OCTOBER 2015 | ISSUE 258 | www.linuxjournal.com

LINUX JOURNAL

Since 1994: The Original Magazine of the Linux Community

AN INDEPTH LOOK AT WI-FI TECHNOLOGY

DODGE BLOCKED NETWORKS with an RPi

+ Android Apps for Science

RASPBERRY PI

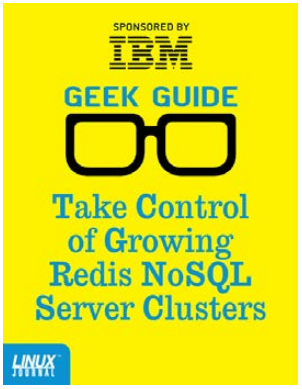
BUILD A LARGE-SCREEN COMMAND CENTER

Amazon's Virtual Private Cloud and the New AWS CLI Tools

FIXING POTENTIAL BOTTLENECKS IN WEB APPS

PERSONAL BOUNDARIES AND THE CLOUD

WATCH: ISSUE OVERVIEW



Take Control of Growing Redis NoSQL Server Clusters

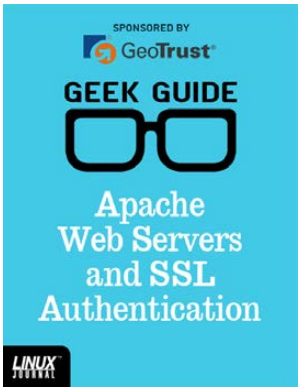
Author: Reuven M. Lerner

Sponsor: IBM

Linux in the Time of Malware

Author: Federico Kereki

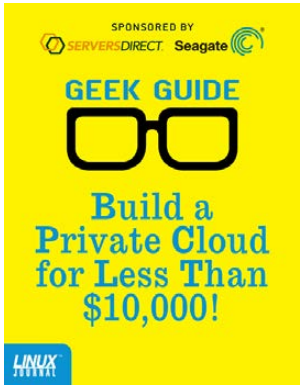
Sponsor: Bit9 + Carbon Black



Apache Web Servers and SSL Encryption

Author: Reuven M. Lerner

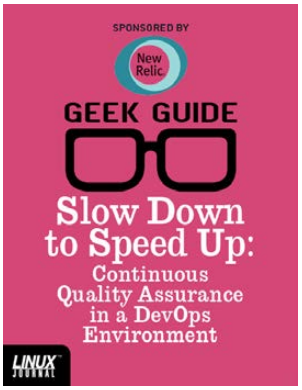
Sponsor: GeoTrust



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

Author: Mike Diehl

Sponsor: Servers Direct and Seagate



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

Author: Bill Childers

Sponsor: New Relic

Enterprise Monitoring Success

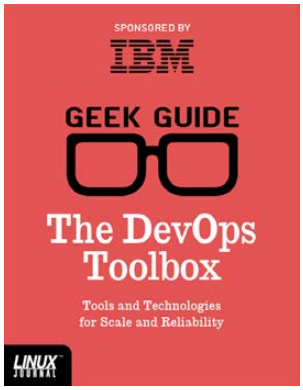
Author: Mike Diehl

Sponsor: InterMapper

Beyond Cron: How to Know When You've Outgrown Cron Scheduling—and What to Do Next

Author: Mike Diehl

Sponsor: Skybot



The DevOps Toolbox: Tools and Technologies for Scale and Reliability

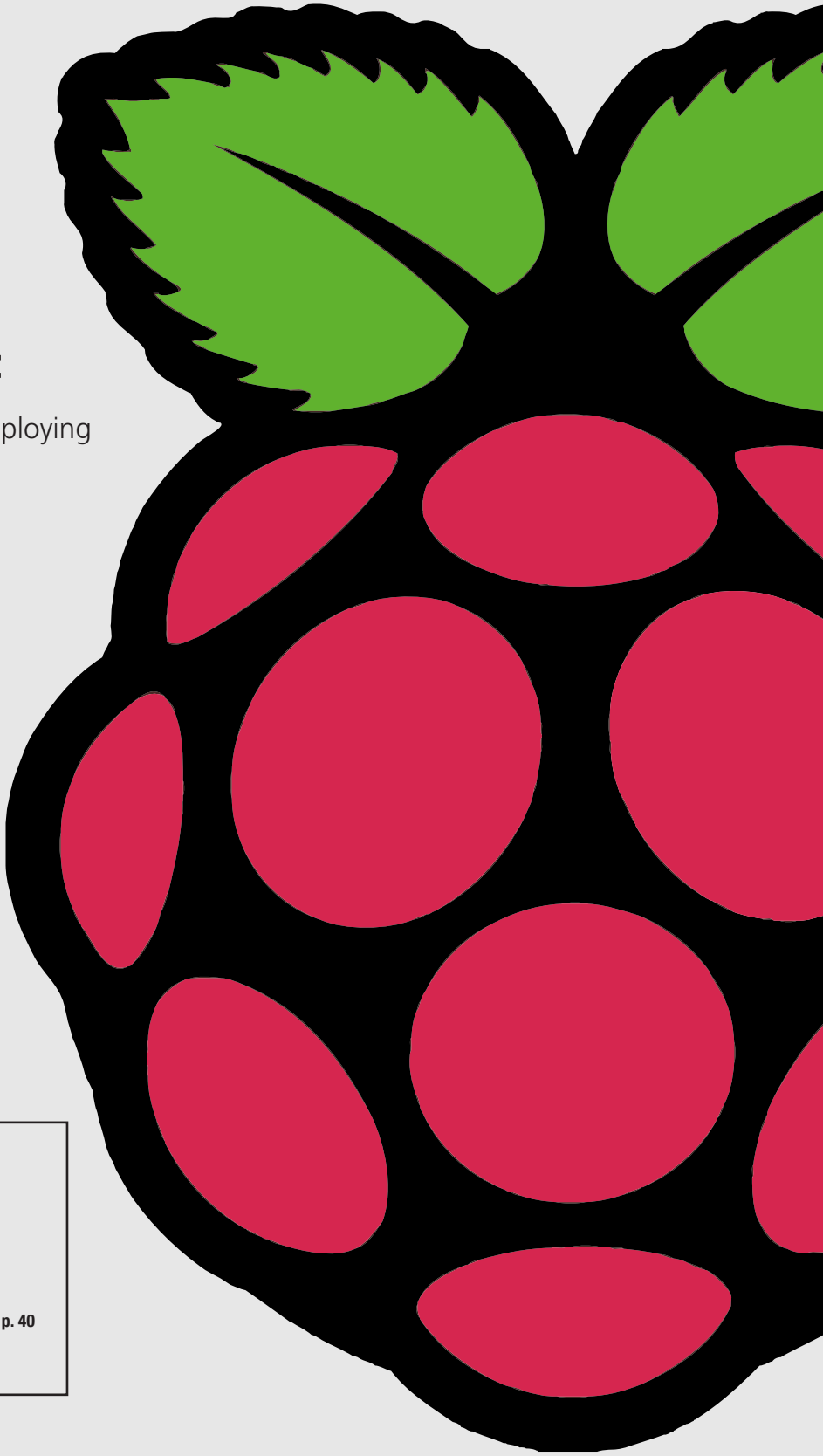
Author: Bill Childers

Sponsor: IBM

CONTENTS

OCTOBER 2015
ISSUE 258

RASPBERRY PI

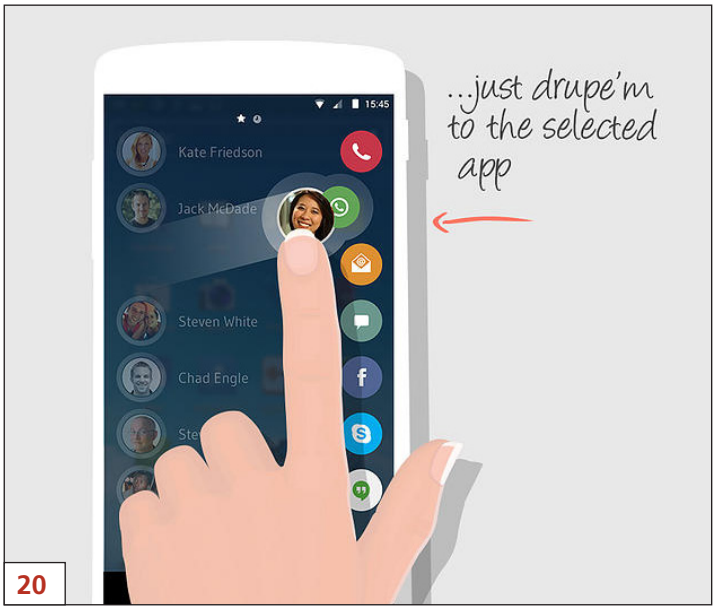


FEATURES

ON THE COVER

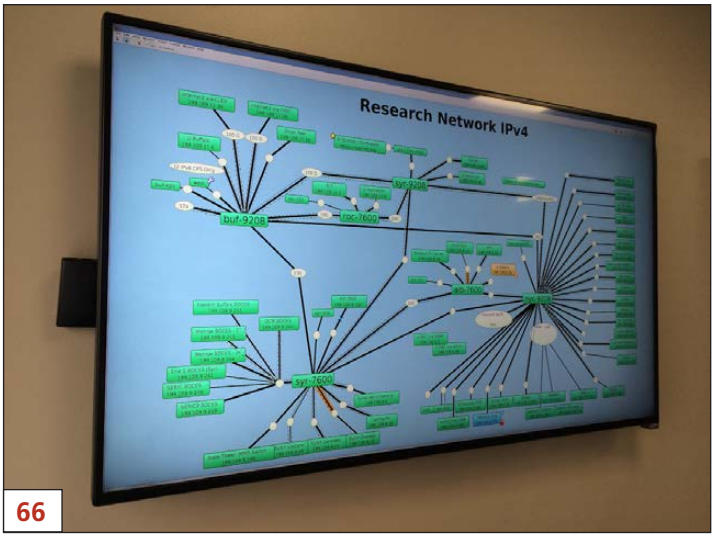
COLUMNS

IN EVERY ISSUE



20

26



66

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

LINUX JOURNAL

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

Contributing Editors

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

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

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

Seattle GNU/Linux Conference

Seattle's grassroots free software summit welcomes **Shauna Gordon-McKeon** and **Richard Stallman**!

Join speakers and participants from around the world for the third year of Seattle's free, as in freedom and beer, GNU/Linux Conference.

With 32 talks, two keynotes, and the inagural Cascadia Community Builder Award, this year is sure to be a blast!

October 23rd and 24th, 2015,
Seattle Central College campus
1701 Broadway Seattle, WA

→ Visit **SeaGL.org** for more information.

SHAWN POWERS

A Tasty Slice of Awesome

I love the flavor of raspberries, but quite honestly, the seeds gross me out. They get stuck in my teeth, and whenever I crunch them, it feels like I have a mouth full of sand. That (among other geeky reasons) is why I love Raspberry Pi devices so much. They have all the awesome, with none of the seeds! This month is our Raspberry Pi issue, and if you've ever wanted a reason to fiddle with one of the tiny computers, look no further.

Before getting into the RPi projects, Reuven M. Lerner takes us back into the world of slow Web applications. Last month he described how to identify some of the bottlenecks for which Web apps are prone; this month he starts to explore what to do about those bottlenecks. Whether you need to beef up your database or tweak some of your code, Reuven will help you not only identify, but also fix come common issues causing slowdowns.

Dave Taylor plays with blocks again this issue, but since he invites us to play along, we won't complain. Seriously though, Dave continues his series on how to figure out the number of words you can possibly get from a given set of children's letter blocks. It's amazing how many words can be generated from a limited set of letters! Speaking of limited sets of letters, Kyle Rankin helps us master three-letter acronyms—well, a few of them anyway. Kyle shows how to use his beloved command line to manipulate Virtual Private Cloud (VPC) instances in the EC2 component of Amazon's Web Services (AWS). The Web interface is convenient, but there are many times it is easier (and more important, scriptable) to be able to utilize the command line for common tasks. Kyle describes how to use those tools and how to get around some of the

VIDEO:
Shawn Powers runs through the latest issue.

frustrations as well.

My family recently moved into a new house, and one of the first things I did was install Wi-Fi. The old plaster and lath walls make signal penetration a bit frustrating, and while I was figuring out how to best place access points, it occurred to me how complicated wireless networking has become. This month I start a two-part series on Wi-Fi. I cover the various technologies this month, and next month I'll talk about the actual installation. Even if you're a Wi-Fi veteran, you might find something interesting as I tear apart the spectra.

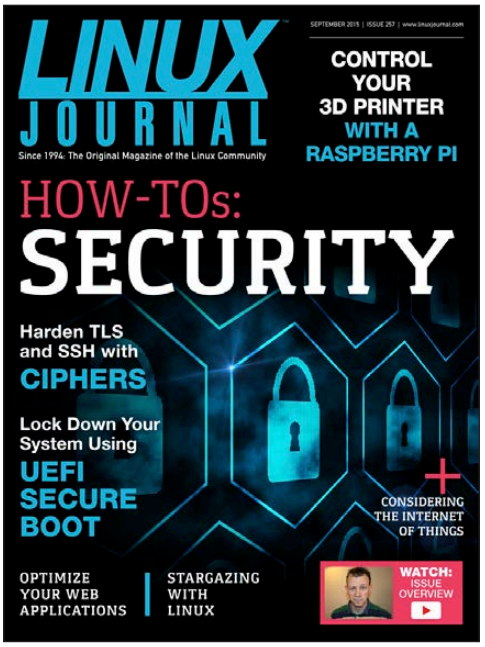
Finally, let's look at the Raspberry-flavored section of this issue. Bill Childers starts with a really cool project that turns a Raspberry Pi server at your home or business into an Internet portal allowing access even when on severely restricted networks. If you've ever been at a hotel and discovered that port 22 traffic is blocked, you'll want to check out Bill's article. He not only will help you get SSH working again, but he also uses a Raspberry Pi to make your entire computing experience useful again.

John S. Tonello tackles a Raspberry Pi project this month as well, but he takes advantage of the new Raspberry Pi 2. In fact, he uses two of them. John's goal was to replace a bank of ten 19-inch monitors, which were used as a sort of dashboard for monitoring, with two 4K-resolution monitors. That's a lot of screen real estate, and the new quad-core RPi units were up to the task. Before you tackle a similar project, I urge you to read John's article. It will save you hours of frustration and give you a heads up on what sort of performance to expect before you spend the money on a giant 4K television.

As always, this issue of *Linux Journal* contains lots of things I haven't covered in this intro. We have brief reviews, announcements, fun programs and plenty of useful tech tips. The star of the show, however, is a $35 computer. It's been several years since the Raspberry Pi was first introduced, and it still remains the favorite platform for weekend projects everywhere. The new RPi units are far more powerful, yet they manage to stay affordable enough for most hobbyists to purchase them without breaking the bank. Plus, they still don't have any seeds!■

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



Hash Tables

Regarding Mihalis Tsoukalos' article "Hash Tables—Theory and Practice" in the August 2015 issue: thank you very much for your interesting article about this important data structure. It was well written and illustrated with programming examples. However, the results of ht1.c and ht2.c (execution time vs. table size) are not surprising. Since the hash function of ht1.c uses only the first character of the key, there are effectively no more than 26 different outcomes of the hash function. Since each result maps to one bucket (which are all different in the best case of no collisions), no more than 26 buckets will be employed. Once `tablesize` exceeds 26, there will be no increase in performance at all (26 being the number of uppercase ASCII characters).

As for ht2.c, the same effect is expected at a table size of about 500, assuming a maximum word length of, say, 20 and a maximum contribution of 26 per character, yielding roughly 26 x 20 different outcomes of the hash function. In practice, the limit should be observed even earlier. This is what we see in the diagrams.

Better results with respect to table size scaling can be expected using the following hash function:

```
int hash_3(char *str, int tablesize)
{
#define MAX_LEN     6
static int v[MAX_LEN] = {1, 26, 26*26, 26*26*26,
➥26*26*26*26, 26*26*26*26*26};
int value, i;
if (str==NULL)
{
return -1;
}
value=0;
for (i=0; *str && i<MAX_LEN; i++, str++)
{
value += (toupper(*str) - 'A' + 1) * v[i];
}
```

```
return value % tablesize;
}
```

Provided a large table size, this function has up to 0.3 billion different outcomes (26^6), at a slightly higher computation cost. Increasing `MAX_LEN` beyond 6 is possible but demands 64-bit integer arithmetic.
—Florian Gagel

Slay Some Orcs

Thanks for Shawn Powers' "Slay Some Orcs" article in the August 2015 issue of *Linux Journal*.

In addition to the mentioned Steam and Desura on-line game stores, I also suggest looking at GOG (http://www.gog.com). Although most of its titles are old (GOG stands for Good Old Games), there are some current games including Linux ones. If both GOG and Steam have a game, I usually purchase it from GOG because all of its games are DRM-free and you don't need to run a distribution client (like Steam) to run the game.

Three games I would add to your list:

1) *Pillars of Eternity* (Steam and GOG): a modern strategy/adventure game in the style of *Baulder's Gate*.

2) *Middle Earth: Shadow of Mordor* (Steam): the Linux version is out. With an article title like "Slay Some Orcs", how could this one be missed? Lots of Orc to kill here. (But there's an out: the article was written before the Linux version release.)

3) *Dominions 4: Thrones of Ascension* (Desura): old-school, very deep, turn-based strategy.

I'd also like to suggest a monthly half-page "Linux Games Update" column.
—Richard

Richard, thanks for the links and also for the column idea. If nothing else, I'll try to be more game-aware in my UpFront pieces. Please don't hesitate to send me game recommendations.—Shawn Powers

Using MySQL for Load Balancing and Job Control under Slurm

Regarding Steven Buczkowski's "Using MySQL for Load Balancing and Job Control under Slurm" in the August 2015 issue: I have a question on the queuing system to process big data.

If you're using MySQL on different nodes, how are you dealing with each system's data locking? Or, is the DB on the head node and you're just running via client access on the nodes? It would be good to hear your setup, but I like the idea.

—Gary

Steven Buczkowski replies:
Ahhh, you got to the problem faster than I did. I was still a fairly naïve slurm user when I originally put this idea together, and it turns out, I was trying to populate the db under slurm (that is, on multiple nodes, simultaneously...). Somehow, I managed to escape having that come back and bite me as resource contention for some time. I don't understand how I managed that for so long.

When I did start having trouble, I tried a number of mutex-like things to let the first instance of my process populate the db while the other spawned instances blocked until this was done: all very ugly and very inefficient use of the cluster resources.

In the end, the cleanest approach was to do the db population outside the sbatch call (that is, on the head node) and start the parallel processing only once the stack table was built. The problem here is that I was keying the table off the slurm process ID, and on the head node, we don't have access to that yet. The solution was to have a wrapper script that first generates a random number to use as a slurm procID proxy to build the table, then kicks off the table build with that ID, then starts sbatch and passes that random number in to my processing script as either a command-line parameter or as an environment variable via sbatch's `--export` *keyword. The table pop process keys off this random number, but the table updates when closing out a process can add the real procID if you want that for accounting purposes.*

Welcome to Issue 10000000? Really?

I'm probably not the first one to write about it, but it would seem that Shawn Powers erred when he said "Welcome to issue 10000000" (August 2015 issue). He most likely meant "Welcome to issue 100000000." It's okay, we all make miskates...uh mistakes, I meant.

—Pierre

Binary Fail

No matter how many times I count

them, I can find only seven zeros behind the one. I read that as binary 10000000 or hexadecimal 80, aka decimal 128. But the cover says Issue 256. I could not believe that slipped by so I counted them again—still seven. Ooops.
—Bonnie Packert

Pierre and Bonnie, never before have I gotten so much e-mail as when I miscounted zeros in issue 256! Truly, people e-mailed me, Tweeted at me and even instant-messaged me! Of course, you are both correct, and I made a silly error. I guess if I'm going to make a mistake, I'd rather it be something silly like that as opposed to something in an instructional article.

But yes, I totally dropped the ball on that one—or maybe the zero!—Shawn Powers

Gettin' Sticky with It

Regarding Shawn Powers' "Gettin' Sticky with It" in the June 2015 UpFront section: just a comment about listing the attributes of a directory. In the article, the following command was given:

```
ls -l / | grep tmp
```

You could use:

```
ls -ld /tmp
```

The `-d` option makes it list the directory and not the contents.

Linux Journal is the only mag that I've continued enjoying since its beginning. Thanks for the good work of keeping Linux and its special features in focus.
—Dan

Cool, thanks Dan! I never knew about the `-d` *flag. Also thanks for the great compliment!—Shawn Powers*

"What Does Fast Mean?"

I think you made a mistake here [see Reuven M. Lerner's "What Does Fast Mean?" in the September 2015 issue]:

> Let's now replace that high-speed wire with a satellite link. Suddenly, because it takes time to send the data from one network to another, you have decreased your latency while keeping your bandwidth identical.

