty 38400 /dev/console' >
 ➥/home/nifty/etc/inittab
tar cf - /usr/share/zoneinfo | (cd /home/nifty; tar xvpf -)
systemd-nspawn -bD /home/nifty
```

After you have executed the `nspawn` above, you will be presented with a "nifty login" prompt. Log in as root (there is no password—yet), and try a few more commands. You immediately will notice that `ps` and `top` work, and there is now a /proc.

You also will notice that the processes that appear in the child container also appear on the host system, but different PIDs will be assigned between the parent and child.

Note that you'll also receive the message: "The kernel auditing subsystem is known to be incompatible with containers. Please make sure to turn off auditing with 'audit=0' on the kernel command line before using systemd-nspawn. Sleeping for 5s..." The audit settings don't seem to impact the BusyBox container login, but you can adjust your kernel command line in your grub configuration (at least to silence the warning and stop the delay).

Running Dropbear SSH in Your Container

It's best if you configure a non-root user of your system and forbid network root logins. The reasoning will become clear when I address container security.

Run all of these commands as root within the container:

```
cd /bin
ln -s dropbearmulti-x86_64 dropbear
ln -s dropbearmulti-x86_64 ssh
ln -s dropbearmulti-x86_64 scp
ln -s dropbearmulti-x86_64 dropbearkey
ln -s dropbearmulti-x86_64 dropbearconvert
```

Above, you have established the names that you need to call Dropbear, both the main client and server, and the sundry key generation and management utilities.

You then generate the host keys that will be used by this container, placing them in a new directory /home/nifty/etc/dropbear (as viewed by the host):

```
mkdir /etc/dropbear
dropbearkey -t rsa -f /etc/dropbear/dropbear_rsa_host_key
dropbearkey -t dss -f /etc/dropbear/dropbear_dss_host_key
dropbearkey -t ecdsa -f /etc/dropbear/dropbear_ecdsa_host_key
```

Various directories are then created that you will need shortly:

```
mkdir -p /var/log/lastlog
mkdir /home
mkdir /var/run
mkdir /tmp
mkdir /var/tmp
chmod 01777 /tmp /var/tmp
```

You then create the inittab, which will launch syslogd and Dropbear once at startup (in addition to the existing getty that is respawned whenever it dies):

```
echo ::sysinit:/bin/syslogd >> /etc/inittab
echo '::sysinit:/bin/dropbear -w -p 2200' >> /etc/inittab
```

Next, you add a shadow file and create a password for root:

```
echo root:::::::: > /etc/shadow
chmod 600 /etc/shadow
echo root:x:0: > /etc/group
passwd -a x root
```

Note that the BusyBox `passwd` call used here generated an MD5 hash—there is a 1 prefix in the second field of /etc/shadow for root. Additional hashing algorithms are available from this version of the passwd utility (the options `-a s` will generate a 5 SHA256 hash, and `-a sha512` will generate a 6 hash). However, Dropbear seems to be able to work only with 1 hashes for now.

Finally, add a new user to the system, and then halt the container:

```
adduser -h /home/luser -D luser
passwd -a x luser

halt
```

You should see container shutdown messages that are similar to a system halt.

When you next start your container, it will listen on socket 2200 for connections. If you want remote hosts to be able to connect to your container from anywhere on the network, run this command as root on the host to open a firewall port:

```
iptables -I INPUT -p tcp --dport 2200 --syn -j ACCEPT
```

The port will be open only until you reboot. If you'd like the open port to persist across reboots, use the `firewall-config` command from within the X Window System (set the port on the second tab in the GUI).

In any case, run the container with the previous `nspawn` syntax, then try to connect from another shell within the parent host OS with the following:

```
ssh -l luser -p 2200 localhost
```

You should be able to log in to the luser account under a BusyBox shell.

Executing Programs with Runtime Dependencies

If you copy various system programs from /bin or /usr/bin into your container, you immediately will notice that they don't work. They are missing shared objects that they need to run.

If you had previously copied the gawk binary in from the host:

```
cp /bin/gawk /home/nifty/bin/
```

you would find that attempts to execute it fail with "gawk: not found" errors (on the host, there usually will be explicit complaints about missing shared objects, which are not seen in the container).

You easily can make most of the 64-bit libraries available with an argument to `nspawn` that establishes a bind mount:

```
systemd-nspawn -bD /home/nifty --bind-ro=/usr/lib64
```

Then, from within the container, run:

```
cd /
ln -s usr/lib64 lib64
```

You then will find that many 64-bit binaries that you copy in from the host will run (running `/bin/gawk -V` returns "GNU Awk 4.0.2"—an entire Oracle 12c instance is confirmed to run this way). The read-only library bind mount also has the benefit of receiving security patches immediately when they appear on the host.

There is a significant security problem with this, however. The root user in the container has the power to `mount -o remount,rw /usr/lib64` and, thus, gain write access to your host library directories. In general, you cannot give root to a container user that you don't know and trust—among other problems, these mounts can be abused.

You also might be tempted to mount the /usr/lib directory in the same manner. The difficulty you will find is that the systemd binary will be found under that directory

tree, and `nspawn` will try to execute it in preference to BusyBox init. Enabling 32-bit runtime support likely will involve more directory and mounting gymnastics than was required for /usr/lib64.

And now, I'm going off on a tangent.

systemd Service Files

You will need to call on the host PID 1 (systemd) directly to launch your container in an automated manner, potentially at boot. To do this, you need to create a service file.

Because there is a dearth of clear discussion on moving inittab and service functions into systemd, I'll cover all the basic uses before creating a service file for the container.

Start by configuring a telnet server. The telnet protocol is not secure, as it transmits passwords in clear text. Don't practice these examples on a production server or with sensitive information or accounts.

Classical telnetd is launched by the inetd superserver, both of which are implemented by BusyBox. Let's configure inetd for telnet on port 12323. Run the following as root on the host:

```
echo '12323 stream tcp nowait root
 ➥/home/nifty/bin/telnetd telnetd -i -l
/home/nifty/bin/login' >> /etc/inetd.conf
```

After the configuring above, if you manually launch the inetd contained in BusyBox, you will be able to telnet to port 12323. Note that the V7 platform does not include a telnet client by default, so you either can install it with yum or use the BusyBox client (which the example below will do). Unless you open up port 12323 on your firewall, you will have to telnet to localhost.

Make sure any inetd that you started is shut down before proceeding to create an inetd service file below:

```
echo '[Unit]
Description=busybox inetd
#After=network-online.target
Wants=network-online.target

[Service]
#ExecStartPre=
#ExecStopPost=
#Environment=GZIP=-9

#OPTION 1
ExecStart=/home/nifty/bin/inetd -f
Type=simple
KillMode=process

#OPTION 2
#ExecStart=/home/nifty/bin/inetd
#Type=forking
```

```
#Restart=always
#User=root
#Group=root

[Install]
WantedBy=multi-user.target' >
 ➥/etc/systemd/system/inetd.service

systemctl start inetd.service
```

After starting the inet service above, you can check the status of the dæmon:

```
[root@localhost ~]# systemctl status inetd.service
inetd.service - busybox inetd
   Loaded: loaded (/etc/systemd/system/inetd.service; disabled)
   Active: active (running) since Sun 2014-11-16 12:21:29 CST;
           ➥28s ago
 Main PID: 3375 (inetd)
   CGroup: /system.slice/inetd.service
           ➥3375 /home/nifty/bin/inetd -f

Nov 16 12:21:29 localhost.localdomain systemd[1]: Started
 ➥busybox inetd.
Try opening a telnet session from a different console:

/home/nifty/bin/telnet localhost 12323
```

You should be presented with a login prompt:

```
Entering character mode
Escape character is '^]'.

S
Kernel 3.10.0-123.9.3.el7.x86_64 on an x86_64
localhost.localdomain login: jdoe
Password:
```

Checking the status again, you see information about the connection and the session activity:

```
[root@localhost ~]# systemctl status inetd.service
inetd.service - busybox inetd
   Loaded: loaded (/etc/systemd/system/inetd.service; disabled)
   Active: active (running) since Sun 2014-11-16 12:34:04 CST;
           ➥7min ago
 Main PID: 3927 (inetd)
   CGroup: /system.slice/inetd.service
           ➥3927 /home/nifty/bin/inetd -f
           ➥4076 telnetd -i -l /home/nifty/bin/login
           ➥4077 -bash
```

You can learn more about systemd service files with the `man 5 systemd.service` command.

There is an important point to make here—you have started inetd with the "-f Run in foreground" option. This is not how inetd normally is started—this option is commonly used for debugging activity. However, if you were starting inetd with a classical inittab entry, `-f` would be useful in conjunction with "respawn". Without `-f`, inetd immediately will fork into the background; attempting to

respawn forking dæmons will launch them repeatedly. With `-f`, you can configure init to relaunch inetd should it die.

Another important point is stopping the service. With a foreground dæmon and the `KillMode=process` setting in the service file, the child telnetd services are not killed when the service is stopped. This is not the normal, default behavior for a systemd service, where all the children will be killed.

To see this mass kill behavior, comment out the `OPTION 1` settings in the service file (/etc/systemd/system/inetd.service), and enable the default settings in `OPTION 2`. Then execute:

```
systemctl stop inetd.service
systemctl daemon-reload
systemctl start inetd.service
```

Launch another telnet session, then stop the service. When you do, your telnet sessions will all be cut with "Connection closed by foreign host." In short, the default behavior of systemd is to kill all the children of a service when a parent dies.

The `KillMode=process` setting can be used with the forking version of inetd, but the "-f Run in foreground" in the first option is more specific and, thus, safer.

You can learn more about the `KillMode` option with the `man 5 systemd.kill` command.

Note also that the `systemctl status` output included the word "disabled". This indicates that the service will not be started at boot. Pass the `enable` keyword to `systemctl` for the service to set it to launch at boot (the `disable` keyword will undo this).

Make some note of the commented options above. You may set environment variables for your service (here suggesting a compression quality), specify a non-root user/group and commands to be executed before the service starts or after it is halted. These capabilities are beyond the direct features offered by the classical inittab.

Of course, systemd is capable of spawning telnet servers directly, allowing you to dispense with inetd altogether. Run the following as root on the host to configure systemd for BusyBox telnetd:

```
systemctl stop inetd.service

echo '[Unit]
Description=mytelnet

[Socket]
```

```
ListenStream=12323
Accept=yes

[Install]
WantedBy=sockets.target' >
 ➥/etc/systemd/system/mytelnet.socket

echo '[Unit]
Description=mytelnet

[Service]
ExecStart=-/home/nifty/bin/telnetd telnetd -i -l
 ➥/home/nifty/bin/login
StandardInput=socket' >
 ➥/etc/systemd/system/mytelnet@.service

systemctl start mytelnet.socket
```

Some notes about inetd-style services:

- The socket is started, rather than the service, when inetd services are launched. Similarly, they are enabled to set them to launch at boot.
- The @ character in the service file indicates this is an "instantiated" service. They are used when a number of similar services are launched with a single service file (getty being the prime example—they also work well for Oracle database instances).
- The - prefix above in the path to the telnet server indicates that systemd should not pay attention to any stats return codes from the process.
- In the client telnet sessions, the command `cat /proc/self/cgroup` will return detailed connection information for the IP addresses involved.

At this point, I have returned from my long-winded tangent, so now let's build a service file for the container. Run the following as root on the host:

```
echo '[Unit]
Description=nifty container

[Service]
ExecStart=/usr/bin/systemd-nspawn -bD /home/nifty
KillMode=process' > /etc/systemd/system/nifty.service
```

Be sure that you have shut down any other instances of the nifty container. You optionally can disable the console getty by commenting/removing the first line of /home/nifty/etc/inittab. Then use PID 1 to launch your container directly:

```
systemctl start nifty.service
```

If you check the status of the service, you will see the same level of

information that you previously saw on the console:

```
[root@localhost ~]# systemctl status nifty.service
nifty.service - nifty container
   Loaded: loaded (/etc/systemd/system/nifty.service; static)
   Active: active (running) since Sun 2014-11-16 14:06:21 CST;
           ➥31s ago
 Main PID: 5881 (systemd-nspawn)
   CGroup: /system.slice/nifty.service
           ➥5881 /usr/bin/systemd-nspawn -bD /home/nifty

Nov 16 14:06:21 localhost.localdomain systemd[1]: Starting
 ➥nifty container...
Nov 16 14:06:21 localhost.localdomain systemd[1]: Started
 ➥nifty container.
Nov 16 14:06:26 localhost.localdomain systemd-nspawn[5881]:
 ➥Spawning namespace container on /home/nifty
 ➥(console is /dev/pts/4).
Nov 16 14:06:26 localhost.localdomain systemd-nspawn[5881]:
 ➥Init process in the container running as PID 5883.
```

Memory and Disk Consumption

BusyBox is a big program, and if you are running several containers that each have their own copy, you will waste both memory and disk space.

It is possible to share the "text" segment of the BusyBox memory usage between all running programs, but only if they are running on the same inode, from the same filesystem. The text segment is the read-only, compiled code of a program, and you can see the size like this:

```
[root@localhost ~]# size /home/busybox-x86_64
   text    data     bss     dec     hex filename
 942326   29772   19440  991538  f2132 /home/busybox-x86_64
```

If you want to conserve the memory used by BusyBox, one way would be to create a common /cbin that you attach to all containers as a read-only bind mount (as you did previously with lib64), and reset all the links in /bin to the new location. The root user could do this:

```
systemctl stop nifty.service

mkdir /home/cbin
mv /home/nifty/bin/busybox-x86_64 /home/cbin
mv /home/nifty/bin/dropbearmulti-x86_64 /home/cbin
cd /
ln -s home/cbin cbin
cd /home/nifty/bin
for x in *; do if [ -h "$x" ]; then rm -f "$x"; fi; done
/cbin/busybox-x86_64 --list | awk '{print "ln -s
 ➥/cbin/busybox-x86_64 " $0}' | sh
ln -s /cbin/dropbearmulti-x86_64 dropbear
ln -s /cbin/dropbearmulti-x86_64 ssh
ln -s /cbin/dropbearmulti-x86_64 scp
ln -s /cbin/dropbearmulti-x86_64 dropbearkey
ln -s /cbin/dropbearmulti-x86_64 dropbearconvert
```

You also could arrange to bind-mount the zoneinfo directory, saving a little more disk space in the container (and giving the container patches for time zone

It might interesting to launch tens, hundreds, or even thousands of containers at once.

data in the bargain):