Shouldn't this be *increased* your latency? Other then that, it was a great read.
—Vic

Reuven M. Lerner replies: *Good catch! You're totally right.*

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!

The open source world comes to Raleigh, NC

presented by IT-oLogy™ *in partnership with* opensource.com

2015 ALL THINGS OPEN®

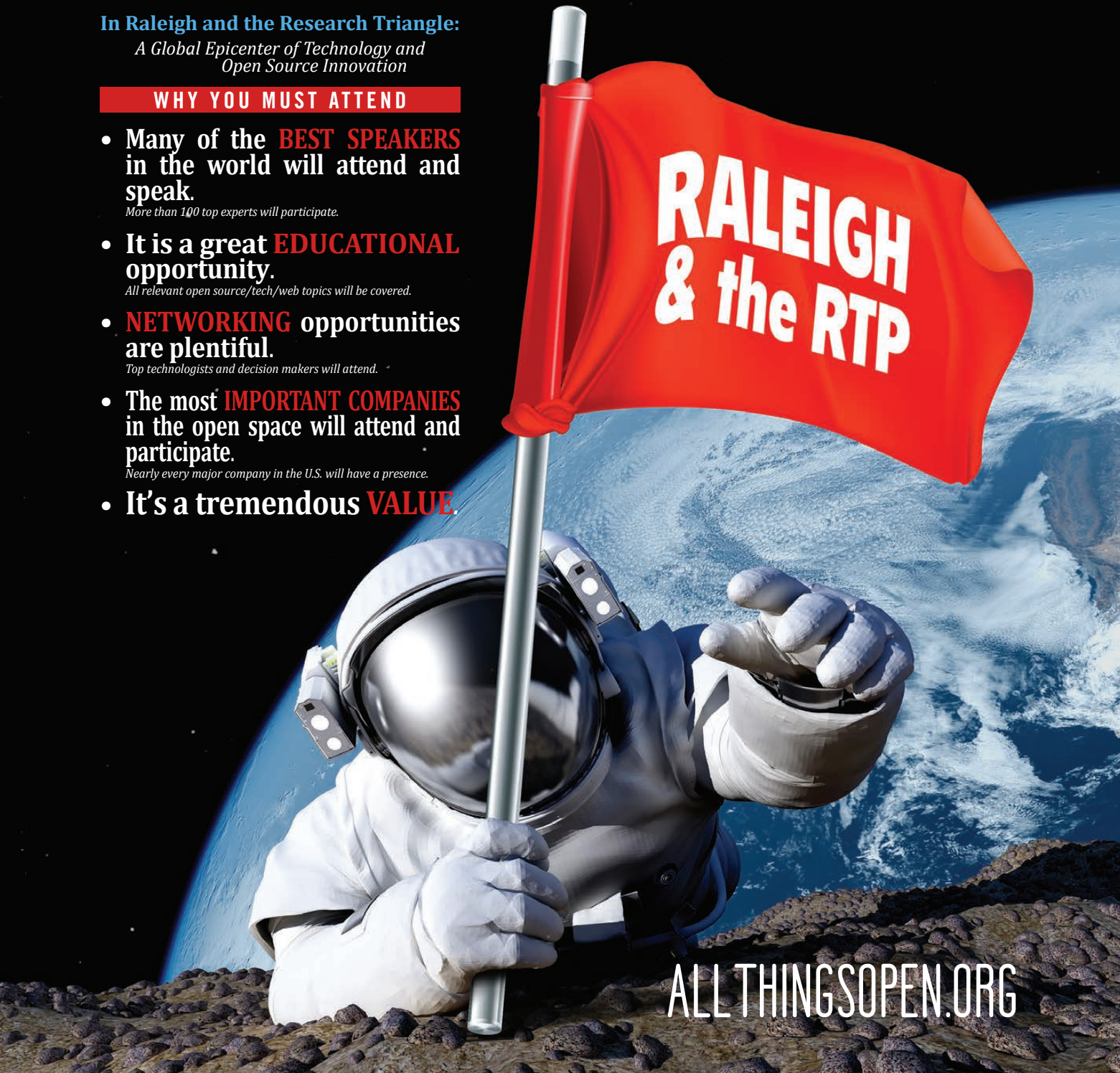
OCTOBER 19 & 20 | RALEIGH CONVENTION CENTER

In Raleigh and the Research Triangle:

A Global Epicenter of Technology and Open Source Innovation

WHY YOU MUST ATTEND

- Many of the **BEST SPEAKERS** in the world will attend and speak.
  More than 100 top experts will participate.
- It is a great **EDUCATIONAL** opportunity.
  All relevant open source/tech/web topics will be covered.
- **NETWORKING** opportunities are plentiful.
  Top technologists and decision makers will attend.
- The most **IMPORTANT COMPANIES** in the open space will attend and participate.
  Nearly every major company in the U.S. will have a presence.
- It's a tremendous **VALUE**.



ALLTHINGSOPEN.ORG

diff -u
WHAT'S NEW IN KERNEL DEVELOPMENT

Over time, memory can become more and more fragmented on a system, making it difficult to find contiguous blocks of RAM to satisfy ongoing allocation requests. At certain times the running system may compact regions of memory together to free up larger blocks, but **Vlastimil Babka** recently pointed out that this wasn't done regularly enough to avoid latency problems for code that made larger memory requests.

Vlastimil wanted to create a new per-CPU dæmon, called **kcompactd**, that would do memory compaction as an ongoing system activity.

The basic objection, voiced by **David Rientjes**, was that creating a whole new thread on all CPUs carried its own overhead issues. He suggested having one of the other per-CPU threads simply take on the additional memory compaction responsibilities. He identified the **khugepaged** dæmon as the best candidate.

Vlastimil actually had identified khugepaged as a candidate and rejected it, on the grounds that khugepage dealt only with **THP** (Transparent HugePages) memory use cases. These are an abstraction layer above regular memory allocation, so it wouldn't cover all possible cases, only user code that dealt with THPs.

David argued that THP allocations were where most compaction problems occurred, and that other allocation systems, like the **SLUB** allocator (used for highly efficient kernel allocations), were not part of the problem.

Eventually, it came out that David actually envisioned two forms of memory compaction. The first would be a periodic compaction effort that would happen regardless of the state of RAM. The second would be a compaction effort that would be triggered when particular regions of RAM were detected as being overly fragmented. By splitting these two forms of memory compaction from each other, David felt it would be possible to piggy-back various pieces of functionality onto different existing threads and avoid having to create a new per-CPU thread in the kernel.

A final design did not get hashed

out during the discussion, but no one seemed to be saying that memory compaction itself was a bad goal. The question always was how to implement it. **Mel Gorman** even suggested that a fair bit of the work could be done from user space, via the **SysFS** interface. But, that idea wasn't explored during the discussion, so it seems that only technical obstacles could get in the way of some background memory compaction going into the kernel.

One problem with enabling the **CONFIG_TRACING** option in the kernel, as **Tal Shorer** recently pointed out, is that it would enable absolutely every tracepoint, causing a significant performance penalty. It made more sense, he felt, to allow users to enable tracepoints on just the subsystems they were interested in testing.

He posted a patch to do this. Or rather, he posted a patch to ditch the old system and allow users to enable tracepoints on only the **GPIO subsystem**. He picked GPIO, he said, as a test case. If it met with approval, he offered to submit patches for all the rest of the subsystems.

Because of the overall kernel development cycle, it took a couple weeks for his patches to make the rounds. The new merge window had opened, so folks like **Steven Rostedt**, who ordinarily would be the one looking over Tal's submission, were too busy for any new code, until after the merge window had closed again.

Once that was ironed out though, he seemed to have mostly positive feedback for Tal. It looks as though tracepoints soon will be per subsystem, rather than kernel-wide.

In order to allow non-root users to write good system monitoring software, **Prarit Bhargava** wanted

to expose the **MSR** values to non-root users, on a read-only basis. MSR registers are an **Intel**-specific set of registers that Intel originally intended for its own debugging purposes and made no guarantees that future chipsets would provide the same values. But over time, those registers seem to have coalesced around debugging and monitoring features, and the Linux kernel does expose them to the root user.

Prarit felt that allowing read-only access would avoid any potential security issues, because users would be unable to mess around with the actual values of those registers. But as other folks pointed out, the dangers of the MSR registers also existed in the kernel objects and regions of memory they exposed. Even read-only access could be sufficient for a hostile user to gain enough information to attack a running system successfully.

So, working with **Andy Lutomirski** and **Pavel Machek**, Prarit wrote a **PMU** (Performance Monitoring Unit) driver that would expose only a specifically whitelisted set of MSR data to users. This way, they could write their system monitoring software without risking a new attack, and if Intel changed the MSR registers in future chips, the changes would need to be vetted and the whitelist would need to be updated before any of that altered data would be exposed to regular users.

As an example of the importance of this particular issue, **Len Brown** mentioned during the discussion that **Lawrence Livermore National Laboratory** was keenly interested in the design and outcome of Prarit's efforts. The folks there wanted a secure and efficient way to access those MSR registers, and this work would provide it.—**ZACK BROWN**

They Said It

It's a funny thing about life; if you refuse to accept anything but the best, you very often get it.
—W. Somerset Maugham

Self-development is a higher duty than self-sacrifice.
—Elizabeth Cady Stanton

Mistakes are part of the dues one pays for a full life.
—Sophia Loren

A moment's insight is sometimes worth a lifetime's experience.
—Oliver Wendell Holmes Jr.

Cheese—milk's leap toward immortality.
—Clifton Fadiman

usenix

LISA15

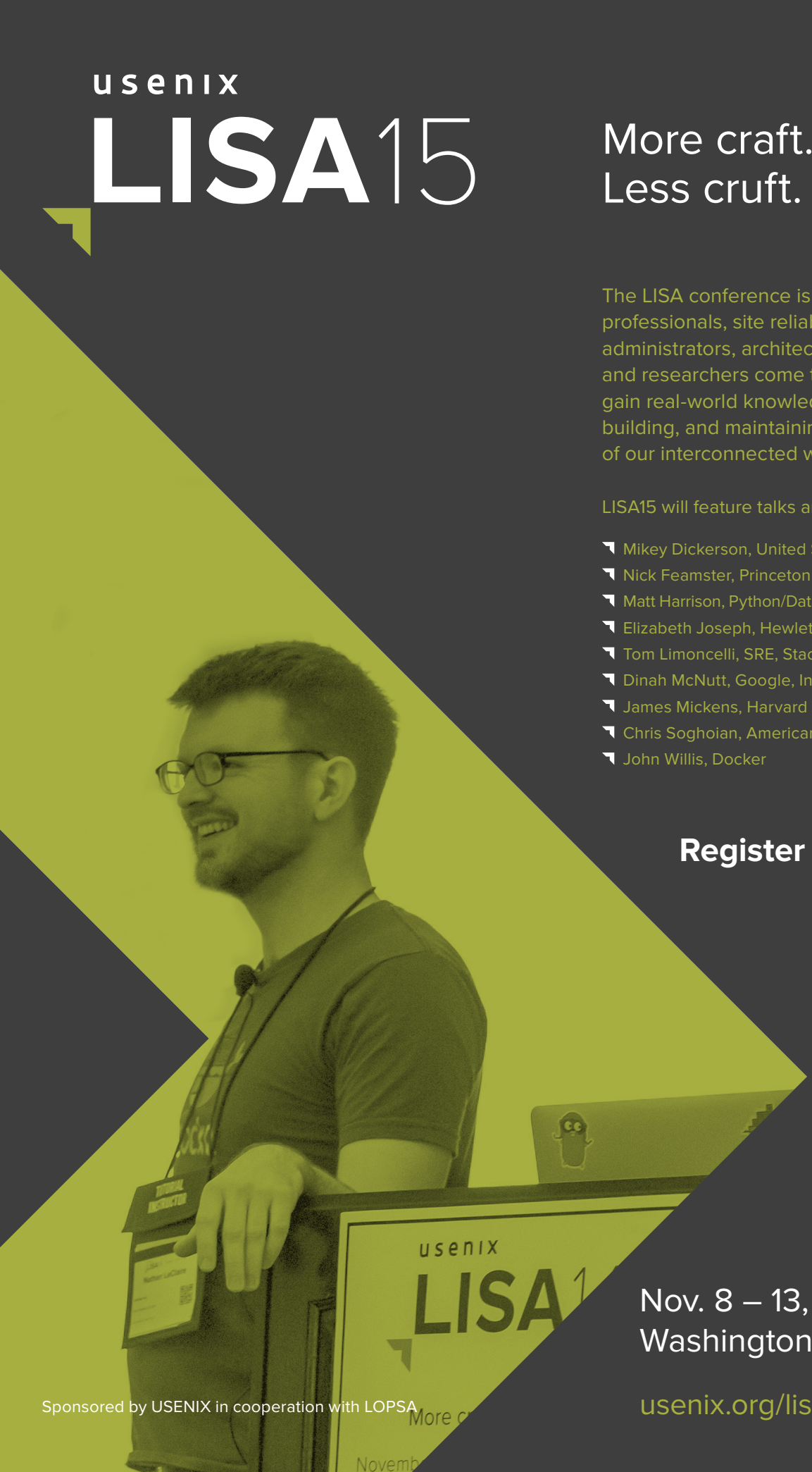
More craft. Less cruft.

The LISA conference is where IT operations professionals, site reliability engineers, system administrators, architects, software engineers, and researchers come together, discuss, and gain real-world knowledge about designing, building, and maintaining the critical systems of our interconnected world.

LISA15 will feature talks and training from:

- Mikey Dickerson, United States Digital Service
- Nick Feamster, Princeton University
- Matt Harrison, Python/Data Science Trainer, Metasnake
- Elizabeth Joseph, Hewlett-Packard
- Tom Limoncelli, SRE, Stack Exchange, Inc
- Dinah McNutt, Google, Inc
- James Mickens, Harvard University
- Chris Soghoian, American Civil Liberties Union
- John Willis, Docker

Register Today!



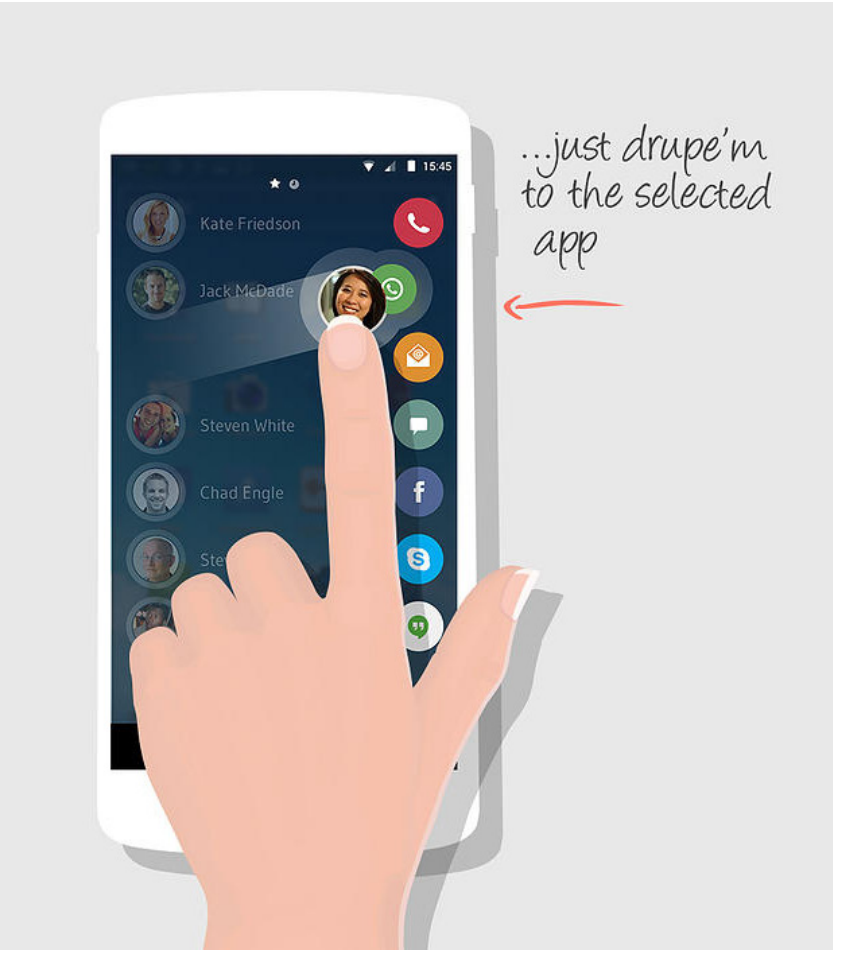
Nov. 8 – 13, 2015
Washington, D.C.

usenix.org/lisa15

Sponsored by USENIX in cooperation with LOPSA

Android Candy: Who Ya Gonna Call? Anyone!

I have a love/hate relationship with the contact manager on my phone. I absolutely love having all of that information available, and I love even more that it syncs with my Google contacts. What I don't love, however, is the cumbersome way you have to scroll around looking for who you want to call. Yes, I normally just look at my recent calls to find who I want, but it seems like there should be a better way to look at my contacts.



(Screenshot from http://getdrupe.com)

Enter Drupe. It feels a bit like a mix between a game and a standardized test's matching exercise. That might sound like an odd comparison, but Drupe makes it easy not only just to call, but also to communicate with contacts with a simple drag. Want to call Mom? Drag her face to the phone icon. Want to text your significant other? Drag his or her face to the SMS icon. (Okay, maybe "drag her face" sounds a bit violent.)

If you like a graphical, easy way to start communicating with your contacts, I urge you to give Drupe a try. It's completely free, supports a huge number of protocols, and it even offers free themes so you can change the way it looks.

—SHAWN POWERS

Protection, Privacy and Playoffs

I'm not generally a privacy nut when it comes to my digital life. That's not really a good thing, as I think privacy is important, but it often can be very inconvenient. For example, if you strolled into my home office, you'd find I don't password-protect my screensaver. Again, it's not because I want to invite snoops, but rather it's just a pain to type in my password every time I come back from going to get a cup of tea. (Note: when I worked in a traditional office environment, I did lock my screen. I'm sure working from a home office is why I'm comfortable with lax security.)

Somewhere I don't like to slack off on security is when I'm out and about, especially on someone else's network. Make that a public Wi-Fi network, and my paranoia level skyrockets. I always have a VPN set up at home so I can connect remotely, but sometimes my home bandwidth is just too slow for comfortable remote use. Recently I paid $39.95 for a year subscription to http://www.privateinternetaccess.com, and I couldn't be any happier with my decision.

PIA offers PPTP, OpenVPN, L2TP/IPSec and the ability to connect up to five devices at the same time. That means I can protect my network (I don't really trust Charter) and connect my laptop while on the road. PIA offers unlimited bandwidth, it doesn't log traffic, and even the account purchasing is anonymous. You can buy your subscription with a Starbucks gift card if you want!

One of my favorite features, however, is that with the option of multiple VPN gateway locations, you can mask your traffic to get around blackout restrictions. For me, that means I can watch Tigers baseball games normally unavailable to me in the MLB app by routing my traffic through the West coast. The $40 price tag seemed pretty steep at first, but with the amount of use I've gotten out of PIA, I'll be signing up again next year without question. You get a seven-day money-back guarantee, so there's no reason not to try it. Visit http://www.privateinternetaccess.com for more details!—**SHAWN POWERS**

Science on Android

I have covered a lot of different scientific packages that are available under Linux in this space, but the focus has been on Linux running on desktop machines. This has been rather short-sighted, however, as lots of other platforms have Linux available and shouldn't be neglected. So in this article, I start looking at the type of science you can do on the Android platform. For my next several articles, I plan to include occasional Android applications that you may find useful.

The first application I want to explore is Maxima for Android (https://sites.google.com/site/maximaonandroid). Maxima has been under development since the 1960s at MIT, and it continues to this day under an open-source license. It is a full computer algebra system, written in Common Lisp. Luckily, Embeddable Common Lisp has been ported to Android, so this provides the Lisp engine needed to run the Maxima engine.

Installing Maxima on your Android device is a two-step process. The first step is to install the base application from the Google Play store. Once it is installed and you run it for the first time, the application will unpack the bulk of the application. Because this section is so large, the app will ask you where to unpack it, so you can put it on some other media if you don't have a lot of room in internal storage.

When you initially start Maxima, you get a display of the licensing information and the version number, and then you are presented with an input prompt labeled as "(%i1)". Depending on the device you are using and the age of your eyes, you may have a hard time seeing the output displayed on the screen. If so, you can zoom in and out with a two-finger pinch or zoom, just like in other Android applications. Your other option is actually to change the font size. You can use the command `textSize:XX`, where XX is the size you want to use for the font in pixels. The suggested range is 10–50 pixels.

At the bottom of the screen, you will find a text entry box outlined in orange. This is where you enter the Maxima commands you want to run. Once you have entered your command and press Enter, it is displayed in the main window, along with the results. The MathJax library handles the pretty-print display of

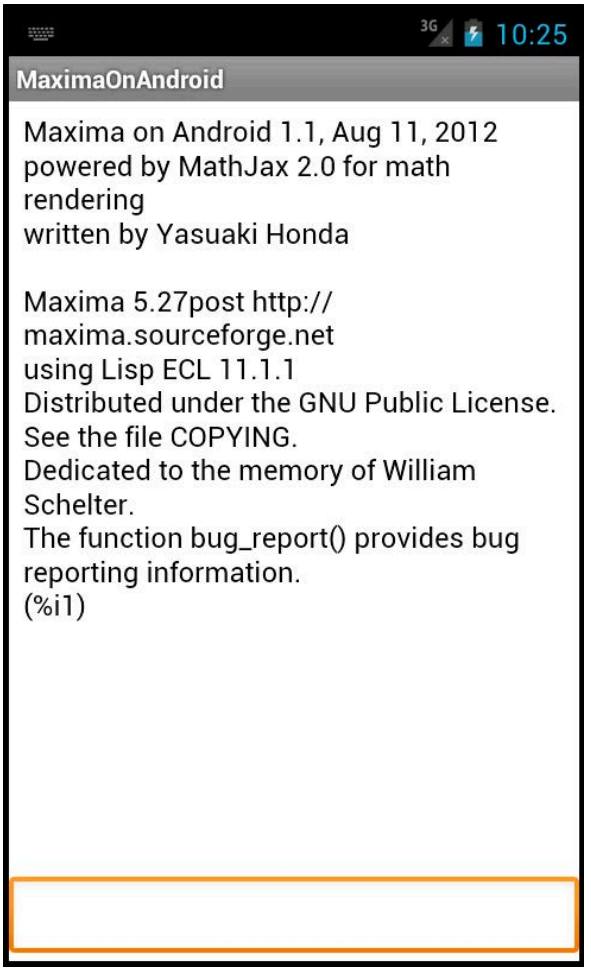


Figure 1. When you start Maxima, you get the standard license and version information.

this mathematical information. The history mechanism used in order to re-use previous commands is very intuitive. When you tap a previous command, it is copied and pasted into the text input command box, ready for you to make any necessary edits before executing it again.

The Android version of Maxima includes a full set of documentation that is available by tapping on the menu icon and selecting the "Manual" option. A lot of examples are available in each section of the manual that you may want to try. Assuming that this would be a common thing people would want to do, you simply can tap the example you want to work with, which will copy the example into the command text input box. This way you can find an example that is close to what you want to try to do and easily copy it into the command box. You then can edit it and make any required changes before running it. This really can speed up any research work you are trying to do with Maxima.

There is also a function named `example()`. If you call `example` without any parameters, it will generate a list of all of the subjects that have examples provided. You then can look at the examples for a specific subject by calling `example` with the named subject. For instance, you can learn about arrays with the command `example(arrays)`. This will pull up and run a series of examples on how to use arrays within Maxima.

The other useful function for speeding up research is the ability to load files of Maxima commands. Maxima for Android automatically looks in the main Download directory when you try to load files. Also, files

with the ".txt" filename ending are found and loaded without having to include the file ending. This means if you have a file named "my_script.txt" in the Download directory, you can load it with the command `load(my_script)`. If you use Dropbox and synchronize the Download directory, you easily can move files back and forth between your Android device and your desktop.

One major area that is not completely implemented yet is the graphical system for plots and graphs. However, new features are being added with each new version. The graphical functions actually are provided by gnuplot. The functions currently implemented are `plot2d`, `plot3d`, `implicit_plot`, `contour_plot`, `draw`, `draw2d` and `draw3d`. These commands pop open a new window to display the results of the `plot` command.

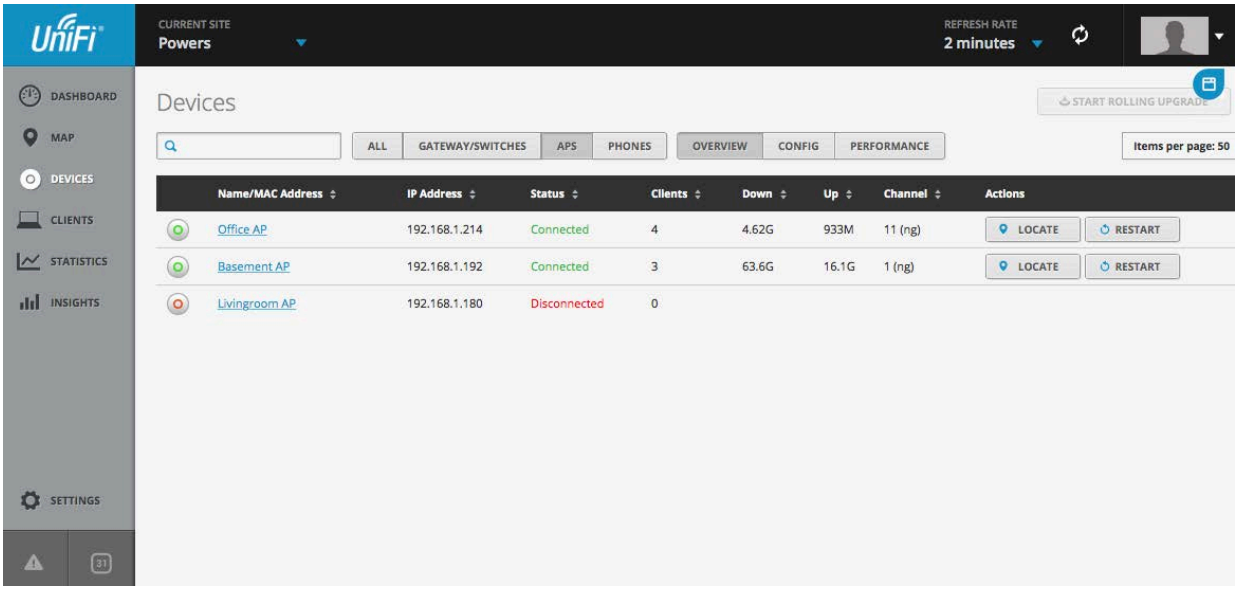
Because everything is full screen on an Android, you need to tap the back icon to get back to the main Maxima window. If you want to see the plot again, you can tap on the menu and select the "Graph" option. This will re-open the last plotting window.

Another major issue is the lapack library. Trying to load it will cause an error, so you won't have access to the optimized linear algebra functions provided through lapack.

When you are doing complicated research, you may need to stop part way through the process and pick it up again at a later time. This is handled by the concept of a session. Tapping the menu icon, you can select the "Session" option. This will pop open a sub-menu where you can choose either to "Save", "Restore" or "Playback" your session. You have just one saved session state at a time, however, so it is really useful only when you need to stop in the middle of a workflow.

If you are working on a larger project where you are using the same libraries for an extended period of time, you can set initialization code that gets run whenever Maxima starts. This code needs to be stored in the file /data/local/tmp/maxima-init.mac. Any arbitrary Maxima code can be placed here, so you can initialize a pretty complicated environment if you like.

Now you can carry Maxima around in your pocket, ready to work out all of those troublesome problems that come up in everyday life. So, you won't have any excuse for not solving the equations you need in order to plot out your space travel, all on the train ride to work. Just promise to be good, and don't try to use it on your next physics exam.**—JOEY BERNARD**



Roll Your Own Enterprise Wi-Fi

As you can tell by my Wi-Fi focus in The Open-Source Classroom this month, I really love wireless networking. I've implemented wireless solutions for schools on a shoestring budget, and I've helped plan campus-wide rollouts of redundantly controlled enterprise solutions. The one thing I really like about the enterprise solutions is the single point of management. The problem is those controllers and enterprise-capable access points are so darn expensive! That's why I love Ubiquiti.

The UniFi line of products from Ubiquiti is affordable and reliable, but the really awesome feature is its (free!) Web-based controller app. The only UniFi products I have are wireless access points, even though the company also has added switches, gateways and even VoIP products to the mix. Even with my limited selection of products, however, the Web controller makes designing and maintaining a wireless network not just easy, but fun!

The Web controller runs happily on a little Atom Web server, and if it happens to go off-line, the access points keep working independently. If you'd like robust hardware with a powerful Web controller application, but you're not ready to invest the thousands of dollars for a traditional enterprise solution, check out the UniFi product line (https://www.ubnt.com). If you're half as nerdy as I am, you won't regret the decision!—**SHAWN POWERS**

Non-Linux FOSS: Code Your Way To Victory!

One of my favorite things about grade school was when the teacher would review for a test by playing *Jeopardy*. I'm pretty old, so my version of classroom *Jeopardy* was done on a chalkboard with the teacher reading answers from index cards, but the new computer-based versions I see in schools are at least as cool. There's just something about learning while playing games that is awesome.

Learning to write code is a chore that often is considered boring. Heck, the only way I'm able to slog through the process is to have a project in

mind. I need to be solving a problem in order to find the motivation to write the code, and learning the "tools" really becomes part of the enjoyment. Thankfully, I'm not alone in my desire to tie learning into fun. The folks over at CodeCombat have created a gaming system that uses proper coding techniques to help your hero solve a quest.

The "game" actually feels like a cross between *The Incredible Machine* (or *Crayon Physics* if you're too old to remember *TIM*), and the old LOGOS turtle we guided around the screen in the 1980s and 1990s. Some of the gaming elements seem like a stretch (magic boots give you the ability to write code that moves your character), but the goofiness adds to the fun a bit. Plus, the graphics are really well done, and the background music rocks.

CodeCombat is free to use, and it includes 100 or so coding exercises. If you're really into the game and want to improve your coding skills further, there is a $9.99/month subscription that opens up video tutorials, more levels and so on. There's even a price break if you buy multiple subscriptions for those little coders in your life.

Even though the game itself isn't FOSS, and actually runs in a Web browser instead of a particular platform, I put Code Combat in our Non-Linux FOSS spot this month because it teaches code. Teaching people to write code is the first step in creating a FOSS world, and for that reason, I also award Code Combat this month's Editors' Choice Award. Check it out today, and see if it's a good fit for you or other potential coders in your life: https://codecombat.com.

—SHAWN POWERS

REUVEN M. LERNER

Parts of a Slow Web Application

Where are the potential bottlenecks in scaling a Web app, and what can you fix?

In my last article, I started discussing how to optimize a Web application, looking at the different aspects of an app and where the slowness might come from. I described several different factors that can lead to a Web application feeling slow for the end user: network speed and latency, server application speed, client-side loading and rendering, and finally, the execution of the client-side JavaScript programs. Each of those factors contributes to the feeling of speed that a user gets when using a Web application, and it's our job as engineers to try to reduce them.

So in this article, I cover a number of the places where you can look to try to reduce the problematic times that I described—that is, specific areas of your application that might be slow and might well need some tuning or improvements. In my next article, I'll look at specific programs and techniques you can use to identify problematic spots in your applications and, thus, help them improve.

Where Do You Look?

One of the most infuriating things someone can say to a programmer is "It doesn't work." What doesn't work? What did it do yesterday? What causes the computer to stop working? When people say to me that their system "doesn't work", and then expect me to find the problem, I suggest that such a statement is similar to going to a doctor's office and saying, "It hurts me somewhere."

Similarly, if you're going to find the bottlenecks and slow-moving parts in your system, you're going to need to look at various parts of it and consider

It used to be that the major consideration for software developers was the speed of the CPU on which programs ran. However, for modern Web applications, I/O speed and memory are far more important.

how much they might be contributing to the slowness. Only after thinking about the individual parts and understanding what about each of them might be slow, can you then try to fix any problems you might have.

Infrastructure

I'm not a hardware guy; I often joke that I reach my limits when I change a light bulb. That said, there's no doubt that understanding your hardware, at least a bit, is going to help you optimize your software.

It used to be that the major consideration for software developers was the speed of the CPU on which programs ran. However, for modern Web applications, I/O speed and memory are far more important.

Your application—especially if it's written in a high-level language, and even more so when the application is under heavy load—will consume a large amount of memory. As a result, you're going to want to load up on RAM. This is also true for your database; the best way to kill a database's performance is to start using virtual memory. If your server is using the disk because it ran out of RAM, you're going to suffer from a massive performance hit.

By contrast, if you can give your database a large amount of memory, it will be able to cache more data in RAM and not have to go to disk nearly as often. If you're using PostgreSQL for a database, setting the "shared buffers" and "effective cache size" parameters are crucial, in that they tell the database how much memory it can expect to use. This helps the PostgreSQL server figure out whether it should jettison data that it already has cached or load additional data into the shared buffers.

This also points to the advantages of having multiple servers, rather than a single server. If you're running a small Web site, the RAM

requirements of your HTTP server and your database server might well clash. This especially will manifest itself when you get a lot of traffic. Your HTTP server will use more memory, but so will your database, and at a certain point, they might collide. Using more than one computer not only allows you to scale out the Web and database servers more easily, but it also ensures that neither will interfere with the other.

The other factor to consider, as I mentioned above, is I/O. Again, I'm not a major hardware expert, but you want to consider the speed of the disks that you're using. Today, as people increasingly move to ritualized servers in which the underlying hardware is abstracted away, it's sometimes hard to evaluate, let alone choose, the hardware on which your systems are running. Even if you can't do that though, you can and should try hard to avoid putting any production-level system on a shared machine.

The reason is simple. On a shared machine, the assumption is that each application will play nicely with the others. If one application suddenly starts to crunch through the disk a great deal, everyone's I/O suffers. I experienced this when working on my PhD dissertation. My software would back up its database once per hour, and the managers of my university's computer cluster told me that this was causing unacceptably slow performance for other users. (We found a way to back up things in a way that was less resource-intensive.)

The question that many people used to ask about servers was "Buy or build?"—meaning, whether you should create your own special-purpose server or buy one off the shelf. Today, very few companies are building their own servers, given that you're often talking about commodity hardware. Thus, the question now is "Buy or rent?"

I must say that until recently, I was of the opinion that having your own server, on which you have relatively full control, was the way to go. But I must admit that after working on several scalability projects, I'm warming up to the idea of deploying a swarm of identical VMs. Each individual VM might not be very powerful, but the ability to scale things up and down at the drop of a hat can more than outweigh that need.

The bottom line is that when you're looking into servers, there are (as always) many different directions

to explore. But if you think that your system might need to scale up rapidly, you should seriously consider using a "cloud" platform. More important than the CPUs is the amount of RAM and ensuring that your virtual machine is the only user of your physical machine.

Oh, and how many servers should you set up? That's always a hard question to answer, even if you know how many users you expect to have visit. That's because servers will behave differently under different loads, depending on a wide variety of factors. No matter what, you should give yourself a comfortable margin of error, as well as have contingency plans for how to scale up in the case of wildly successful PR. In my next article, I'll discuss some strategies for figuring out how many servers you'll need and what sort of margin to leave yourself.

HTTP Server

Now that you have some hardware, you need to consider your HTTP server. This is a matter of taste, personal preferences and considerable debate, and it also depends in no small part on what technology you're using. For many years, I have used Apache httpd as my Web server—not because it's super fast, but because it's very easy to configure and has so many plug-in modules available. But, even I must admit that nginx is far more scalable than Apache. The fact that Phusion Passenger, a plugin for both Apache and nginx, works with both Ruby and Python and is trivially easy to install convinced me to switch to nginx.

Whereas Apache uses multiple threads or processes to handle its connections, nginx does so using a single thread, in what's known as the "reactor pattern". As such, it's generally much more scalable.

If you're trying to remove potential bottlenecks in your system, then having a high-powered HTTP server is going to be necessary. But of course, that's not enough; you also want the server to run as quickly as possible and to be taxed as little as possible.

To make the server run faster, you'll want to examine its configuration and remove any of the modules you don't really need. Debugging is great, and extra features are great, but when you're trying to make something as efficient as possible, you're going to need to trim anything that's not crucial to its functioning. If you're using Apache and are including modules that make debugging easier and faster, you should remove them, keeping them (of course) in your development and staging systems.

You also want to tax your HTTP server as little as possible, so that it can concentrate its efforts on servicing HTTP clients. There are a number of ways to do this, but they all basically involve ensuring that requests never get to the server. In other words, you'll want to use a variety of caches.

Within your application, you'll want to cache database calls and pages, so that someone who requests a page on your system turns to the HTTP server only if it's really necessary. Modern frameworks, such as Rails and Django, are designed to let you cache pages in an external system, like memcached or Redis, such that requesting /faq from your server will be served from the cache instead.

Beyond the caching that you'll want to do in your application, you probably will want to put a front-end Web cache, such as Varnish, between your servers and the outside world. In this way, any static asset (such as JavaScript, CSS or images) that users request will come from that cache, rather than having to go to the server.

Going one step further than this, in a move that a growing number of large sites are making, you could (and probably should) use a content distribution network (CDN), on which your static assets reside. In this way, someone who goes to your site hits your server only for the dynamic parts of your application; everything else is served by a third party. Your server can spend all of its time worrying about the application itself, not all of the stuff that makes it pretty and functional for the end user.

Database

Another point of great debate, as well as a bottomless pit (or, if you're a consultant, an endless opportunity!) of work, is that of databases. Whether you're using a relational database, a NoSQL database or a combination of the two, all databases are massively complex pieces of software, and they need a great deal of configuration, care and attention.

Actually, that's not true. You can, in many cases, get away with not tuning your database configuration. But if you want to scale your system up to handle high load levels, you're going to want to keep as much of the database in memory as you can and tune your queries as much as possible.

You're also going to want to ensure that the database is doing the best job it can of caching queries and then of results. Databases naturally do this as much as they can; keeping query results in memory is a way to boost the speed. But many databases let you configure this memory and also tune the ways in

which it is allocated. PostgreSQL, for example, uses its statistics (collected via VACUUM) so that it'll know what should stay in memory and what can go. But at the application level, you can cache queries and their results, allowing the application to bypass the database completely and, thus, lighten its load.

There is another complication when it comes to databases, namely the fact that so many modern Web application frameworks generate SQL for you automatically using object-relational mappers (ORMs). It's true that in most cases ORM-generated SQL is more than adequate for the task, and that the overhead of generating SQL on the fly and of the many object layers required to do so, is worth the cost.

But, there are numerous cases in which ORM-generated SQL was not very efficient, often because the programmer's assumption didn't match that of the ORM. A classic example of this is in Ruby on Rails if you retrieve numerous objects from the database after a query. From the Ruby code, it feels like you're just iterating over a large number of objects. But from the SQL side, each object over which you iterate is firing off SQL queries, potentially clogging the database.

Using a slow-query log or logging (as PostgreSQL allows) all queries that take longer than a certain minimal threshold is a great way to start looking for the things that are taking a long time in the database.

But even then, you might not find your performance problems. I recently was working with clients, helping them optimize their database, which we found was taking a very long time to execute a lot of queries. And yet, when we looked in the logs, nothing was written there. The problem wasn't that each individual query was taking a long time, but that there was a huge number of small queries. Our optimization didn't speed up their query, but rather replaced a "for" loop, in which the database was queried repeatedly, with a single, large query. The difference in execution speed was truly amazing, and it demonstrated that in order to debug ORM problems, it's not enough to know the high-level language. You really do need to know SQL and how the SQL is being executed in the database.

In some cases, there's nothing you can do to prevent the application from hitting the database—and hitting it hard with lots of queries. In such cases, you might well want to use a master-slave model, in which all of the read-only queries are directed to a set of slave (read-only) databases, and write queries are directed to a master

server. Master-slave configurations assume that most of the queries will be reading, not writing, and if that's the case for your system, you're in luck.

If that's not the case for your database, and master-slave doesn't do the trick, your solutions will be limited by the database you're using. Some offer master-master replication, giving you multiple databases to which you can send write requests. Some of them, especially in the NoSQL world, shard automatically. But no solution is perfect, and especially when it comes to master-master replication, things can get a bit dicey, even for the most experienced database hands.

The bottom line is that your database is quite possibly going to be the biggest bottleneck in your system. Try to keep the application away from it as much as possible, caching whatever you can. And, try to keep the database tuned as well as possible, using the current best practices for the system you're using, so that it'll be working at peak performance.

Conclusion

In my last article, I talked about the basic, high-level places where there might be problems with scaling a Web application. Here, I discussed, in greater detail, what sorts of things you can do when your application is showing, or might be showing, scaling issues. Next month, I'll look at some specific tools and techniques that you can use to understand how scalable your application is *before* hordes of users come and use it.■

Reuven M. Lerner trains companies around the world in Python, PostgreSQL, Git and Ruby. His ebook, "Practice Makes Python", contains 50 of his favorite exercises to sharpen your Python skills. Reuven blogs regularly at http://blog.lerner.co.il and tweets as @reuvenmlerner. Reuven has a PhD in Learning Sciences from Northwestern University, and he lives in Modi'in, Israel, with his wife and three children.

Resources

The 2nd edition of the *Art of Scalability*, by Martin Abbott and Michael Fisher (published by Addison-Wesley), describes in great detail the different parts of a Web application and the ways you can and should scale it up. They describe a thought process, rather than a set of technologies, for analyzing the architecture of a Web application. If this is a subject that is of interest to you, it's likely worth reading this book.