```
cd /home/nifty/usr/share
rm -rf zoneinfo
```

Then the service file is modified to bind /cbin and /usr/share/zoneinfo (note the altered syntax for sharing /cbin below, when the paths differ between host and container):

```
echo '[Unit]
Description=nifty container

[Service]
ExecStart=/usr/bin/systemd-nspawn -bD /home/nifty
--bind-ro=/home/cbin:/cbin --bind-ro=/usr/share/zoneinfo
KillMode=process' > /etc/systemd/system/nifty.service

systemctl daemon-reload

systemctl start nifty.service
```

Now any container using the BusyBox binary from /cbin will share the same inode. All versions of the BusyBox utilities running in those containers will share the same text segment in memory.

Infinite BusyBox

It might interesting to launch tens, hundreds, or even thousands of containers at once. You could launch the clones by making copies of the /home/nifty directory, then adjusting the systemd service file. To simplify, you will place your new containers in /home/nifty1, /home/nifty2, /home/nifty3 … using integer suffixes on the directories to differentiate them.

Please make sure that you have disabled kernel auditing to remove the five-second delay when launching containers. At the very least, press e at the grub menu at boot time, and add the `audit=0` to your kernel command line for a one-time boot.

I'm going to return to the subject of systemd "instantiated services" that I touched upon with the telnetd service file that replaced inetd. This technique will allow you to use one service file to launch all of your containers. Such a service has an @ character in the filename that is used to refer to a particular, differentiated instance of a service, and it allows the use of the %i placeholder within the

service file for variable expansion. Run the following on the host as root to place your service file for instantiated containers:

```
echo '[Unit]
Description=nifty container # %i

[Service]
ExecStart=/usr/bin/systemd-nspawn -bD /home/nifty%i
 ➥--bind-ro=/home/cbin:/cbin --bind-ro=/usr/share/zoneinfo
KillMode=process' > /etc/systemd/system/nifty@.service
```

The `%i` above first adjusts the description, then adjusts the launch directory for the `nspawn`. The content that will replace the `%i` is specified on the `systemctl` command line.

To test this, make a directory called /home/niftyslick. The service file doesn't limit you to numeric suffixes. You will adjust the SSH port after the copy. Run this as root on the host:

```
cd /home
mkdir niftyslick
(cd nifty; tar cf - .) | (cd niftyslick; tar xpf -)
sed "s/2200/2100/" < nifty/etc/inittab > niftyslick/etc/inittab

systemctl start nifty@slick.service
```

Bearing this pattern in mind, let's create a script to produce these containers in massive quantities. Let's make a thousand of them:

```
cd /home
for x in $(seq 1 999)
do
  mkdir "nifty${x}"
  (cd nifty; tar cf - .) | (cd "nifty${x}"; tar xpf -)
  sed "s/2200/$((x+2200))/" < nifty/etc/inittab >
   ➥nifty${x}/etc/inittab
  systemctl start nifty@${x}.service
done
```

As you can see below, this test launches all containers:

```
$ ssh -l luser -p 3199 localhost
The authenticity of host '[localhost]:3199 ([::1]:3199)'
 ➥can't be established.
ECDSA key fingerprint is 07:26:15:75:7d:15:56:d2:ab:9e:
➥14:8a:ac:1b:32:8c.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '[localhost]:3199' (ECDSA)
 ➥to the list of known hosts.
luser@localhost's password:
~ $ sh --help
BusyBox v1.21.1 (2013-07-08 11:34:59 CDT) multi-call binary.

Usage: sh [-/+OPTIONS] [-/+o OPT]... [-c 'SCRIPT'
 ➥[ARG0 [ARGS]] / FILE [ARGS]]

Unix shell interpreter

~ $ cat /proc/self/cgroup
10:hugetlb:/
9:perf_event:/
```

```
8:blkio:/
7:net_cls:/
6:freezer:/
5:devices:/
4:memory:/
3:cpuacct,cpu:/
2:cpuset:/
1:name=systemd:/machine.slice/machine-nifty999.scope
```

The output of systemctl will list each of your containers:

```
# systemctl
...
machine-nifty1.scope    loaded active running   Container nifty1
machine-nifty10.scope   loaded active running   Container nifty10
machine-nifty100.scope  loaded active running   Container nifty100
machine-nifty101.scope  loaded active running   Container nifty101
machine-nifty102.scope  loaded active running   Container nifty102
...
```

More detail is available with `systemctl status`:

```
machine-nifty10.scope - Container nifty10
   Loaded: loaded (/run/systemd/system/machine-nifty10.scope;
                   ➥static)
  Drop-In: /run/systemd/system/machine-nifty10.scope.d
           ➥90-Description.conf, 90-Slice.conf,
           ➥90-TimeoutStopUSec.conf
   Active: active (running) since Tue 2014-11-18 23:01:21 CST;
           ➥11min ago
   CGroup: /machine.slice/machine-nifty10.scope
           ➥2871 init
           ➥2880 /bin/syslogd
           ➥2882 /bin/dropbear -w -p 2210