Learn How To Master Big Data

BigData TECHCON

November 2-4, 2015
CHICAGO

Holiday Inn Chicago Mart Plaza River North

Choose from 55+ classes and tutorials!

Attend Big Data TechCon to get practical training on Hadoop, Spark, YARN, R, HBase, Hive, Predictive Analytics, and much more!

Take a Big Data analytics tutorial, dive deep into machine learning and NoSQL, learn how to master MongoDB and Cassandra, discover best practices for using graph databases such as Neo4j and more. You'll get the best Big Data training at Big Data TechCon!

www.BigDataTechCon.com

People are talking about BigData TechCon!

Great for quickly coming up to speed in the big data landscape.
—Ben Pollitt, Database Enginee, General Electric

There was a large quantity and variety of educational talks with very few sales lectures. It was just informative and inspiring. This was the best conference ever! Get a ticket for 2015!
—Byron Dover, Big Data Engineer, Rubicon Project

A BZ Media Event

DAVE TAYLOR

Words from Letter Sets, Part II

Dave continues to build the block letter word puzzle, exploring algorithms and edge cases.

Last month, I started to explore how to figure out what words we could make from a set of letter cubes, based on a letter that a reader sent me. If you think about the problem, however, you realize that it has more general applicability in games like *Scrabble* and *Words with Friends* too: "Here's a set of letters, some possibly repeating, what words can I make with them?"

Of course, in games where there are interlocking words, there's always a spare letter or two that you're working with above and beyond what's on your rack or in your hand, but the same basic principle holds true, right? In my last article, we ascertained that the fast way to identify possible word candidates for a set of letters was through a rather tricky regular expression:

```
^[word]{length}$
```

You'll doubtless recall that ^ is the special character that matches the very beginning of the line, and $ is the one that matches the last character. In a regular expression (or "regex" as we old-timers call them), anything with a {} notation is a repeat count, so what we're basically looking for here are words that are composed exclusively of the letters in the [] notated set that are {} characters long.

In practice, `^[chicken]{7}$` matches the word chicken (of course), along with any other seven-letter words that can be spelled by using just those letters—which turns out to be "chicken". So, that wasn't a great demo. But, let's say we remove the trailing character, allowing matching words that include all the letters but are longer than just seven characters. Now the results are

We need to explode the word so it's all separate letters, process those letters, then reassemble the word afterward.

a smidgen more interesting:

```
checking
chicken
chickens
chinning
```

You can see that they don't work simply by length, but the first has a "g" that we don't have in the original set, the third an "s", and the last has one "n" more than the original pattern included.

That's why this is a two-step solution!

Let's capture this first portion in code before we proceed, however, keeping in mind that we can simplify the expression by removing duplicate characters (so "chicken" can become "chiken"). Here's my code snippet:

```
length="$(echo "$1" | wc -c)"
length=$(( $length - 1 ))

unique="$(echo $1 | sed 's/./&\
/g' | tr '[[:upper:]]' '[[:lower:]]' | sort | \
uniq | fmt | tr -C -d '[[:alpha:]]')"

regex="^[$unique]{$length}"

echo "Raw word list of matches for letterset $unique:"

grep -E $regex "$dictionary"
```

As you immediately can see, the real work of the script is in removing duplicate letters from the word pattern—a rather crazy-complicated pipe that works like this:

```
echo user specified pattern |
  append a carriage return to each individual letter |
  translate uppercase to lowercase |
  sort the letters alphabetically |
  remove duplicates with "uniq" |
  merge the multiple lines to a single line with "fmt" |
  remove any non-alphabetic characters with "tr"
```

That's a lot of work, but it's a tricky task. We need to explode the word so it's all separate letters, process those letters, then reassemble the word afterward.

In this case, running this with our word of "chicken" (but with the

trailing $ still missing) produces:

```
Raw word list of matches for letterset cehikn:
checking
chicken
chickens
chinning
```

Notice the alphabetically sorted, dupe-removed "cehikn" in the prompt.

So we don't forget, let's go back into the code and change the pattern assembly line to include the $:

```
regex="^[$unique]{$length}$"
```

But first....

Back to Those Letter Blocks

The thing about everything we've written so far is that it assumes we want to use every single letter every time we run the program, and that might not be a valid assumption. For example, perhaps a four-letter word out of five blocks is an acceptable solution, with the spare block put aside for that particular day.

To match that scenario, a single comma will do the work as long as the length parameter in the regular expression indicates the minimum acceptable length. In other words, instead of something like `[chicken]{7}`, use a pattern like this: `[chicken]{5,}`.

The good news? This provides a lot more matches:

```
Raw word list of 5 or longer matches for letterset cehikn:
check
cheek
chick
chicken
chink
hence
niche
niece
```

What I did in the code was change the length calculation always to calculate two less than the actual length—really the only change needed:

```
minlength=$(( $length - 3 ))
```

Well, I guess I tweaked the echo statement too:

```
echo "Raw word list of $minlength or longer matches \
   for letterset $unique:"
```

Still, that's not much of a change and a big boost to the number of possible words that match the given set of characters!

But which actually can be assembled without needing duplicates of letters that aren't available?

Making Sure We Use Only the Available Letters

My first thought on solving this problem involved sorting the letters in the words and comparing, but that works only if the two words are the same length. Consider the pattern "cats" and the word "cast". Sort them by letter and they're the same: "acst". Easy.

But can you spell "chick" with the letters in "chicken"? Can you spell "cheek"? Yes, and no.

The smarter way to manage the process is through a "picking" algorithm where the program goes through each letter of the input pattern (for example, the letters on the blocks) and checks against the potential match. Each time a letter is matched, it's removed. This then makes it easy to identify the situation where there are letters that are used too often, as the double "e" of "cheek" above demonstrates.

If there are letters that aren't axed in the potential word by the time all the letters of the pattern are exhausted, that's a fail too (representing a letter that doesn't appear in the input set), but that already should be caught earlier in the program when we are screening against our letter set.

In other words, the word "cheek" can't be made out of "chicken", but neither can "chicky": there's no "y". But that situation will have been identified in the first half of the program anyway, and the word would never have made it into the test list at this point.

Still, a bit of analysis shows that it's not really a big deal because the algorithm remains the same: start with the test word and the pattern. For each letter in the pattern, remove that letter from the test word if present. When done, the test word should be consumed entirely by the substitution process and no letters should remain standing.

If it meets the minimum length requirement (because "hi" shouldn't be a good match for the letter set "chicken") and has no remaining letters, it's a win! We've found another word that can be made from the letters of the pattern.

And coding it? That's next month. Stay tuned.■

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

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

KYLE RANKIN

AWS EC2 VPC CLI

What's better than three-letter acronyms? Three-letter command-line tools.

There's just something about the fresh start you get with a new job. Both my previous job and my new one began with the opportunity to build a new infrastructure from scratch. In both cases, as is common with startup infrastructure in its early stages, everything was to be built using Amazon Web Services (AWS), specifically using its Elastic Cloud Computing (EC2) infrastructure. Two significant things had changed in the years between the two jobs that led me to a fresh approach to the infrastructure, and those are the things I'm going to explore in this article: Amazon's Virtual Private Cloud (VPC) and the AWS command-line tools.

VPC has been around for some time, and it was around back when I started work on my previous infrastructure, but it was an extra feature. At the time, you defaulted to what Amazon now calls "EC2 Classic", which meant you essentially were sharing an internal IP space with everyone else on EC2. With VPC, you can define different subnets, including ones that have only non-Internet-routable RFC1918 IP addresses. You get to choose which hosts get external IPs and which don't, and you can define not only what subnet a host is in, but you also even can assign it a specific internal IP if you want. Additionally, you have more control over Security Groups (which allow you to assign firewall rules to groups of hosts), and you can filter not only incoming (ingress) traffic but also egress (outgoing) traffic. You even can add or remove security groups from a host after it has spawned—something just not possible in EC2 Classic.

Today, new Amazon accounts are VPC only. It's clear that Amazon is phasing out EC2 Classic not just by calling it "Classic" and making VPC

the default, but also by creating new lower-cost instance types that are available only on VPC. Even though by default the subnets that come with a VPC behave much like on EC2 Classic with public and private IPs, there still are enough differences and potential once you consider private subnets that it forces you to take a fresh approach to how you build things.

The Amazon Command-Line Interface

It may not come as any surprise at all to my regular readers that when I initially got started with Amazon infrastructure, I avoided the Web interface whenever possible and stuck with command-line tools. Although back then there were a number of different libraries (such as the boto library for Python) that you could use to interact with Amazon's API with software, the main command-line interface for Amazon EC2 was known as EC2 API Tools (or the ec2-api-tools package in Ubuntu). It was Java based and provided a way to perform all of the common functions you might perform on the Web interface via commands named `ec2-<some function>`. It wasn't so bad once you got used to it. The primary complaint I had with EC2 API tools was how slow it was between issuing a command and getting a response back.

A year or so ago I discovered that Amazon had developed a new Python-based command-line tool called AWS CLI and made a mental note to try it when I had a chance. Since I already had scripts in place that took advantage of the EC2 API tools, and they worked, I didn't have any compelling reason at the time to switch. Now that I was faced with building new scripts for a new infrastructure, however, it seemed as good a time as any to try it out.

Using AWS CLI

I'm going to skip the details of how to install and configure AWS CLI since that's already well documented on the main page for Amazon CLI, and it's probably already packaged for your Linux distribution (and if not, you can do `pip install awscli` to get it). Although you can use AWS CLI for a number of different AWS APIs, I'm going to stick to how it compares to EC2 API tools. Instead of every command starting with `ec2-`, every command starts with `aws ec2` followed by the command. Fortunately, most of the commands match in both environments, so where before you might have typed:

```
$ ec2-describe-instances
```

Now you type:

```
$ aws ec2 describe-instances
```

Unfortunately, the similarities pretty much end there. Where with `ec2-describe-instances` you could just append a list of IDs, with AWS CLI, you need to use the `--instance-ids` argument first:

```
$ aws ec2 describe-instances --instance-ids i-someid
```

Another difference between the two tools is the output. You can select between a table, text and JSON output with AWS CLI; however, the text output ends up being quite different from the format in EC2 API tools. This means if, like me, you have tools that parse through that output, you'll need to rework them.

Check Whether a Host Exists

For instance, I wrote a wrapper script to spawn new EC2 hosts that captures all of the common options I might pass along the command line based on a host's name. One check I perform before I spawn a host though is to test whether I already have an instance tagged with that host's name. So something like this:

```
ec2-describe-tags --region $REGION | cut -f5 |
 ➥egrep -q "^$MYHOST$"
```

becomes something like this:

```
aws ec2 describe-tags --region $REGION | grep Name |
 ➥grep instance | cut -f5 | egrep -q "^$MYHOST$"
```

Find the Security Group ID by Name

I had to make another change to my script when it came to assigning security groups. With EC2 Classic, you could refer to security groups by name when spawning a host by passing multiple `-g` options to `ec2-run-instances`. Once you use VPCs though, you must refer to a security group by its ID when spawning a host. What I do is assign all of the security groups that belong to a host to a variable like so:

```
SEC_GROUPS='default foo'
```

Then I iterate through that list to pull out the IDs:

```
for g in ${SEC_GROUPS}; do
  SGID=$(aws ec2 describe-security-groups --filters
   ➥"Name=group-name,Values=$g" |
grep ^SECURITYGROUPS | cut -f3)
  SGIDS="$SGIDS $SGID"
done
```

Then I can pass `--security-group-ids $SGIDS` to my `aws ec2 run-instances` command.

Find the Subnet ID by Name

Another difference I ran across was in handling VPC subnets. With EC2 Classic, you didn't have to worry about subnets at all, and although you can label subnets with names in a VPC, when you spawn a host, it wants you to specify the ID. Again, I put the subnet label into a variable and use the AWS CLI to pull out the ID:

```
SUBNETID=$(aws ec2 describe-subnets --filters
➥"Name=tag:Name,Values=${SUBNET}" | grep ^SUBNETS | cut -f8)
```

Pull Data about a Specific Host

Back with EC2 API tools, it was easy to get the information about a specific host:

```
$ ec2-describe-instances | grep myhostname
```

Or the shorthand form:

```
$ ec2din | grep myhostname
```

Unfortunately, things aren't quite as easy with AWS CLI due to how the text output is formatted. Instead, you should use the `--filters` option to limit your search only to instances that have a Name tag that matches your hostname:

```
$ aws ec2 describe-instances --filters
➥"Name=tag:Name,Values=myhostname"
```

I ended up turning the previous command into a quick script called ec2dingrep:

```
#!/bin/bash

HOSTNAME=$1
aws ec2 describe-instances --filters
➥"Name=tag:Name,Values=${HOSTNAME}"
~
```

All in all, it really took me only part of a day to climb the learning curve of interacting with VPCs with AWS CLI and rewrite the script I use to spawn servers. Although so far it does take me a bit longer to string together the commands, much of that is muscle memory, and the much better speed of AWS CLI over EC2 API tools makes it worth it.■

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.

Resources

Amazon CLI Documentation: https://aws.amazon.com/cli

SHAWN POWERS

Wi-Fi, Part I: the Technology

If terms like 802.11ac, MIMO, channel bandwidth and frequency overlap are confusing, read on!

My family and I recently moved into a new house. Well, that's only partially true. We moved into a 63-year-old house, but it's new to us. One of the first things I did in the "new" house was to set up Wi-Fi. Thanks to the multiple floors and old plaster and lath walls, setting up a reliable wireless network proved to be quite challenging. I learned a lot along the way and figured it would be a perfect topic for an Open-Source Classroom series. So in this article, I discuss the technologies themselves. It's important to understand how Wi-Fi works so you know what sort of setup you want to install. Is 802.11ac worth the cost? Will it make a difference? Should you abandon the 2.4GHz frequency range altogether? Those questions will be easier to answer once you understand how the various standards work.

Frequency Ranges

There currently are three common frequency ranges used in wireless networking. The most common is the 2.4GHz range. The newer 802.11ac standard (and oddly, the mostly failed 802.11a standard) uses the 5GHz range. And, a handful of devices currently are utilizing the 900MHz range of the spectrum for networking. The latter isn't used very commonly, but that likely will change with the ratification of 802.11ah, which will standardize the spectrum for some really interesting use cases. For now, however, Wi-Fi access points are either 2.4GHz, 5GHz or both.

2.4GHz

The frequencies between 2.4 and 2.5GHz are broken up into 14 channels, all 5MHz apart. Channel 1 starts at 2.412GHz, and channel 14 is actually 12MHz above channel 13 (Figure 1). Every country handles the availability of those channels differently, but in the US, channels

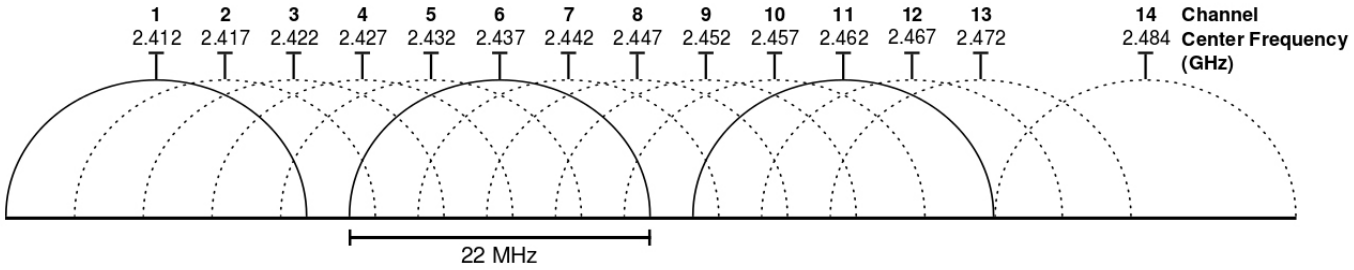


Figure 1. The 2.4GHz spectrum is so crowded and has so few usable channels (image from Wikipedia).

1–11 are available for public use. If wireless networks used only 5MHz of bandwidth to transmit and receive data, that would mean we'd have 11 channels to use, and we could have up to 11 access points in one place without worrying about interference.

Unfortunately, that's not how Wi-Fi works. Every wireless device using the 2.4GHz frequency range actually uses 22MHz of bandwidth when sending and receiving data. That increased bandwidth allows for more data to get through, and 22MHz was decided to be the standard. That means if a user transmits on channel 1, it bleeds 11MHz above and 11MHz below 2.412GHz. If you do the math, that means channel 1 actually bleeds over channel 2 and channel 3! Since each channel bleeds over its two adjacent channels, the only usable channels in the 2.4GHz range are channel 1, channel 6 and channel 11. (Again, see Figure 1 for a visual representation.)

In Japan, channels 12–14 are available for public use, and that makes channel 14 an option, but only barely. If you look at the graph, you can see channels 13 and 14 bump right up against each other.

In order to be complete, I also should mention that 802.11n allows for 40MHz of bandwidth per channel, even in the 2.4GHz frequency range. That increased bandwidth per channel means more throughput, but it comes at a terrible cost. Since the entire usable spectrum between channels 1 and 11 is 55MHz, there's really room only for a single device in the 2.4GHz spectrum to operate at 40MHz! It might seem like you could fit two channels in that space, but there's just not enough space in the legally usable spectrum to fit more than one channel that wide. So although it's technically possible to use 40MHz-wide channels with 2.4GHz using 802.11n, by default, it has to use 20MHz. Even

when changing the default setting to force 40MHz, devices must scale back to 20MHz if they sense interference. There are a few notable products that ignore that requirement (the original Apple Airport Extreme for instance), but it's just not wise to waste all the available spectrum on a single channel! If you want to use 40MHz-wide channels with 802.11n, the answer is to use 5GHz.

5GHz

The world of 5GHz networking is an interesting one. All the way back in 1999, 802.11a was released that functioned in the 5GHz range. The spectrum at that frequency is far less busy, and using a different signal modulation, it allowed for up to 54Mbps speeds. 802.11a used 20MHz of bandwidth, and since there is much more room in the 5GHz spectrum range, overlapping channels were far less of an issue. Unfortunately, 802.11a suffered from range and interference problems. The higher frequency doesn't penetrate walls as well, and as such, it wasn't as reliable in a building environment. Because it offered more potential bandwidth (54Mbps over 802.11b's 11Mbps), it was popular in corporate environments, but it was outmoded quickly by 802.11g. In 2003, 802.11g took the frequency modulation technology from 802.11a and applied it in the 2.4GHz spectrum. That allowed the best of both worlds, because it achieved up to 54Mbps while penetrating walls just as good as 802.11b did.

I mentioned that 802.11n can use 2.4GHz, but it also can use 5GHz for transmitting and receiving. The main problem is that not all devices that claim to be 802.11n are equipped with 5GHz radios. They're technically 802.11n-compatible, but only if that 802.11n is using the 2.4GHz spectrum. It's very frustrating. Most people end up implementing 802.11n on *both* 2.4GHz and 5GHz spectra. Thankfully, the 5GHz spectrum is both bigger and less busy, because not as many devices currently are utilizing it. Also, thanks to MIMO technology (more on that later), the penetration and interference isn't as much of an issue as it was with 802.11a.

Figure 2 shows the 5GHz band along with available channels. If you use the same 20MHz channel width used by the 2.4GHz channels, that means you have upwards of 25 non-overlapping channels from which to choose! The reality isn't quite that rosy, however, because more than half the available frequencies are called DFS channels, or Dynamic Frequency Selection channels. They work, but if your hardware finds

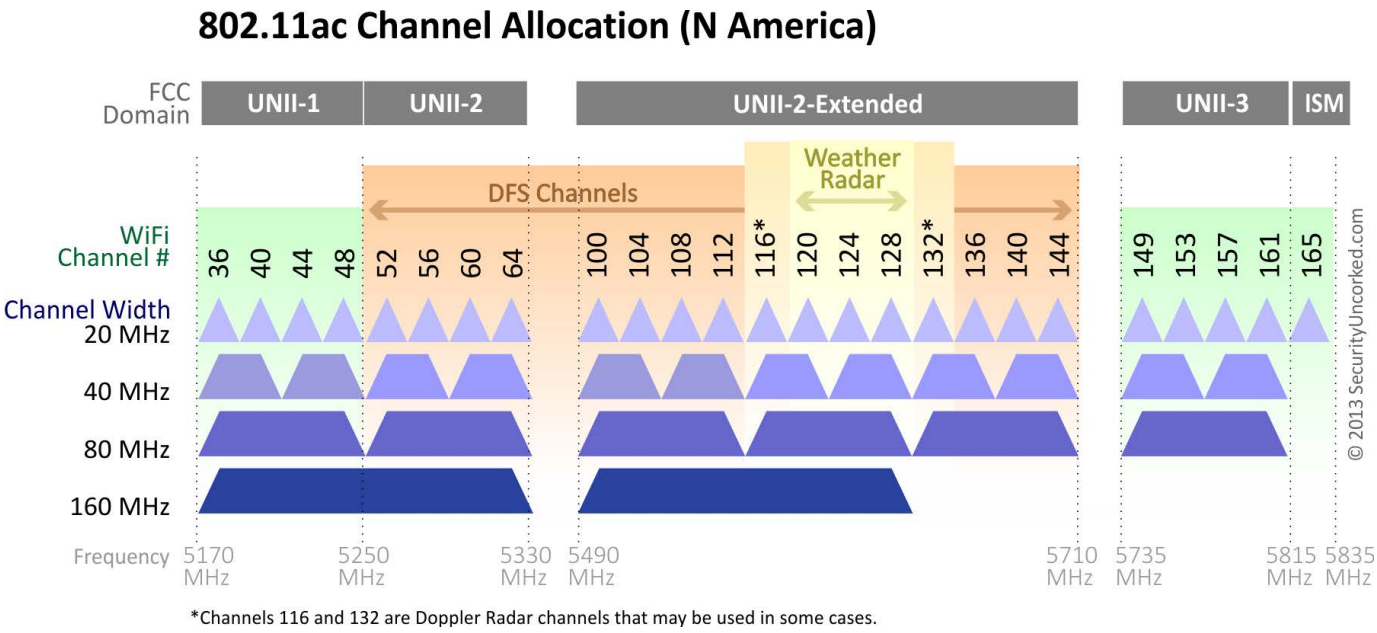


Figure 2. The 5GHz Channel Allocation (image from http://securityuncorked.com)

them being used by military or weather radios, it is forced to switch to another channel, even if you assigned the DFS channel manually. It's nice that we're able to use those frequencies, but we're truly second class citizens, because we always get moved if the channels are officially in use.

For most people, that means using the nine available non-overlapping channels that are outside the DFS range. You can see that channels 36, 40, 44, 48, 149, 153, 157, 161 and 165 are available to use without concern of getting bumped by weather or military. Things do become a little more complicated if you want to use more bandwidth per channel, however. Remember that 802.11n can use 40MHz per channel, which you see in Figure 2 will limit your available channels to four non-DFS, or 12 if you include DFS. Still, four completely open 40MHz channels is far, far better than the single 40MHz channel available in the 2.4GHz spectrum!

The new kid on the block is, of course, 802.11ac. This new standard uses *only* the 5GHz spectrum, and it allows a wide variety of channel widths. The old standard of 20MHz is allowed, but also widths of 40, 80 and even 160MHz are allowed for some serious throughput! As channel width increases, the number of non-overlapping channels decreases, of course, so Figure 2 shows how many options you have in

each scenario. I don't think there are any devices currently on the market that support 160MHz channels, but I could be wrong.

Why Is 802.11ac So Much Faster? (MIMO)

You might be wondering why 802.11ac is touted as so much faster than even 40MHz-wide 802.11n access points. The answer is similar to why 802.11n is so much faster than 802.11g.

You probably noticed that 802.11n and 802.11g both use the 2.4GHz spectrum (Table 1), and yet although 802.11g is limited to 54Mbps, the "n" standard can achieve speeds up to 300Mbps. That's because with 802.11n, Multiple Input Multiple Output (MIMO) technology allows multiple antennas to transmit and receive at the same time.

You can shop at the store and see 802.11n access points and routers boasting speeds of 150Mbps and others boasting 300Mbps. That's a multiplier based on the number of antennas the access points use at once. (Note: some vendors claim 802.11n access points can achieve 600Mbps, but they're just using fancy marketing to sound faster than they really are. They're simply adding 300Mbps transmit and 300Mbps receive to get 600; you can't really get 600Mbps transfer rates.) The 802.11n spec allows for four data streams at once, which is what makes it so much faster than 802.11g, which uses only a single data stream.

With 802.11ac, that number has been increased to a possible eight simultaneous data streams, meaning the potential throughput is more than 2Gbps. No hardware currently supports that many data streams, nor the channel width required to achieve it, but the standard supports a very fast future for 802.11ac.

5GHz Flexibility Solves Penetration Problem

The original problem faced by 802.11a's use of the 5GHz spectrum

Table 1. Details on the Various Standards

Standard	Frequency	Channel Bandwidth	MIMO Streams	Max Speed
802.11b	2.4GHz	22MHz	1	11Mbit
802.11a	5GHz	20MHz	1	54Mbit
802.11g	2.4GHz	20MHz	1	54Mbit
802.11n	2.4GHz/5GHz	20/40MHz	up to 4	up to 150Mbit (per stream)
802.11ac	5GHz	20/40/80/160MHz	up to 8	up to 866Mbit (per stream)

has been solved by several means. First, far more channels are available, so an increased number of access points can be near each other without concern of overlapping each other. That means a higher density of access points and a higher density of supportable clients.

The other "solution" to the 5GHz propagation problem is that with MIMO technology, a weaker signal is still very fast, because multiple antennas can work together even on a degraded connection. With 802.11a, there was only a single stream of data, so if the signal got weak, the throughput slowed way down. With MIMO, you get the advantage of multiple weaker signals, which add up to a decent speed even in an area of weak reception.

So What on Earth to Buy?!?!?

Although it would be nice simply to decide to go 5GHz only and buy the newest, fastest 802.11ac access point you can, it's not really a practical idea. Many devices still contain only 2.4GHz 802.11n radios, so you have to provide something on the overly saturated 2.4GHz spectrum. Thankfully, most access points provide dual-band support, and you can have both 5GHz and 2.4GHz options available.

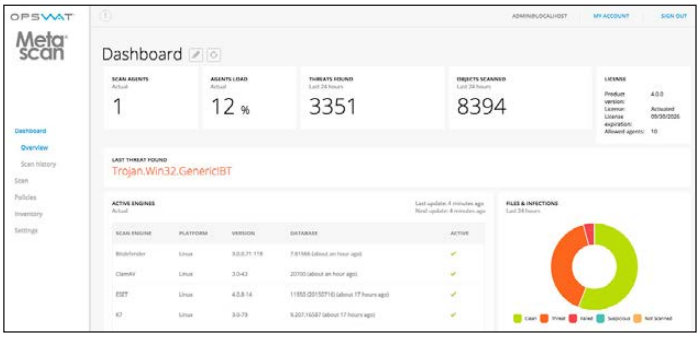
Sometimes it's better to have separate 2.4GHz and 5GHz access points, because you might need more 5GHz APs to cover the same area. Sometimes devices that support 5GHz networks will default to 2.4GHz if both are available with the same SSID. It can be very frustrating to figure out how best to support your devices!

In my next article, I'll tackle the actual planning, purchasing and installation of a wireless network in your home or business. Thankfully, having a firm understanding of the technology makes the next part a lot more fun. In the meantime, I recommend checking out your wireless devices to see what type of radios they have. Do you have anything that will take advantage of a 5GHz network? Anything that is 802.11ac-compatible? Do you want to future-proof your wireless network? Answer those questions to prepare for Part II of this series.■

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.

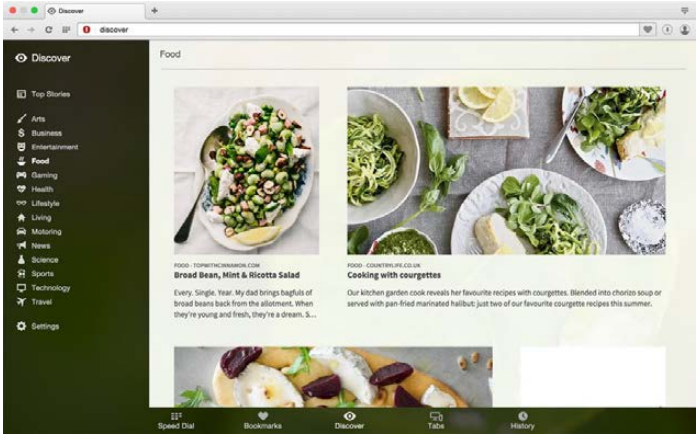
Send comments or feedback via http://www.linuxjournal.com/contact or to ljeditor@linuxjournal.com.

OPSWAT's Metascan



For years companies have been using Metascan from OPSWAT to catch threats reliably on files entering their organizations. Metascan, whose latest release also nabs threats to Linux systems, is a multi-scanning solution for ISVs, IT admins and malware researchers that detects and prevents known and unknown threats. The multi-scanning feature consists of multiple anti-malware engines from vendors like Bitdefender, ESET, Threat Track and others, a technology that increases detection rates for all types of malware without the hassle of licensing and maintaining the multiple engines. Myriad 64-bit Linux distributions are supported, including Debian, Red Hat Enterprise Linux, CentOS and Ubuntu. As part of its enhanced scalability, Metascan for Linux provides load balancing for high-volume scanning and can be used in high-availability deployments. Another important enhancement is the flexibility of the scan workflows, allowing different scan options by file type and source to offer increased security and threat prevention.

http://www.opswat.com



Opera Software ASA's Opera Browser

The most significant advancement in Opera 31, the updated, multiplatform browser from Opera Software ASA, is its startup time. Opera for computers now starts up to 70% faster compared to previous versions. This is especially important and helpful for devices with slower, traditional hard drives. Opera's developers also redesigned pages in the browser to make everything look sleek. In complement, Opera's Discover, a feature that serves users the most recent local content, has been given a makeover and now can be customized based on topic category and location. Finally, synchronization across devices also has been expanded and improved, allowing users to access their bookmarks, Speed Dial sites and tabs from anywhere.

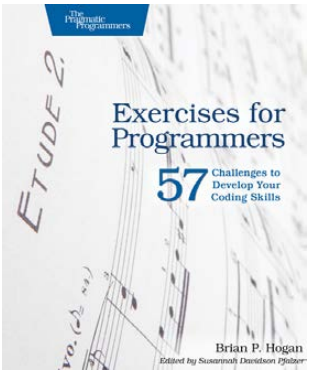
http://www.opera.com

SUSE Linux Enterprise Server for SAP Applications

Giving a boost to both the Linux platform in general and SUSE Linux Enterprise Server for SAP Applications specifically is new functionality to support IBM Power Systems running the SAP HANA platform. SUSE Linux customers now can take advantage of IBM's recently introduced series of Power Systems Solution Editions for SAP HANA, which can be tailored to meet enterprise-customer requirements for mission-critical, real-time analytics with high reliability, availability and serviceability. Customers seeking the business benefits of running in-memory databases like SAP HANA now can take advantage of the faster throughput, increased bandwidth and enterprise-grade reliability of IBM POWER8. Business users take in massive amounts of data and get vital, actionable information out in real time. The relevant version of SUSE Linux Enterprise Server for SAP Applications is version 11 Service Pack 4.

http://www.suse.com

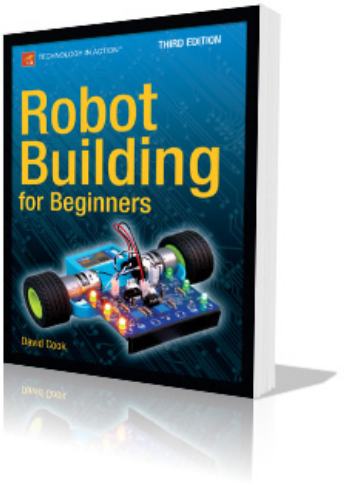
Brian P. Hogan's *Exercises for Programmers* (The Pragmatic Bookshelf)



One of the best ways to learn a programming language is to solve real problems with it. That's what Brian P. Hogan's new book *Exercises for Programmers: 57 Challenges to Develop Your Coding Skills* is all about. Instead of questions rooted in theory, this book from Pragmatic Bookshelf presents problems readers encounter in their day-in-day-out software development. These problems are designed for people learning their first programming language and also can provide a learning path for experienced developers to learn a new language quickly for their next important gig. The 57 exercises in Hogan's book start off simply and conclude with larger programs that synthesize everything. Each problem includes constraints and challenges to push readers further and force them to come up with the solutions. After learning one new language, readers later can work through the book again, using new approaches to solve familiar problems.

http://www.pragprog.com

David Cook's *Robot Building for Beginners*, Third Edition (Apress)



In the third edition of his popular book *Robot Building for Beginners*, author David Cook distills a lifetime of tinkering with robots so that his readers can get a head start on the fun, fulfilling process. *Robot Building* serves up essential, practical knowledge on getting started in amateur robotics and offers a mix of content from serious reference tables and descriptions to personal stories and humorous bits. The autonomous robot that readers build in this book is battery powered and the size of a lunch box. The book is broken up into small chapters, suitable for bedtime reading. The characteristics and purposes of each major component (resistor, transistor, wire and motor) are described and followed by a hands-on experiment to demonstrate. A section demonstrating how to "print" certain parts of the robot with a 3D printer also is included. Because robot building involves soldering, electricity, light machining and some cash, the writing is targeted toward individuals with those capabilities.

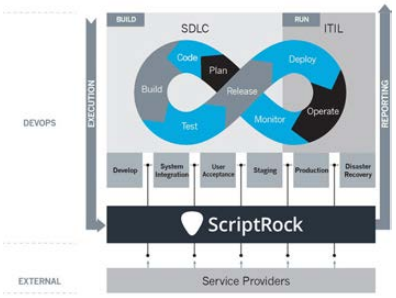
http://www.apress.com

AdaCore
The GNAT Pro Company

AdaCore's GNAT Pro for Wind River VxWorks

AdaCore is taking good care of its GNAT Pro development environment users on the Wind River VxWorks RTOS. The new GNAT Pro for Wind River VxWorks 7, which was developed in close collaboration with Wind River, offers full support for both Ada and the latest version 7 of VxWorks and provides more seamless integration with Wind River Workbench. AdaCore engineers ensured that GNAT Pro will support both single- and multi-core systems and other architectures and added enhancements, such as a completely re-engineered open-source debugger protocol, support for VxWorks 7 kernel modules and real-time processes, and the ability to handle both all-Ada and multi-language applications. AdaCore notes that AdaCore's GNAT Pro is well established among users of Wind River platforms, especially in the aerospace and defense markets.

http://www.adacore.com



ScriptRock

ScriptRock asserts that the update to its eponymous solution for unified integrity "gives everyone involved in the application development, release, and maintenance lifecycle the ability to answer complex questions faster than ever". ScriptRock provides validation of integrity at all phases of the continuous release cycle, automating the discovery, testing and monitoring of applications, systems and infrastructures for vulnerabilities and inconsistencies. In essence, devs, testers and administrators can maintain a grip on what's changing, what's different and what they have. Highlights of the updated ScriptRock's new features include powerful new search capabilities; more intuitive visualizations of scans and node changes; node group comparisons; dynamic node classification; actionable, streamlined scan navigations and interactive scan results that allow for active search filtering and manipulation of visualizations.
http://www.scriptrock.com

Zappix Visual IVR

Zappix's development of its Visual IVR customer service app is informed by research showing that 72% of customers prefer self-service options when resolving support issues. Zappix is a self-help, highly intuitive, Visual IVR app for Android, iPhone and Web that integrates voice and non-voice visual content with customer service channels—for example, phone/voice, Web, mobile on-line forms and multimedia resources. An updated version of the app adds new self-service features that are available while on hold for customer service and even after a session with a customer service representative. With these new capabilities, says Zappix, the app reduces support ticket volumes and the burden on a company's customer service team. This solution enables a smooth customer-service experience by answering customer-service issues as quickly as possible. The self-service technologies benefit both customers and companies, sayeth Zappix, with companies gaining a cost-effective alternative to live agent support and customers empowered to solve problems effectively on their own.
http://www.zappix.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.

Securi-Pi: Using the Raspberry Pi as a Secure Landing Point

Ever get stuck on a strange and untrusted network? Use your RPi to get around that!

BILL CHILDERS

Like many *LJ* readers these days, I've been leading a bit of a techno-nomadic lifestyle as of the past few years—jumping from network to network, access point to access point, as I bounce around the real world while maintaining my connection to the Internet and other networks I use on a daily basis. As of late, I've found that more and more networks are starting to block outbound ports like SMTP (port 25), SSH (port 22) and others. It becomes really frustrating when you drop into a local coffee house expecting to be able to fire up your SSH client and get a few things done, and you can't, because the network's blocking you.

However, I have yet to run across a network that blocks HTTPS outbound (port 443). After a bit of fiddling with a Raspberry Pi 2 I have at home, I was able to get a nice clean solution that lets me hit various services on the Raspberry Pi via port 443—allowing me to walk around blocked ports and hobbled networks so I can do the things I need to do. In a nutshell, I have set up this Raspberry Pi to act as an OpenVPN endpoint, SSH endpoint and Apache server—with all these services listening on port 443 so networks with restrictive policies aren't an issue.

NOTES

This solution will work on most networks, but firewalls that do deep packet inspection on outbound traffic still can block traffic that's tunneled using this method. However, I haven't been on a network that does that...yet. Also, while I use a lot of cryptography-based solutions here (OpenVPN, HTTPS, SSH), I haven't done a strict security audit of this setup. DNS may leak information, for example, and there may be other things I haven't thought of. I'm not recommending this as a way to hide all your traffic—I just use this so that I can connect to the Internet in an unfettered way when I'm out and about.

Getting Started

Let's start off with what you need to put this solution together. I'm using this on a Raspberry Pi 2 at home, running the latest Raspbian, but this should work just fine on a Raspberry Pi Model B, as well. It fits within the 512MB of RAM footprint quite easily, although performance may be a bit slower, because the Raspberry Pi Model B has a single-core CPU as opposed to the Pi 2's quad-core. My Raspberry Pi 2 is behind my home's router/firewall, so I get the added benefit of being able to access my machines at home. This also means that any traffic I send to the Internet appears to come from my home router's IP address, so this isn't a solution designed to protect anonymity. If you don't have a Raspberry Pi, or don't want this running out of your home, it's entirely possible to run this out of a small cloud server too. Just make sure that the server's running Debian or Ubuntu,

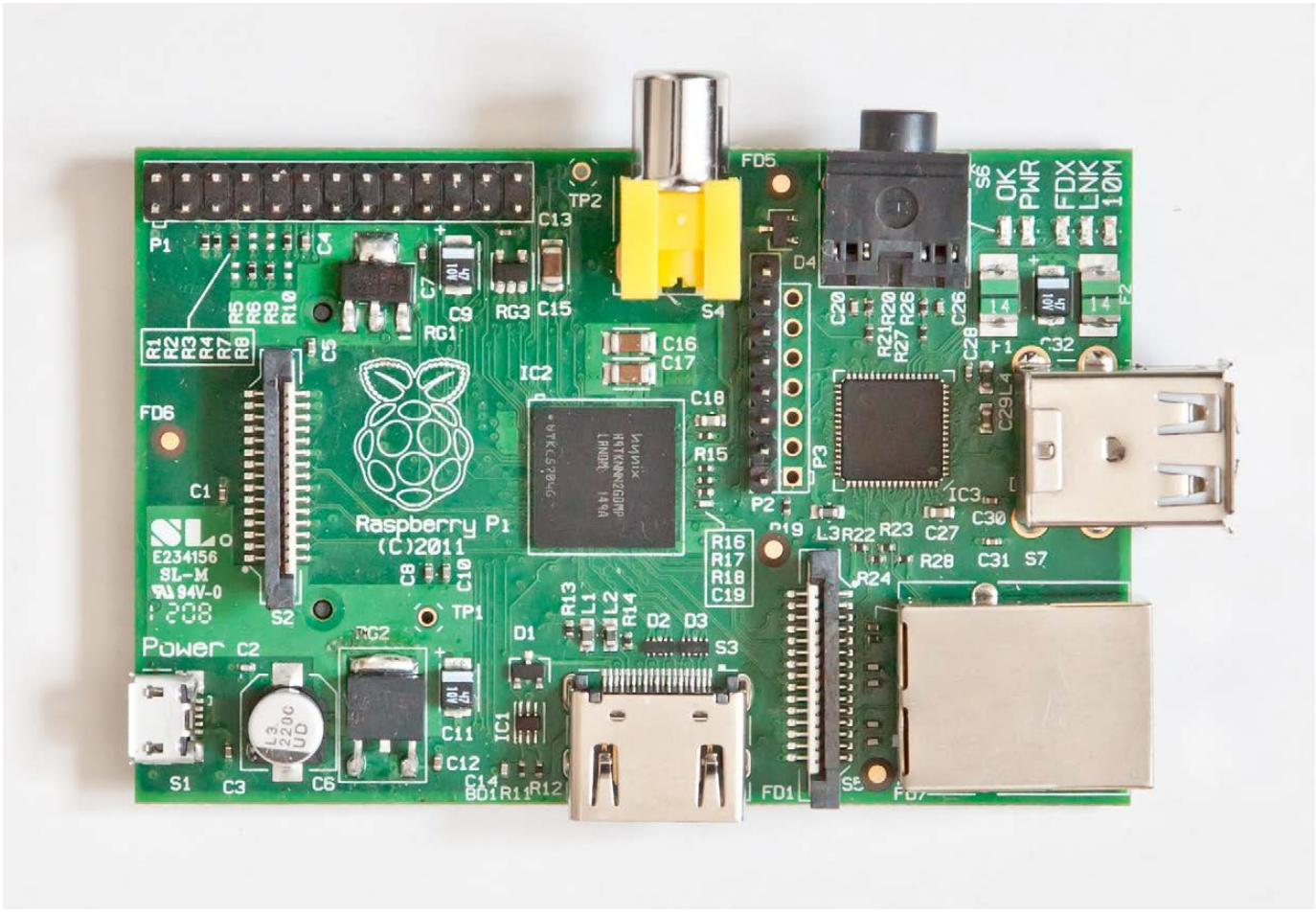


Figure 1. The Raspberry Pi, about to become an encrypted network endpoint.

as these instructions are targeted at Debian-based distributions.