Nov 18 23:01:21 localhost.localdomain systemd[1]:
 ➥Starting Container nifty10.
Nov 18 23:01:21 localhost.localdomain systemd[1]:
 ➥Started Container nifty10.
```

The raw number of containers that you can launch with this approach is more directly impacted by kernel limits than general disk and memory resources. Launching the containers above used no swap on a small system with 2GB of RAM.

After you have investigated a few of the containers and their listening ports, the easiest and cleanest way to get all of your containers shut down is likely a reboot.

Container Security

A number of concerns are raised with these features:

1) Since BusyBox and Dropbear were not installed with the RPM host package tools, updates to them will have to be loaded manually. It will be important to check from time to time if new versions are available and if any security flaws have been discovered. If it is necessary to load new versions, the binaries should be copied to all containers that are potentially used, which should then be restarted (especially if a security

The crux is that untrusted users cannot have the container root, any more than you would give them full system root.

issue is involved).

2) Control of the root user in the container cannot be passed to an individual that you do not trust. For a particular example, if the lib64/cbin/zoneinfo bind mounts above are used, the container root user can issue the command:

```
mount -o remount,rw /usr/lib64
```

at which point the container root will have full write privileges on your 64-bit libraries, container bin or zoneinfo. The systemd-nspawn man page goes even further, with the warning:

> Note that even though these security precautions are taken systemd-nspawn is not suitable for secure container setups. Many of the security features may be circumvented and are hence primarily useful to avoid accidental changes to the host system from the container. The intended use of this program is debugging and testing as well as building of packages, distributions and software involved with boot and systems management.

The crux is that untrusted users cannot have the container root, any more than you would give them full system root. The container root will have the CAP_SYS_ADMIN privilege, which allows full control of the system. If you want to isolate non-root users further, the container environment does limit non-root users' visibility into host activities, as they cannot see the full process table.

3) Note that the BusyBox su and passwd utilities above do not work when installed in the manner outlined here. They lack the appropriate filesystem permissions. To fix this, `chmod u+s busybox-x86_64` could be executed, but this is also distasteful from a security perspective. Removing the links and copying the BusyBox binary to su and passwd before applying the setuid privilege

might be better, but only slightly. It would be best if su was unavailable and another mechanism was found for password changes.

4) The `-w` argument to the Dropbear SSH server above prevents root logins from the network. It is somewhat distasteful, from a security perspective, to relax this limitation. The net effect is that root is locked out of active use in the container when `-w` is forced, and su/passwd do not have setuid. If it is at all possible to live with such an arrangement for your container, try to do so, as the security is much improved.

systemd Controversy

There is a high degree of hostility toward systemd from users of Linux. This hostility is divided into two main complaints:

- The classic inittab from UNIX System V should not be changed because it is well understood.
- Increasing features are bundled into systemd that bring dangerous complexity to a critical system process.

Toward the first point, nostalgia for legacy systems is not always misguided, but it cannot be allowed to hinder progress unreasonably. A classic System V init is not able to nspawn and has far less control over processes running on a system. The features delivered by systemd surely justify the inconvenience of change in many situations.

Toward the second point, much thought was placed into the adoption of the architecture of systemd by skilled designers from diverse organizations. Those most critical of the new environment should acknowledge the technical success of systemd as it is adopted by the majority of the Linux community.

In any case, the next decade will see popular Linux server distributions equipped with systemd, and competent administrators will not have the option of ignoring it. It is unfortunate that the introduction of systemd did not include more guidance for the user community, but the new features are compelling and should not be overlooked.■

Charles Fisher has an electrical engineering degree from the University of Iowa and works as a systems and database administrator for a Fortune 500 mining and manufacturing corporation. He has previously published both journal articles and technical manuals on Linux for *UnixWorld* and other McGraw-Hill publications.

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

WEBCASTS

Learn the 5 Critical Success Factors to Accelerate IT Service Delivery in a Cloud-Enabled Data Center

Today's organizations face an unparalleled rate of change. Cloud-enabled data centers are increasingly seen as a way to accelerate IT service delivery and increase utilization of resources while reducing operating expenses. Building a cloud starts with virtualizing your IT environment, but an end-to-end cloud orchestration solution is key to optimizing the cloud to drive real productivity gains.

> **http://lnxjr.nl/IBM5factors**

Modernizing SAP Environments with Minimum Risk—a Path to Big Data

Sponsor: SAP | Topic: Big Data

Is the data explosion in today's world a liability or a competitive advantage for your business? Exploiting massive amounts of data to make sound business decisions is a business imperative for success and a high priority for many firms. With rapid advances in x86 processing power and storage, enterprise application and database workloads are increasingly being moved from UNIX to Linux as part of IT modernization efforts. Modernizing application environments has numerous TCO and ROI benefits but the transformation needs to be managed carefully and performed with minimal downtime. Join this webinar to hear from top IDC analyst, Richard Villars, about the path you can start taking now to enable your organization to get the benefits of turning data into actionable insights with exciting x86 technology.

> **http://lnxjr.nl/modsap**

WHITE PAPERS

White Paper: JBoss Enterprise Application Platform for OpenShift Enterprise

Sponsor: DLT Solutions

Red Hat's® JBoss Enterprise Application Platform for OpenShift Enterprise offering provides IT organizations with a simple and straightforward way to deploy and manage Java applications. This optional OpenShift Enterprise component further extends the developer and manageability benefits inherent in JBoss Enterprise Application Platform for on-premise cloud environments.

Unlike other multi-product offerings, this is not a bundling of two separate products. JBoss Enterprise Middleware has been hosted on the OpenShift public offering for more than 18 months. And many capabilities and features of JBoss Enterprise Application Platform 6 and JBoss Developer Studio 5 (which is also included in this offering) are based upon that experience.

This real-world understanding of how application servers operate and function in cloud environments is now available in this single on-premise offering, JBoss Enterprise Application Platform for OpenShift Enterprise, for enterprises looking for cloud benefits within their own datacenters.

> **http://lnxjr.nl/jbossapp**

WHITE PAPERS

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

Sponsor: Red Hat | **Topic:** Linux Management

Linux has become a key foundation for supporting today's rapidly growing IT environments. Linux is being used to deploy business applications and databases, trading on its reputation as a low-cost operating environment. For many IT organizations, Linux is a mainstay for deploying Web servers and has evolved from handling basic file, print, and utility workloads to running mission-critical applications and databases, physically, virtually, and in the cloud. As Linux grows in importance in terms of value to the business, managing Linux environments to high standards of service quality — availability, security, and performance — becomes an essential requirement for business success.

> http://lnxjr.nl/RHS-ROI

Standardized Operating Environments for IT Efficiency

Sponsor: Red Hat

The Red Hat® Standard Operating Environment SOE helps you define, deploy, and maintain Red Hat Enterprise Linux® and third-party applications as an SOE. The SOE is fully aligned with your requirements as an effective and managed process, and fully integrated with your IT environment and processes.

Benefits of an SOE:

SOE is a specification for a tested, standard selection of computer hardware, software, and their configuration for use on computers within an organization. The modular nature of the Red Hat SOE lets you select the most appropriate solutions to address your business' IT needs.

SOE leads to:

- Dramatically reduced deployment time.
- Software deployed and configured in a standardized manner.
- Simplified maintenance due to standardization.
- Increased stability and reduced support and management costs.
- There are many benefits to having an SOE within larger environments, such as:
  - Less total cost of ownership (TCO) for the IT environment.
  - More effective support.
  - Faster deployment times.
  - Standardization.

> http://lnxjr.nl/RH-SOE

DOC SEARLS

Resurrecting the Armadillo

Fifteen years after giving the world a pile of hopefully helpful memes, Cluetrain rides again.

1999 was a crazy year for business on the Internet, and for Linux. It was when Red Hat went public, with a record valuation, and VA Linux followed with a bigger one. Both were cases in point of the dot-com boom, a speculative bubble inflated by huge expectations of what the Internet would mean for business.

In April of that year, Chris Locke, Rick Levine, David Weinberger and I put up a Web site called The Cluetrain Manifesto (**http://cluetrain.com**), attempting to make clear that the Internet was for everybody and everything, and not just for companies looking to exit into wealth on Wall Street.

Our bulls-eye was marketing, which spoke about users in terms that were barely human. "We are not seats or eyeballs or end users or consumers", Cluetrain said. "We are human beings and our reach exceeds your grasp. Deal with it."

We addressed Cluetrain to "People of Earth", built it around 95 theses (because that worked for Martin Luther) and called it a "manifesto" (because that worked for Karl Marx). The "Cluetrain" name came from an old Silicon Valley put-down: "The clue train stopped there four times a day and they never took delivery."

It was a hit. Volunteers translated it into 13 languages. *The Wall Street Journal* covered it on the front page of its Marketplace section. Book offers came in. We accepted one and wrote the book version of The Cluetrain Manifesto that summer. It came out in January 2000 and was a business bestseller, even though it could also be read for free on-line at the Cluetrain Web site. It went on