Installing and Configuring BIND

Once you have your platform up and running—whether it's a Raspberry Pi or otherwise—next you're going to install BIND, the nameserver that powers a lot of the Internet. You're going to install BIND as a caching nameserver only, and not have it service incoming requests from the Internet. Installing BIND will give you a DNS server to point your OpenVPN clients at, once you get to the OpenVPN step. Installing BIND is easy; it's just a simple `apt-get` command to install it:

```
root@test:~# apt-get install bind9
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  bind9utils
Suggested packages:
  bind9-doc resolvconf ufw
The following NEW packages will be installed:
  bind9 bind9utils
0 upgraded, 2 newly installed, 0 to remove and
 ➥0 not upgraded.
Need to get 490 kB of archives.
After this operation, 1,128 kB of additional disk
 ➥space will be used.
Do you want to continue [Y/n]? y
```

There are a couple minor configuration changes that need to be made to one of the config files of BIND before it can operate as a caching nameserver. Both changes are in /etc/bind/named.conf.options. First, you're going to uncomment the "forwarders" section of this file, and you're going to add a nameserver on the Internet to which to forward requests. In this case, I'm going to add Google's DNS (8.8.8.8). The "forwarders" section of the file should look like this:

```
forwarders {
    8.8.8.8;
};
```

The second change you're going to make allows queries from your internal network and localhost. Simply add this line to the bottom of the configuration file, right before the `};` that ends the file:

```
allow-query { 192.168.1.0/24; 127.0.0.0/16; };
```

That line above allows this DNS server to be queried from the network it's on (in this case, my network behind my firewall) and localhost. Next, you just need to restart BIND:

```
root@test:~# /etc/init.d/bind9 restart
[....] Stopping domain name service...: bind9waiting
```

```
➥for pid 13209 to die
. ok
[ ok ] Starting domain name service...: bind9.
```

Now you can test `nslookup` to make sure your server works:

```
root@test:~# nslookup
> server localhost
Default server: localhost
Address: 127.0.0.1#53
> www.google.com
Server:         localhost
Address:   127.0.0.1#53

Non-authoritative answer:
Name: www.google.com
Address: 173.194.33.176
Name: www.google.com
Address: 173.194.33.177
Name: www.google.com
Address: 173.194.33.178
Name: www.google.com
Address: 173.194.33.179
Name: www.google.com
Address: 173.194.33.180
```

That's it! You've got a working nameserver on this machine. Next, let's move on to OpenVPN.

Installing and Configuring OpenVPN

OpenVPN is an open-source VPN solution that relies on SSL/TLS for its key exchange. It's also easy to install and get working under Linux. Configuration of OpenVPN can be a bit daunting, but you're not going to deviate from the default configuration by much. To start, you're going to run an `apt-get` command and install OpenVPN:

```
root@test:~# apt-get install openvpn
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  liblzo2-2 libpkcs11-helper1
Suggested packages:
  resolvconf
The following NEW packages will be installed:
  liblzo2-2 libpkcs11-helper1 openvpn
0 upgraded, 3 newly installed, 0 to remove and
➥0 not upgraded.
Need to get 621 kB of archives.
After this operation, 1,489 kB of additional disk
➥space will be used.
Do you want to continue [Y/n]? y
```

Now that OpenVPN is installed, you're going to configure it. OpenVPN is SSL-based, and it relies on both server and client certificates to work. To generate these certificates, you need to configure a Certificate Authority (CA) on the machine. Luckily, OpenVPN ships with some wrapper scripts known as "easy-rsa" that help to bootstrap this process. You'll start by making a directory on

the filesystem for the easy-rsa scripts to reside in and by copying the scripts from the template directory there:

```
root@test:~# mkdir /etc/openvpn/easy-rsa
root@test:~# cp -rpv
 ➥/usr/share/doc/openvpn/examples/easy-rsa/2.0/*
 ➥/etc/openvpn/easy-rsa/
```

Next, copy the vars file to a backup copy:

```
root@test:/etc/openvpn/easy-rsa# cp vars vars.bak
```

Now, edit vars so it's got information pertinent to your installation. I'm going specify only the lines that need to be edited, with sample data, below:

```
KEY_SIZE=4096
KEY_COUNTRY="US"
KEY_PROVINCE="CA"
KEY_CITY="Silicon Valley"
KEY_ORG="Linux Journal"
KEY_EMAIL="bill.childers@linuxjournal.com"
```

The next step is to source the vars file, so that the environment variables in the file are in your current environment:

```
root@test:/etc/openvpn/easy-rsa# source ./vars
NOTE: If you run ./clean-all, I will be doing a
 ➥rm -rf on /etc/openvpn/easy-rsa/keys
```

Building the Certificate Authority

You're now going to run `clean-all` to ensure a clean working environment, and then you're going to build the CA. Note that I'm changing `changeme` prompts to something that's appropriate for this installation:

```
root@test:/etc/openvpn/easy-rsa# ./clean-all
root@test:/etc/openvpn/easy-rsa# ./build-ca
Generating a 4096 bit RSA private key
.....................................................++
.....................................................++
writing new private key to 'ca.key'
-----
You are about to be asked to enter information that
will be incorporated into your certificate request.
What you are about to enter is what is called a
Distinguished Name or a DN.
There are quite a few fields but you can leave some
blank. For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [US]:
State or Province Name (full name) [CA]:
Locality Name (eg, city) [Silicon Valley]:
Organization Name (eg, company) [Linux Journal]:
Organizational Unit Name (eg, section)
 ➥[changeme]:SecTeam
Common Name (eg, your name or your server's hostname)
 ➥[changeme]:test.linuxjournal.com
Name [changeme]:test.linuxjournal.com
Email Address [bill.childers@linuxjournal.com]:
```

Building the Server Certificate

Once the CA is created, you need to build the OpenVPN server certificate:

```
root@test:/etc/openvpn/easy-rsa#
 ➥./build-key-server test.linuxjournal.com
Generating a 4096 bit RSA private key
.....................................................++
writing new private key to 'test.linuxjournal.com.key'
-----
You are about to be asked to enter information that
will be incorporated into your certificate request.
What you are about to enter is what is called a
Distinguished Name or a DN.
There are quite a few fields but you can leave some
blank. For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [US]:
State or Province Name (full name) [CA]:
Locality Name (eg, city) [Silicon Valley]:
Organization Name (eg, company) [Linux Journal]:
Organizational Unit Name (eg, section)
 ➥[changeme]:SecTeam
Common Name (eg, your name or your server's hostname)
 ➥[test.linuxjournal.com]:
Name [changeme]:test.linuxjournal.com
Email Address [bill.childers@linuxjournal.com]:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
Using configuration from
 ➥/etc/openvpn/easy-rsa/openssl-1.0.0.cnf
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
countryName           :PRINTABLE:'US'
stateOrProvinceName   :PRINTABLE:'CA'
localityName          :PRINTABLE:'Silicon Valley'
organizationName      :PRINTABLE:'Linux Journal'
organizationalUnitName:PRINTABLE:'SecTeam'
commonName            :PRINTABLE:'test.linuxjournal.com'
name                  :PRINTABLE:'test.linuxjournal.com'
emailAddress
 ➥:IA5STRING:'bill.childers@linuxjournal.com'
Certificate is to be certified until Sep  1
 ➥06:23:59 2025 GMT (3650 days)
Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
```

The next step may take a while—building the Diffie-Hellman key for the OpenVPN server. This takes several minutes on a conventional desktop-grade CPU, but on the ARM processor of the Raspberry Pi, it can take much, much longer. Have patience, as long as the dots in the terminal are proceeding, the system is building its Diffie-Hellman key (note that many dots are snipped in these examples):

```
root@test:/etc/openvpn/easy-rsa# ./build-dh
Generating DH parameters, 4096 bit long safe prime,
 ➥generator 2
```

```
This is going to take a long time
.....................................................+
<snipped out many more dots>
```

Building the Client Certificate

Now you're going to generate a client key for your client to use when logging in to the OpenVPN server. OpenVPN is typically configured for certificate-based auth, where the client presents a certificate that was issued by an approved Certificate Authority:

```
root@test:/etc/openvpn/easy-rsa# ./build-key
 ➥bills-computer
Generating a 4096 bit RSA private key
.....................................................++
.....................................................++
writing new private key to 'bills-computer.key'
-----
You are about to be asked to enter information that
will be incorporated into your certificate request.
What you are about to enter is what is called a
Distinguished Name or a DN. There are quite a few
fields but you can leave some blank.
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [US]:
State or Province Name (full name) [CA]:
Locality Name (eg, city) [Silicon Valley]:
Organization Name (eg, company) [Linux Journal]:
Organizational Unit Name (eg, section)
 ➥[changeme]:SecTeam
Common Name (eg, your name or your server's hostname)
 ➥[bills-computer]:
Name [changeme]:bills-computer
Email Address [bill.childers@linuxjournal.com]:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
Using configuration from
 ➥/etc/openvpn/easy-rsa/openssl-1.0.0.cnf
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
countryName           :PRINTABLE:'US'
stateOrProvinceName   :PRINTABLE:'CA'
localityName          :PRINTABLE:'Silicon Valley'
organizationName      :PRINTABLE:'Linux Journal'
organizationalUnitName:PRINTABLE:'SecTeam'
commonName            :PRINTABLE:'bills-computer'
name                  :PRINTABLE:'bills-computer'
emailAddress
 ➥:IA5STRING:'bill.childers@linuxjournal.com'
Certificate is to be certified until
 ➥Sep  1 07:35:07 2025 GMT (3650 days)
Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified,
 ➥commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
root@test:/etc/openvpn/easy-rsa#
```

Now you're going to generate an HMAC code as a shared key

to increase the security of the system further:

```
root@test:~# openvpn --genkey --secret
 ➥/etc/openvpn/easy-rsa/keys/ta.key
```

Configuration of the Server

Finally, you're going to get to the meat of configuring the OpenVPN server. You're going to create a new file, /etc/openvpn/server.conf, and you're going to stick to a default configuration for the most part. The main change you're going to do is to set up OpenVPN to use TCP rather than UDP. This is needed for the next major step to work—without OpenVPN using TCP for its network communication, you can't get things working on port 443. So, create a new file called /etc/openvpn/server.conf, and put the following configuration in it:

```
port 1194
proto tcp
dev tun
ca easy-rsa/keys/ca.crt
cert easy-rsa/keys/test.linuxjournal.com.crt ## or whatever
 ➥your hostname was
key easy-rsa/keys/test.linuxjournal.com.key  ## Hostname key
 ➥- This file should be kept secret
management localhost 7505
dh easy-rsa/keys/dh4096.pem
tls-auth /etc/openvpn/certs/ta.key 0
server 10.8.0.0 255.255.255.0 # The server will use this
 ➥subnet for clients connecting to it
ifconfig-pool-persist ipp.txt
push "redirect-gateway def1 bypass-dhcp" # Forces clients
 ➥to redirect all traffic through the VPN
push "dhcp-option DNS 192.168.1.1" # Tells the client to
 ➥use the DNS server at 192.168.1.1 for DNS -
 ➥replace with the IP address of the OpenVPN
 ➥machine and clients will use the BIND
 ➥server setup earlier
keepalive 30 240
comp-lzo # Enable compression
persist-key
persist-tun
status openvpn-status.log
verb 3
```

And last, you're going to enable IP forwarding on the server, configure OpenVPN to start on boot and start the OpenVPN service:

```
root@test:/etc/openvpn/easy-rsa/keys# echo
 ➥"net.ipv4.ip_forward = 1" >> /etc/sysctl.conf
root@test:/etc/openvpn/easy-rsa/keys# sysctl -p
 ➥/etc/sysctl.conf
net.core.wmem_max = 12582912
net.core.rmem_max = 12582912
net.ipv4.tcp_rmem = 10240 87380 12582912
net.ipv4.tcp_wmem = 10240 87380 12582912
net.core.wmem_max = 12582912
net.core.rmem_max = 12582912
net.ipv4.tcp_rmem = 10240 87380 12582912
net.ipv4.tcp_wmem = 10240 87380 12582912
net.core.wmem_max = 12582912
```

```
net.core.rmem_max = 12582912
net.ipv4.tcp_rmem = 10240 87380 12582912
net.ipv4.tcp_wmem = 10240 87380 12582912
net.ipv4.ip_forward = 0
net.ipv4.ip_forward = 1

root@test:/etc/openvpn/easy-rsa/keys# update-rc.d
 ➥openvpn defaults
update-rc.d: using dependency based boot sequencing

root@test:/etc/openvpn/easy-rsa/keys#
 ➥/etc/init.d/openvpn start
[ ok ] Starting virtual private network daemon:.
```

Setting Up OpenVPN Clients

Your client installation depends on the host OS of your client, but you'll need to copy your client certs and keys created above to your client, and you'll need to import those certificates and create a configuration for that client. Each client and client OS does it slightly differently and documenting each one is beyond the scope of this article, so you'll need to refer to the documentation for that client to get it running. Refer to the Resources section for OpenVPN clients for each major OS.

Installing SSLH—the "Magic" Protocol Multiplexer

The really interesting piece of this solution is SSLH. SSLH is a protocol multiplexer—it listens on port 443 for traffic, and then it can analyze whether the incoming packet is an SSH packet, HTTPS or OpenVPN, and it can forward that packet onto the proper service. This is what enables this solution to bypass most port blocks—you use the HTTPS port for all of this traffic, since HTTPS is rarely blocked.

To start, `apt-get install` SSLH:

```
root@test:/etc/openvpn/easy-rsa/keys# apt-get
 ➥install sslh
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  apache2 apache2-mpm-worker apache2-utils
   ➥apache2.2-bin apache2.2-common
  libapr1 libaprutil1 libaprutil1-dbd-sqlite3
   ➥libaprutil1-ldap libconfig9
Suggested packages:
  apache2-doc apache2-suexec apache2-suexec-custom
   ➥openbsd-inetd inet-superserver
The following NEW packages will be installed:
  apache2 apache2-mpm-worker apache2-utils
   ➥apache2.2-bin apache2.2-common
  libapr1 libaprutil1 libaprutil1-dbd-sqlite3
   ➥libaprutil1-ldap libconfig9 sslh
0 upgraded, 11 newly installed, 0 to remove
 ➥and 0 not upgraded.
Need to get 1,568 kB of archives.
After this operation, 5,822 kB of additional
 ➥disk space will be used.
Do you want to continue [Y/n]? y
```

After SSLH is installed, the package installer will ask you if you want to run it in inetd or standalone mode. Select standalone mode, because you want SSLH to run as its own process. If you don't have Apache installed, the Debian/Raspbian package of SSLH will pull it in automatically, although it's not strictly required. If you already have Apache running and configured, you'll want to make sure it *only* listens on localhost's interface and not all interfaces (otherwise, SSLH can't start because it can't bind to port 443). After installation, you'll receive an error that looks like this:

```
[....] Starting ssl/ssh multiplexer: sslhsslh disabled,
 ➥please adjust the configuration to your needs
[FAIL] and then set RUN to 'yes' in /etc/default/sslh
 ➥to enable it. ... failed!
failed!
```

This isn't an error, exactly—it's just SSLH telling you that it's not configured and can't start. Configuring SSLH is pretty simple. Its configuration is stored in /etc/default/sslh, and you just need to configure the `RUN` and `DAEMON_OPTS` variables. My SSLH configuration looks like this:

```
# Default options for sslh initscript
# sourced by /etc/init.d/sslh

# Disabled by default, to force yourself
# to read the configuration:
# - /usr/share/doc/sslh/README.Debian (quick start)
# - /usr/share/doc/sslh/README, at "Configuration" section
# - sslh(8) via "man sslh" for more configuration details.
# Once configuration ready, you *must* set RUN to yes here
# and try to start sslh (standalone mode only)

RUN=yes

# binary to use: forked (sslh) or single-thread
 ➥(sslh-select) version
DAEMON=/usr/sbin/sslh

DAEMON_OPTS="--user sslh --listen 0.0.0.0:443 --ssh
 ➥127.0.0.1:22 --ssl 127.0.0.1:443 --openvpn
 ➥127.0.0.1:1194 --pidfile /var/run/sslh/sslh.pid"
```

Save the file and start SSLH:

```
root@test:/etc/openvpn/easy-rsa/keys#
 ➥/etc/init.d/sslh start
[ ok ] Starting ssl/ssh multiplexer: sslh.
```

Now, you should be able to `ssh` to port 443 on your Raspberry Pi, and have it forward via SSLH:

```
$ ssh -p 443 root@test.linuxjournal.com
root@test:~#
```

SSLH is now listening on port 443 and can direct traffic to SSH, Apache or OpenVPN based on the type of

packet that hits it. You should be ready to go!

Conclusion

Now you can fire up OpenVPN and set your OpenVPN client configuration to port 443, and SSLH will route it to the OpenVPN server on port 1194. But because you're talking to your server on port 443, your VPN traffic won't get blocked. Now you can land at a strange coffee shop, in a strange town, and know that your Internet will just work when you fire up your OpenVPN and point it at your Raspberry Pi. You'll also gain some encryption on your link, which will improve the privacy of your connection. Enjoy surfing the Net via your new landing point! ■

Bill Childers is an IT Manager in Silicon Valley, where he lives with his wife and two children. He enjoys Linux far too much, and probably should get more sun from time to time.

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

Resources

Installing and Configuring OpenVPN: https://wiki.debian.org/OpenVPN and http://cryptotap.com/articles/openvpn

OpenVPN client downloads: https://openvpn.net/index.php/open-source/downloads.html

OpenVPN Client for iOS: https://itunes.apple.com/us/app/openvpn-connect/id590379981?mt=8

OpenVPN Client for Android: https://play.google.com/store/apps/details?id=net.openvpn.openvpn&hl=en

Tunnelblick for Mac OS X (OpenVPN client): https://tunnelblick.net

SSLH—Protocol Multiplexer: http://www.rutschle.net/tech/sslh.shtml and https://github.com/yrutschle/sslh

Build a Large-Screen Command Center with the RPi 2

Two Raspberry Pi 2 model Bs show their stuff by powering two command center 4K monitors.

JOHN S. TONELLO

When the folks who make the Raspberry Pi made good on their plan to release a multi-core version of the tiny computer with 1GB of RAM earlier this year, I saw it as the perfect opportunity to put the single-board Linux box to work—real work—in our company's network operations center.

NYSERNet, Inc., is an optical networking company that provides high-speed connections for New York state's leading universities and research centers. Part of our job is to keep an eye on a fiber network that stretches from Buffalo to New York City. If something does down, we need to know quickly.

In the past, we had a walk-up command center that featured ten wall-mounted 19-inch monitors powered by two large-form Windows PC towers loaded with heavy-duty video cards and enough VRAM to make everything work. The screens showed network maps, data-center views and weather, among other things.

But during a recent office remodel, we decided all the cabling, clunky-looking PCs and video-card sharing needed to go. We wanted the new space—with two new 50-inch Ultra HD monitors instead of the army of 19-inchers—to be clean and uncluttered, and to offer staff and visitors a nice sense of "Aw, cool!"

Enter the Raspberry Pi 2 Model B.

With its powerful new four-core processor and double the RAM of its immediate predecessor, the RPi seemed to be the perfect computer not only to drive our large new 4K monitors, but also to run several important applications used by the NOC team, including a Java-based network map, Iceweasel (a Firefox Web browser derivative) and InterMapper, a proprietary network monitoring tool. Raspbian, a Debian derivative that fits well with our Ubuntu shop, would be an ideal choice for the OS.

Before we could dive in and turn over a very public—and necessary—system to a pair of $35 RPis, I had to make sure they had enough video power to drive the large televisions. After all, we needed the 4K resolution in order to get the pixel depth necessary for sharp screen views and enough graphical real estate to accommodate the applications we'd be running. The old ten-screen setup featured a different application window on each monitor; the two-screen setup needed to show just as many windows.