New Clues was tuned for our time—one in which Internet usage has been migrating from wired to cellular connections, from the Web to apps, and from the Net's wide open spaces to the closed and proprietary walled gardens of Facebook, Twitter, Apple, Google and other feudal lords.

to be published in nine languages, and still sells at a nice clip, 15 years later. Same goes for a 10th anniversary edition that came out in 2010.

Today the word *cluetrain*, which didn't exist before 1999, appears in thousands of books and is tweeted many times per day. One-liners from its list of theses—"Markets are conversations", "Hyperlinks subvert hierarchy"—are quoted endlessly. Not bad for an project that had no promotion, no budget, no conferences, no bumper stickers, no t-shirts and no interest in becoming a business or an institution. It was just a bunch of ideas people could put to use.

The biggest irony of Cluetrain's success is that most books that cite it are marketing books, and most tweets about it seem to be by marketing people. Is marketing better because of it? To some degree, I suppose. But I don't care. Nor do I care that Cluetrain is often credited with having something to do with social media. What I care about is that Cluetrain hasn't yet succeeded at its main mission: to make clear that the Internet is something more than the pipes we get it from, the "content" we find there, and the companies and governments that would have us think they run the thing.

So on August 2, 2014, when somebody pointed me to yet another Cluetrain story that failed to grok what it was really about, I wrote this to the other three guys: "I feel an urge to publish something that says 'The Cluetrain Manifesto was not about clearing the way for social media.'" David Weinberger wrote back, "Anyone ready for a new manifesto?" A few minutes later, he shared the first draft of one: a collection of theses, similar to the

original in style and length. Chris Locke followed with a wordless image of a woman gleefully shooting thumbs-up images out of a machine gun. The rest of the back-and-forth was between David and me. (Chris and Rick stayed busy with other things.) The result was New Clues, which went up on January 8, 2015, at the original Cluetrain site: **http://cluetrain.com/newclues**.

New Clues was tuned for our time—one in which Internet usage has been migrating from wired to cellular connections, from the Web to apps, and from the Net's wide open spaces to the closed and proprietary walled gardens of Facebook, Twitter, Apple, Google and other feudal lords. "An organ-by-organ body snatch of the Internet is already well underway", it says in the preamble. "Make no mistake: with a stroke of a pen, a covert handshake, or by allowing memes to drown out the cries of the afflicted we can lose the Internet we love."

One hundred and twenty one numbered clues follow, under thematic subheads:

- The Internet is us, connected.
- The Internet is nothing and has no purpose.
- The Internet is not content.
- The Net is not a medium.
- How did we let conversation get weaponized anyway?
- Marketing still makes it harder to talk.
- Kumbiyah sounds surprisingly good in an echo chamber.
- The Gitmo of the Net.
- Gravity's great until it sucks us all into a black hole.
- Privacy in an age of spies.
- Privacy in an age of weasels.
- A pocket full of homilies.
- Being together: the cause of and solution to every problem.

We wanted to make New Clues, and every piece of it, as useful, mixable and remixable as possible. So:

- Every subhead and every clue has a link of its own.
- The text is released to the public domain with a Creative Commons

We wanted to make New Clues, and every piece of it, as useful, mixable and remixable as possible.

Zero 1.0 Universal (CC0 1.0) Public Domain Dedication. In other words, no copyright at all. When asked for permission to republish New Clues, David replies, "You don't have our permission. Go ahead!"

- The whole thing is on GitHub, with machine-readable versions (JSON, XML and OPML, so far). The GitHub folks also have offered to set up a way for us to maintain a single data file (YAML) that will automatically create all the other versions we need.

The results so far (I'm writing this a week after it went up) include:

- A listicle that Dave Winer hacked together on his own software, which he improved in the course of posting it. (Some people like the listicle version better than the one-page text one. Try it out: **http://scripting.com**.)

- An artful posting on Backchannel at Medium (by invitation from the great Steven Levy): **https://medium.com/backchannel/internet-under-fire-gets-new-manifests-207a922b459e**.

- A version by Kevin Marks that accepts fragmentations and webmentions.

- A site by John Johnston that randomly generates one clue per click: **http://johnjohnston.info/oddsandends/givemeaclue**.

- Thousands of tweets and re-tweets (hashtags: #cluetrain #newclues).

- Volunteer translations into German, Italian, Italian (yes, there are two different ones), Catalan and French (see Resources).

- Lots of great blog posts, such as this one by JP Rangaswami: **http://confusedofcalcutta.com/2015/01/11/new-clues-calling-on-everyone-to-be-dutiful-individuals**.

- A Gillmor Gang devoted to to the subject, with David and

Now the question is, Will it work? Or will it be, like so much else that gets published on the Web, snow on the water?

myself: **http://techcrunch.com/2015/01/10/gillmor-gang-kind-of-clue**.

- A discussion page on Facebook: **https://www.facebook.com/login.php?next=https%3A%2F%2Fwww.facebook.com%2Fgroups%2Fnewclues%2F**.