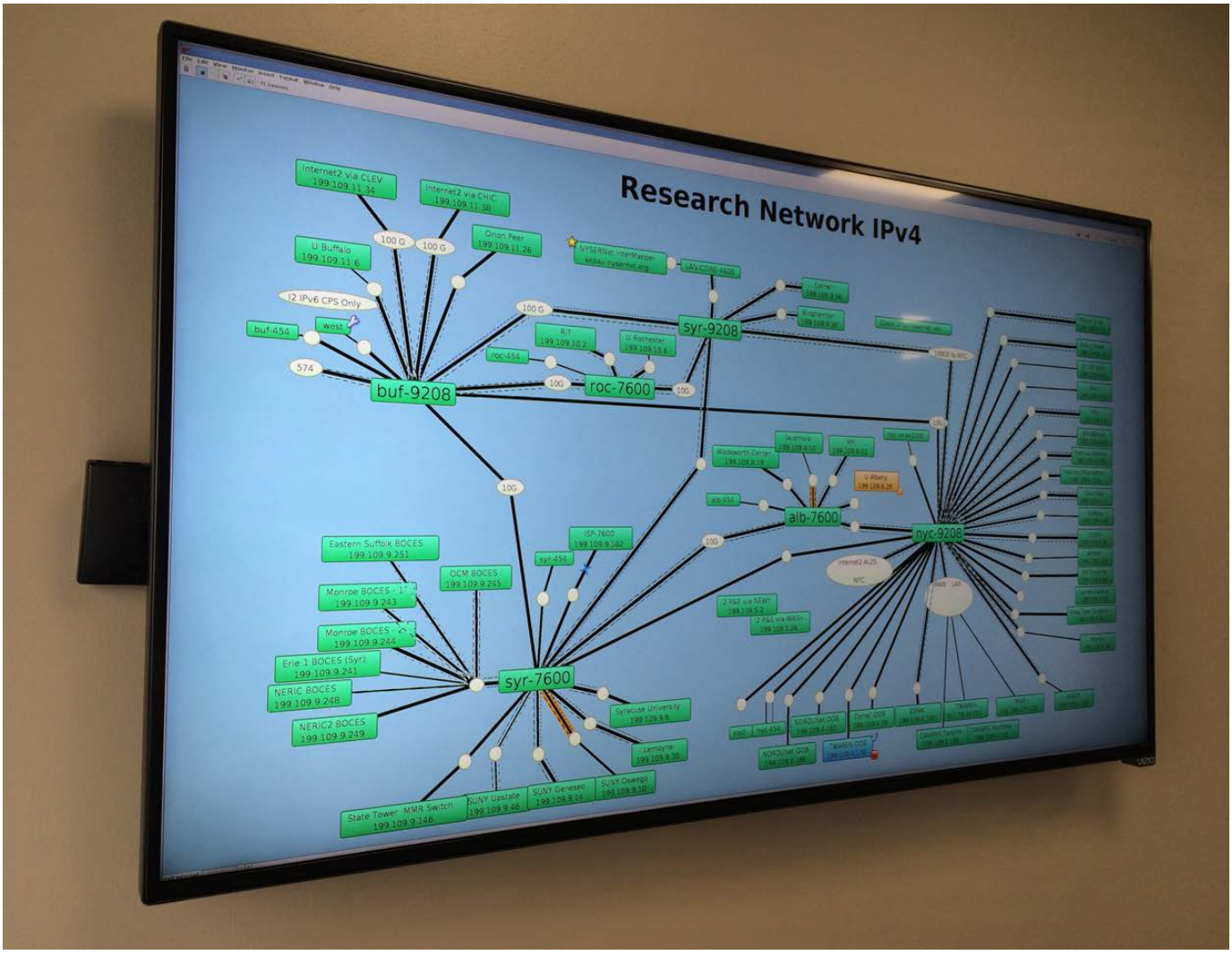


Figure 1. A 50-inch Vizio TV displays a NYSERNet, Inc., network map in high resolution, powered by a Raspberry Pi 2 Model B, the small black box to the left. The RPi is shown pulled out from its normal hiding space behind the screen (photo by John S. Tonello).

I ran some preliminary tests on an HP Mini running Windows 8.1 and on the RPi 2 Model B. The Mini could muster only 1080p, but I found the RPi could indeed provide the resolution I wanted with its built-in full-size HDMI port and on-board Broadcom graphics. I also found I could do it without having to set the poor RPi on fire by overclocking it to the max.

Choosing the RPis

For this project, I needed two RPis, one for each new large-screen TV. You easily can set up more. I ordered

This particular RPi is ideal because it's powerful enough to run high-resolution video and our network-monitoring applications simultaneously.

two of the latest Raspberry Pi 2 Model B kits, which each included a computer, an 8GB SD card, a vented plastic case and a USB power supply. This particular RPi is ideal because it's powerful enough to run high-resolution video and our network-monitoring applications simultaneously. The previous RPi B+ with 512MB of RAM didn't provide quite enough horsepower for what I needed. If you don't need the Ultra HD resolution, an older RPi may work fine.

The rest of this article assumes you know how to set up a Raspberry Pi. If not, I've listed a good article in the Resources section that explains how to do it. I used Raspbian as my base OS.

Once your RPi is up and running, it's a good idea to connect it to your network. If something goes wrong with the video settings during this setup, you'll want another way to edit the configuration files. Giving the RPi an address on your network and setting up SSH will do just that.

If you're able to find a kit with a pre-installed Raspbian (or NOOBS) SD card, that'll work fine. Other flavors, such as Arch Linux ARM and Pidora, also may be good options, since this solution relies more on the RPi's base configuration and less on the OS that sits on top of it.

All told, the kits cost about $65 each. I purchased a couple 12-inch HDMI cables too—one for each RPi-TV pair.

Choosing the Screens

It was important to get high-resolution screens for this project, and after a bit of shopping, I settled on the Vizio 50-inch 4K Ultra HD LED Smart TV (P502UI-B1E). It features 3840x2160 resolution, a 120Hz refresh rate and the 16:9 aspect ratio I wanted. It didn't hurt that this particular television came in at less than $800 delivered.

Configuring the RPi

Using the raspi-config tool (built in to Raspbian), you can fine-tune how the computer handles memory, video display and a host of other RPi parameters. For my purposes, I used it

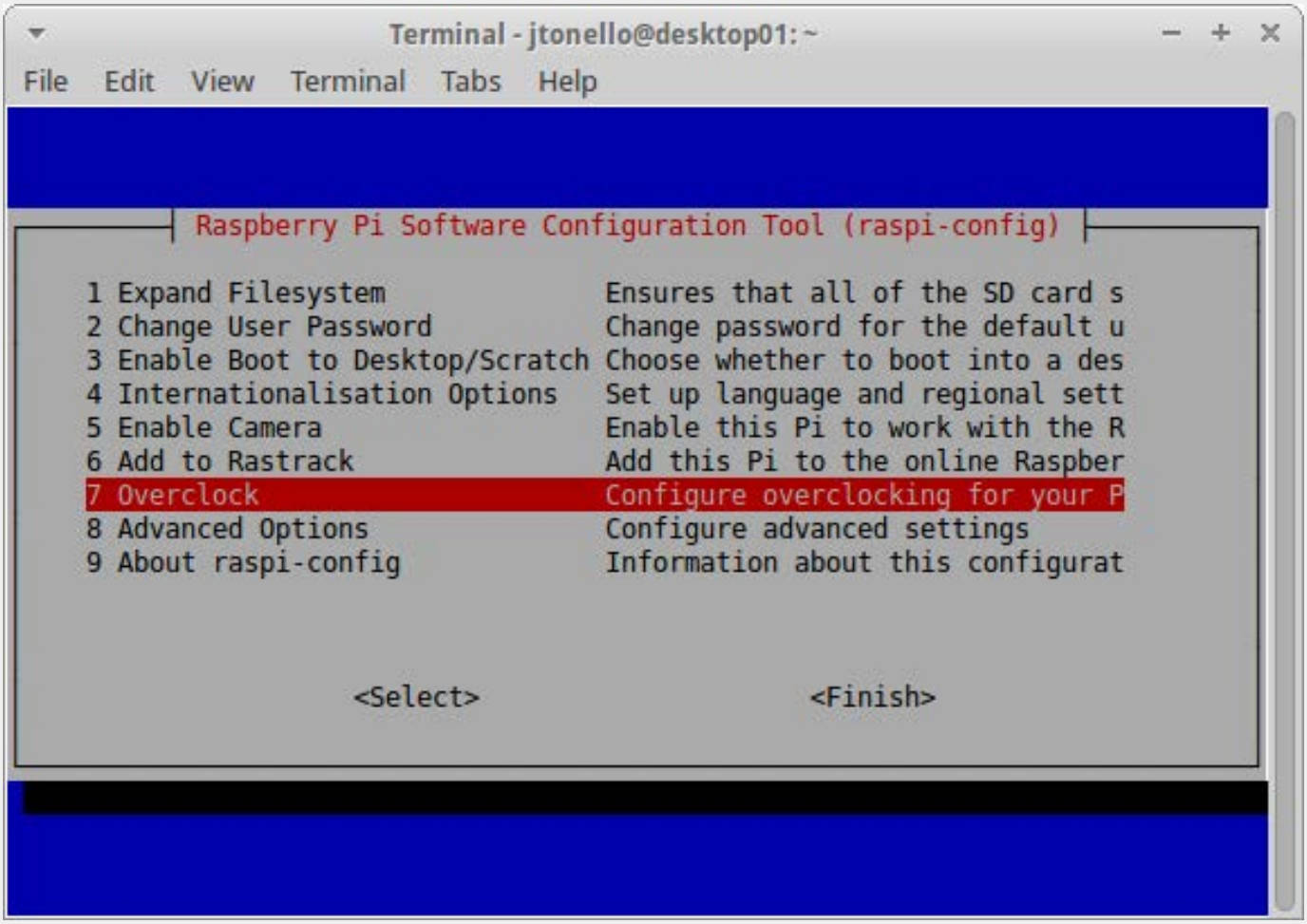


Figure 2. Overclock

to work with three things:

- Overclocking
- Overscan
- Memory Split

By the way, these settings also can be modified by editing /boot/config.txt, so if you don't have raspi-config on your distribution, you can set the parameters using the configuration file. The /boot/config.txt on an RPi is sort of equivalent to BIOS on other computers, and there's good Raspbian documentation available for it. For more information, I've included a link about it in the Resources section.

To start with the modifications, launch raspi-config:

```
$ sudo raspi-config
```

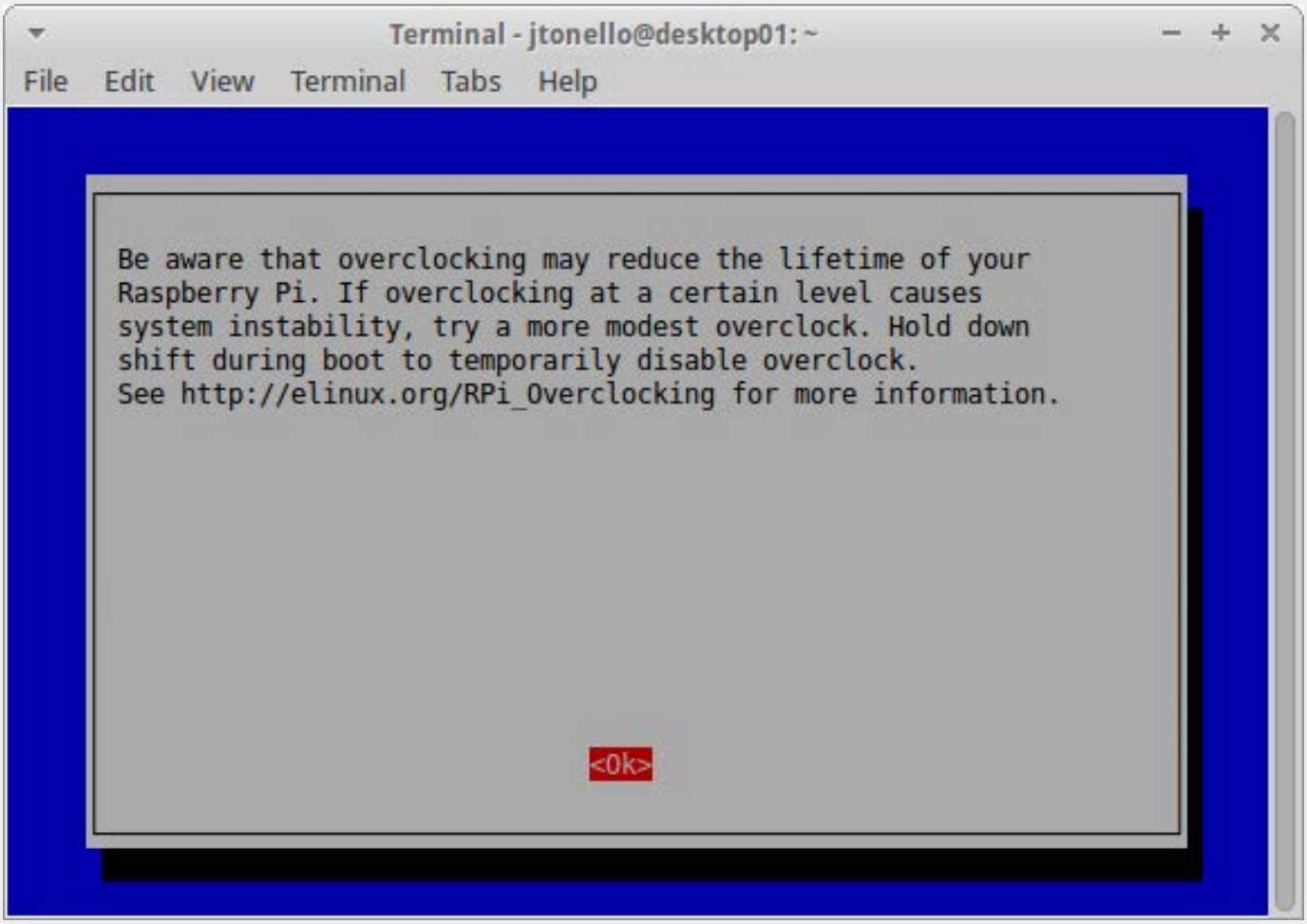


Figure 3. Overclocking Warning

Overclocking is option 7 in the first menu; Overscan and Memory Split are options A1 and A3 under the Advanced Options menu. Start with Overclocking (Figure 2).

After reading the warning, press Enter to acknowledge it (Figure 3). It's important to know that overclocking can make weird things happen to your RPi, so be careful.

You'll be offered six overclocking options. Each offers an increasing level of speed—and heat. Although the RPi 2 Model B is rated at 900MHz, I went ahead and chose the Medium overclock preset, which offers 900MHz ARM, 250MHz core, 450MHz SDRAM and 2 overvolt (Figure 4).

After experimenting with the basic settings (no overclocking), I found that this medium overclocking made the applications run a little faster, meaning the human experience of using the RPi was better.

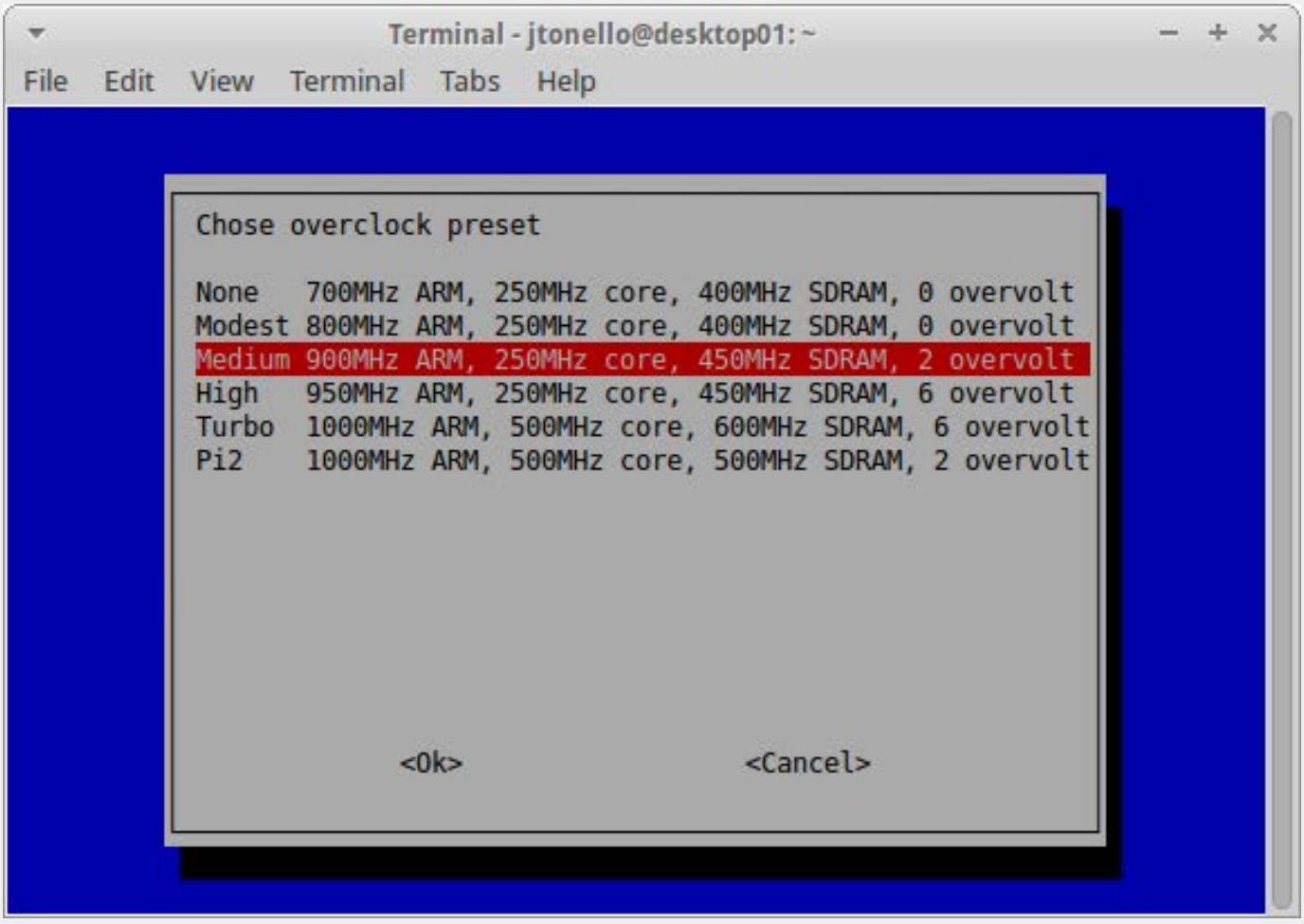


Figure 4. Choosing Overclock Preset

If your applications are less resource-heavy, you probably can skip overclocking altogether. However, even multiple Iceweasel windows or tabs can hog system resources and may lead you to want a little more oomph. Remember, the RPi is being asked to do quite a bit of graphical processing that will use memory and cycles.

Select Medium, tab to Ok and press Enter, and you'll be back at the main raspi-config menu.

Now it's time to make two Advanced Options configuration changes: overscanning and memory. First, let's look at overscanning.

When you first connect your RPi to a monitor and boot it, you may see black bars on the screen. These are unused pixels, and overscan is disabled by default to give you a sort of graphical safe mode. If your display is newer (like my two 4K TVs), you safely can disable

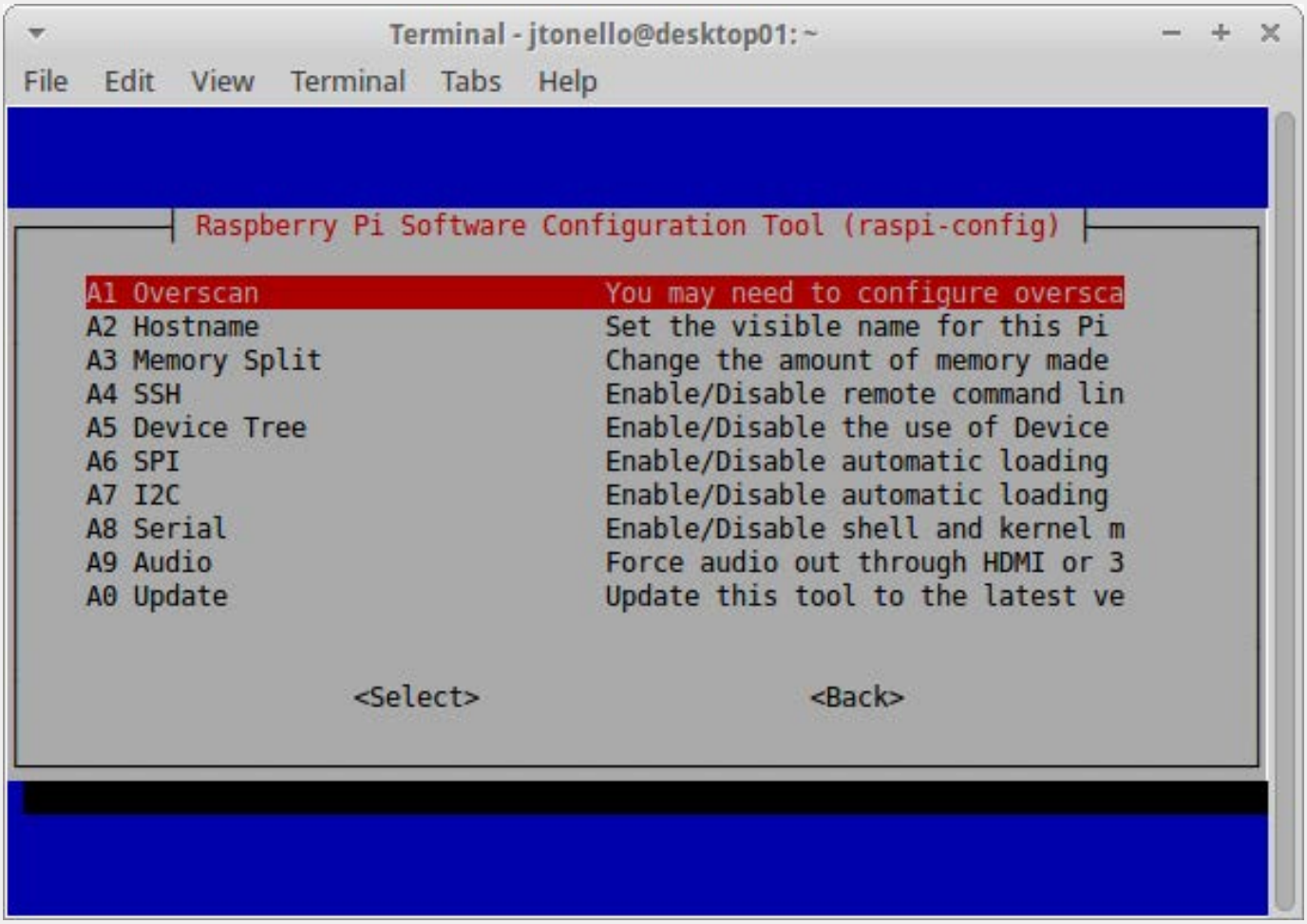


Figure 5. Overscan

overscan and allow the RPi to use all the screen real estate.