Our only common design element between Cluetrain and New Clues is an armadillo. On Cluetrain's index page is the image shown in Figure 1 of a flattened armadillo in the middle of a road, painted over with yellow divider lines. The provenance of the photo is unknown to us. Chris Locke found it somewhere, and nobody has ever stepped forward to claim it.

The one for New Clues is shown in Figure 2. It was posted by e. res on Flickr and made useful by a Creative Commons Attribution 2.0 Generic (CC BY 2.0) license. (The shot was taken at Alki Beach in Seattle, the town where *Linux Journal* was born.) For New Clues, we cropped the shot and made it black and white. For his listicle version, Dave Winer kept the color but darkened it.

Figure 1. Armadillo from Cluetrain's Index Page

Among the few criticisms of New Clues is that it's "not so disruptive" as Cluetrain was. For the last few years, Silicon Valley has been so gaga over disruption that it even has a conference named after it. The term

comes from Clayton Christensen's work on disruptive innovation, which is defined as "a process by which a product or service takes root initially in simple applications at the bottom of a market and then relentlessly moves up market, eventually displacing established competitors". This can apply to ideas as well as technologies. Cases in point: free software and open source. Both of which, of course, informed Cluetrain and New Clues.

Now the question is, *Will it work?* Or will it be, like so much else that gets published on the Web, snow on the water? Can't say, so soon after it's published. But the two publishing dates, a decade and a half apart, came at very different times on the Web, and we did our best to leverage both.

In 1999, the Web was a collection of almost physical places. You *put up* or *built* Web *sites* on *domains* with *locations* that people *visited* and *browsed*. Search engines might take days or weeks to index a page. But then, after blogs came along, with syndication through RSS, what my son Allen in 2003 described as "the Live Web" began to emerge. Technorati and other search engines for live stuff appeared. My October 2005 column in *Linux Journal* was titled "The World Live Web". In it I said the Live Web was "about time and people, rather than

Figure 2. New Clues Armadillo

sites and content". This was a year before Twitter and Facebook took off, and search engines' time-to-index was reduced nearly to zero. Sites today are geysers of content, and the Live Web is a giant short-attention-span theater. What shows there is hyper-social and temporary in the extreme, made more for sharing than for saving.

This is why New Clues is a collection of stuff for sharing, presented atop an old chunk of bedrock, which is what Cluetrain has become. If we had only put it on Medium, or on Facebook, it would have come and gone in a week. As buzz, that's pretty much what it did. But as durable food for re-thinking what the Net is, and how we might lose it, maybe it will have lasting effects. Sure hope so.■

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

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

Resources

Dot-Com Bubble: **http://en.wikipedia.org/wiki/Dot-com_bubble**

Chris Locke: **http://rageboy.com**

Rick Levine: **https://twitter.com/ricklevine**

David Weinberger: **http://www.hyperorg.com/blogger**

The Cluetrain Manifesto: **http://cluetrain.com**

The entire original text of *The Cluetrain Manifesto*: **http://cluetrain.com/book**

"What The Cluetrain Manifesto Teaches Us On Social Media...11 Years Later": **http://visionarymarketing.com/en/blog/2010/02/what-the-cluetrain-manifesto-teaches-us-on-social-media-11-years-later**

The Cluetrain Legacy and Social Media: **http://www.chrisg.com/cluetrain-social-media**

Invasion of the Body Snatchers: **http://en.wikipedia.org/wiki/Invasion_of_the_Body_Snatchers**

CC0 1.0 Universal (CC0 1.0) Public Domain Dedication: **http://creativecommons.org/publicdomain/zero/1.0**

e. res on Flickr: **https://www.flickr.com/photos/iamtheloop**

Creative Commons Attribution 2.0 Generic License: **https://creativecommons.org/licenses/by/2.0**

Backchannel: New Clues: **https://medium.com/backchannel/internet-under-fire-gets-new-manifests-207a922b459e**

Steven Levy: **http://www.stevenlevy.com**

John Johnston: **http://johnjohnston.info**

German Translation: **http://conceptbakery.de/blog/2015/01/11/new-clues-deutsche-uebersetzung-die-neuen-thesen-von-den-verfassern-des-cluetrain-manifest**

Italian Translation 1: **https://medium.com/bee-free-the-social-bee/cluetrain-15-anni-dopo-9d6b4def4d57**

Italian Translation 2: **https://medium.com/@nuovetesi/nuove-tesi-4a1def360351**

Catalan Translation: **https://ca.wikisource.org/wiki/New_clues**

"New Clues: Calling on everyone to be Dutiful Individuals" by JP Rangaswami: **http://confusedofcalcutta.com/2015/01/11/new-clues-calling-on-everyone-to-be-dutiful-individuals**

Gillmor Gang: Kind of Clue: **http://techcrunch.com/2015/01/10/gillmor-gang-kind-of-clue**

Facebook Discussion Page: **https://www.facebook.com/login.php?next=https%3A%2F%2Fwww.facebook.com%2Fgroups%2Fnewclues%2F**

Disrupt Conference: **http://techcrunch.com/event-type/disrupt**

Clayton Christensen: **http://www.claytonchristensen.com**

Disruptive Innovation: **http://www.claytonchristensen.com/key-concepts**

"Snow on the Water": **http://blogs.law.harvard.edu/doc/2014/08/03/snow-on-the-water**

"The World Live Web" by Doc Searls in the October 2005 issue of *LJ*: **http://www.linuxjournal.com/article/8549**

drupalize.me

Instant Access to Premium Online Drupal Training

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

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

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