Select Advanced Options in the main raspi-config menu, and choose Overscan (A1) from the list (Figure 5). You have two options: disable or enable. Choose Disable, and tab to Ok to return to the main menu.

It's often hard to know if overscan will mess things up, so this is a good point to save what you've done and reboot the RPi. Select Finish from the main menu. That'll drop you to your command prompt:

```
$ sudo reboot
```

If your screen looks okay on reboot (no black bars, no odd pixelation and no weird flashing), you should be good to go. If it's screwed up, or screwed up enough that you can't read anything, you'll need to `ssh` in to the RPi from a separate computer

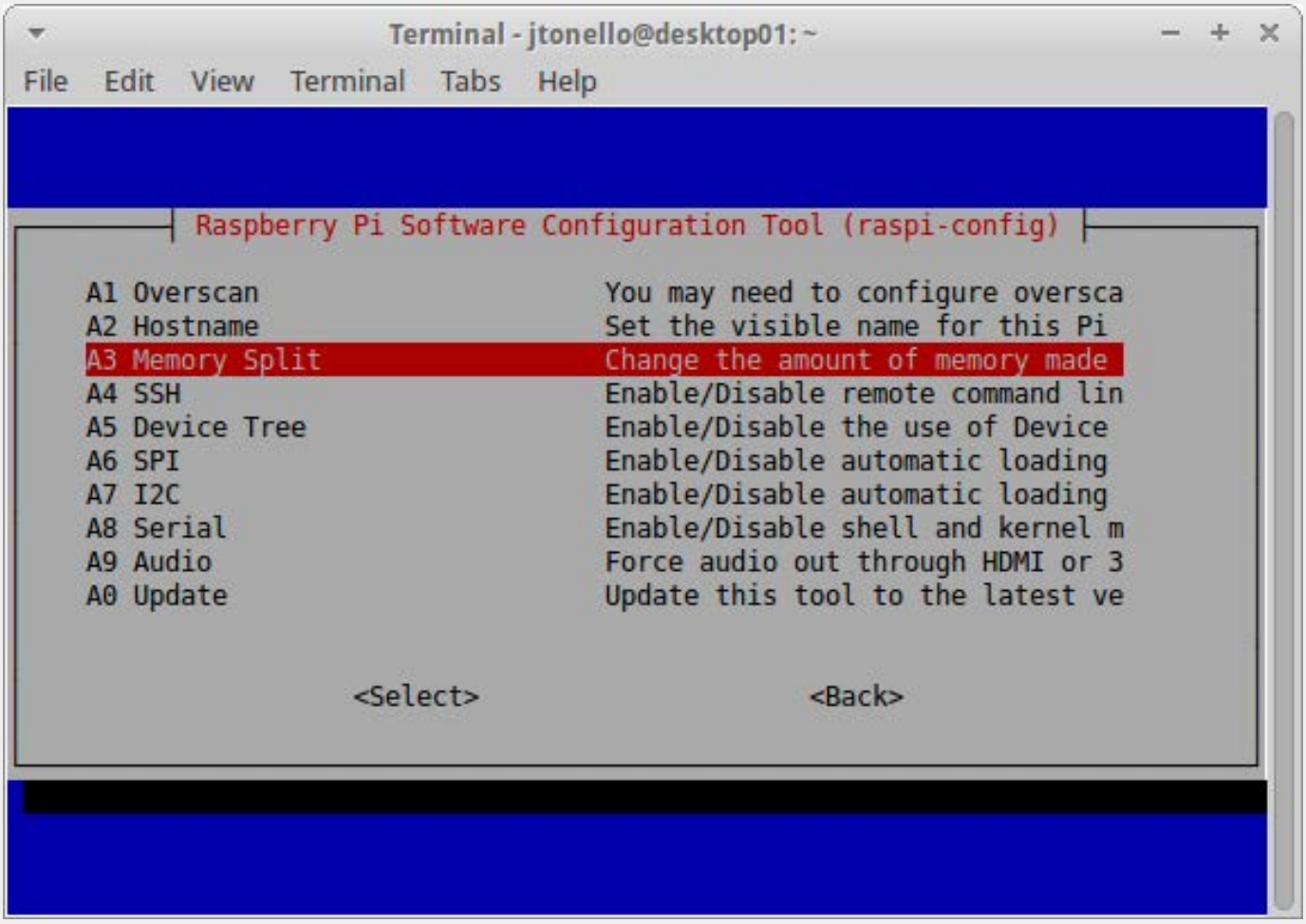


Figure 6. Memory Split

and re-run raspi-config from there. In this unlikely event, repeat the previous steps from a remote shell and choose the Enable overscan option.

Configure Memory Split

Now you can make the modification that will make the biggest difference for this project—changing how the RPi allocates its 1GB of on-board memory:

```
$ sudo raspi-config
```

Enter the Advanced Options menu again, this time selecting the Memory Split (A3) option (Figure 6).

You'll be offered several choices: 16, 32, 128, 256 and 512. Out of the box, the RPi commits 64MB of its 1GB to the GPU. That's not nearly enough to give the pixel resolution we want. After some experimenting, I found that maxing out the memory for the GPU (512) worked best (Figure 7).

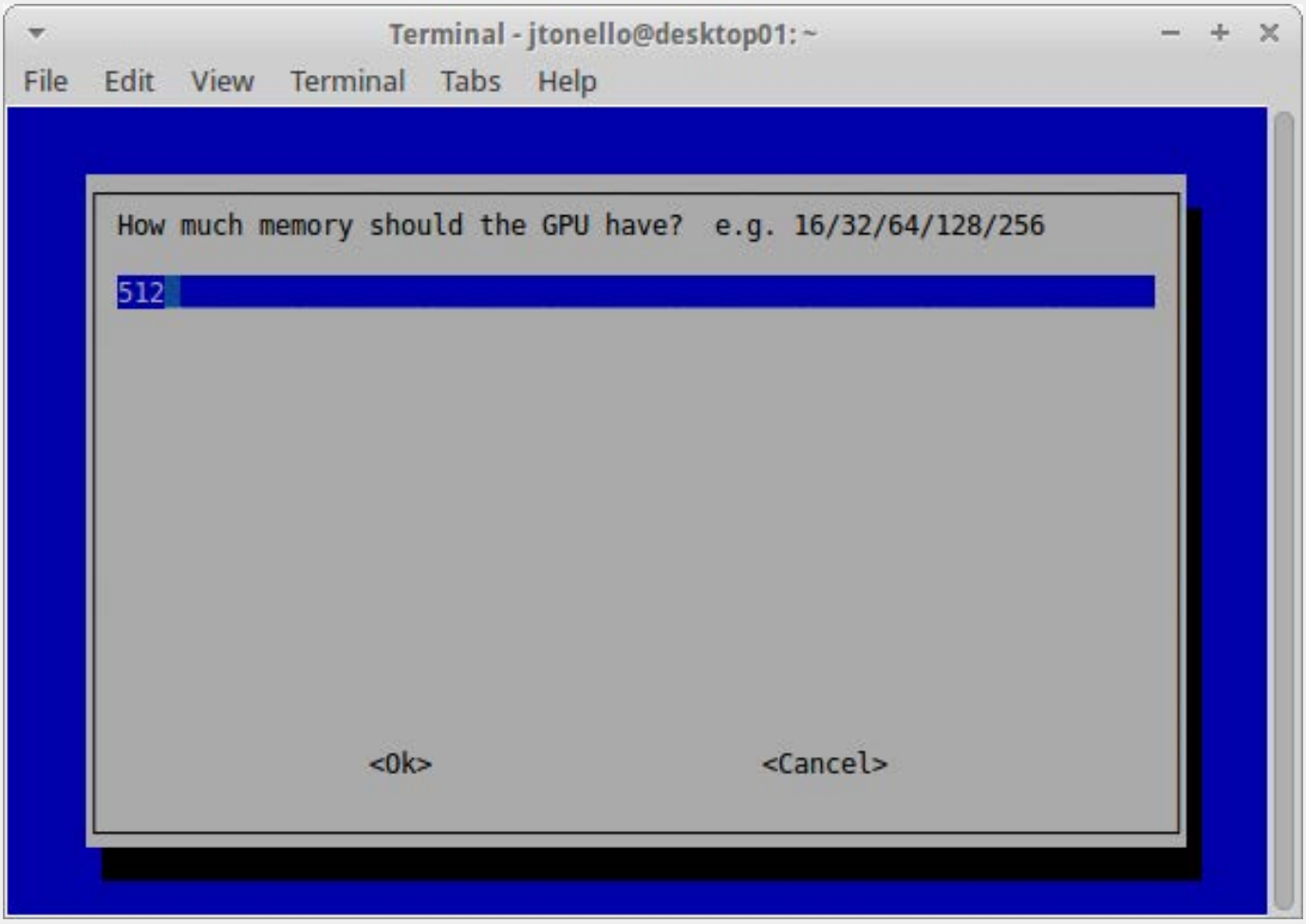


Figure 7. Maxing Out the Memory

This relatively massive amount of video memory delivers the graphical performance you'll need for Ultra HD screens. Still, don't expect it to perform like a desktop. Some graphical lag will remain when you resize or move application windows.

Tab to Ok, tab to Finish and return to the command line. Reboot to make the memory changes take effect.

Modify the Boot Configuration

Once the overscan, overclocking and memory split modifications are complete, the rest of the heavy lifting for this project is done in the boot configuration script: /boot/config.txt. See the Resources section for a link to the complete description of all its features.

Use your favorite text editor to open the file as root:

```
$ sudo vi /boot/config.txt
```

The in-file documentation is helpful and will give you easy-to-follow tips about the various parameters. In this file, you'll also see some of the changes you previously made using raspi-config. If you didn't use raspi-config, you can make the necessary modifications to overscan, overclocking and memory split in the following steps.

First, disable overscanning:

```
disable_overscan=1
```

If you set this in raspi-config, this already will be uncommented and set to 1. Below that entry, you'll see overscan adjustments. These all should be commented out. The same goes for the `framebuffer_width` and `framebuffer_height` settings. Unless you have a particular display in mind, comment out these lines:

```
# uncomment this if your display has a black border
# of unused pixels visible and
# your display can output without overscan
disable_overscan=1

# uncomment the following to adjust overscan. Use
# positive numbers if console goes off screen,
# and negative if there is too much border
#overscan_left=16
#overscan_right=16
#overscan_top=16
#overscan_bottom=16
```

If the RPi doesn't automatically detect your HDMI display, uncomment the `hdmi_force_hotplug=1` line. That should fix it.

In order to get the RPi to support the Ultra HD screen resolution, you'll need to use `hdmi_group` and `hdmi_mode` to enable custom settings. The `hdmi_group` parameter sets the type of display you're using: 0 will auto-detect, 1 will set CEA (typical for televisions), and 2 will set DMT (typical for monitors).

Although I'm using a Vizio TV, I needed to set `hdmi_group` to 2:

```
# Set monitor mode to DMT
hdmi_group=2
```

Depending on the screen you choose, you may need to experiment with these settings a bit. Remember, if something goes wrong and you no longer can read the screen, `ssh` in to your RPi and back out your changes.

Driving the Vizio TV requires a custom resolution that is not offered in any of the preset modes, so you'll need to set the HDMI output format to 87, a custom mode:

```
# Make our custom resolution the default
hdmi_mode=87
```

For a complete list of `hdmi_mode`

values (including resolution and frequency), see the Config.txt link in the Resources section of this article.

With the custom mode set, you now need to add the specific parameters for Coordinated Video Timings (CVT). The format is:

```
hdmi_cvt=<width> <height> <framerate> <aspect>
➥<margins> <interlace> <rb>
```

So, a simplified version of this entry for the 3840x2160 Vizio TV looks like this:

```
hdmi_cvt 3840 2160 24
```

Again, you can modify these settings to match your screen's parameters. It's much like adjusting the screen resolution on any computer monitor—width, height and framerate are key.

I also set the framebuffer width and height to match my `hdmi_cvt` width and height, and then set a high pixel frequency limit:

```
max_framebuffer_width=3840
max_framebuffer_height=2160
hdmi_pixel_freq_limit=400000000
```

After some trial and error, these settings worked well. Additional details are available in the RPi Config (see Resources).

If you didn't use raspi-config to set overclocking and the GPU memory split, add those parameters to the end of /boot/config.txt now:

```
arm_freq=900
gpu_mem=512
```

Your entire /boot/config.txt settings now should look something like this:

```
disable_overscan=1
hdmi_group=2
hdmi_mode=87
hdmi_cvt 3840 2160 24
max_framebuffer_width=3840
max_framebuffer_height=2160
hdmi_pixel_freq_limit=400000000
arm_freq=900
gpu_mem=512
```

Save /boot/config.txt and reboot. When the RPi comes back up, you should have the high-resolution view you want.

Deploying a second RPi is as simple as duplicating the SD card from the RPi you just set up and plugging it in to a second RPi. (See Resources for information on how to do that.)

Mouse and Keyboard Sharing

Because I have both RPis running in graphical mode, I need a keyboard

With x2x, you can move the mouse (and keyboard focus) from one RPi to the other, one monitor to the other, as though the screens were attached to a single computer. It's fast and seamless.

and mouse, but in the NYSERNet, Inc., command center, eliminating clutter is an important goal. When it came time to control the RPis, I didn't want a bunch of keyboards and mice lying around. I wanted just one of each, and I wanted those to control the two computers wirelessly. The answer was SSH and its x2x feature. (See Resources for a full *Linux Journal* x2x how-to.)

With x2x, you can move the mouse (and keyboard focus) from one RPi to the other, one monitor to the other, as though the screens were attached to a single computer. It's fast and seamless.

I attached a Bluetooth USB dongle to the primary RPi I called rpi01. It was immediately detected by the system and connected my Bluetooth keyboard and mouse. I tested the functionality with the window manager, and everything worked fine, although I needed to adjust the mouse speed a bit so it would traverse the vast desktop more quickly.

The RPi is friendly to most modern Bluetooth USB adapters, so you should be able to get one to be plug-and-play. From there, you'll want to share the mouse and keyboard with your secondary RPi (in my case, rpi02). Since rpi01 was my "primary" RPi, and I wanted to connect to rpi02, my "secondary" RPi located to the left (or `-west`) of the other, I opened a terminal on rpi01 and entered the following:

```
pi@rpi01:~$ ssh -X pi@rpi02 x2x -west -to :0
```

This example assumes you're logged in to your primary RPi as the user pi and you want to connect to your secondary RPi as the user pi. Another way of putting it is this: if your Bluetooth mouse and keyboard are working on rpi01 and you want to connect to rpi02 at 192.168.1.11, issue the command:

```
pi@rpi01:~$ ssh -X pi@ 192.168.1.11 x2x -west -to :0
```

The `:0` indicates the primary display. Once connected, the mouse now will move seamlessly from one screen to the other, and the focus for the keyboard will follow. If you want, you can create a script and drop it on the desktop of the primary RPi so you quickly can run SSH x2x after a reboot.

For those who want to do this from two or more systems that don't use Linux, Synergy is a good choice. It's a $10 application that works across Linux, Mac and Windows.

Set Up VNC as a Remote-Control Alternative

If you'd rather not have any keyboard and mouse cluttering your public space where you plan to hang your monitors (or because you just don't use them with the RPis very much), a good alternative to SSH x2x is VNC.

It used to be rather difficult to set up VNC on Linux and access the active session, but that changed with the advent of the vino package. Now, in-session VNC works right out of the box with a simple graphical control panel.

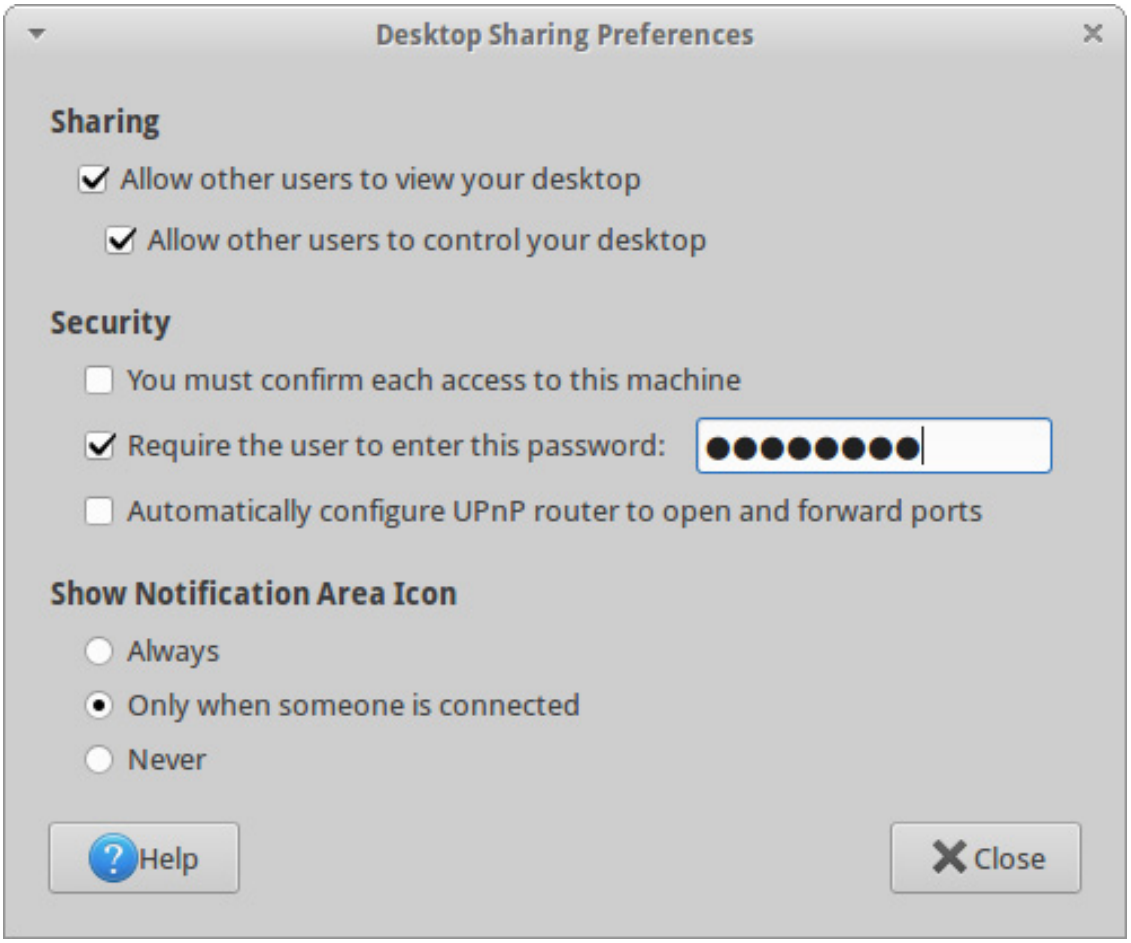


Figure 8. VNC Desktop Sharing Preferences

Use your package manager to install vino on each target RPi. Once installed, open a terminal and run the vino setup utility:

```
$ vino-preferences
```

Under Sharing, enable both "Allow other users to view your desktop" and "Allow other users to control your desktop". Uncheck "You must confirm each access to this machine", so you don't have to acknowledge the connection on the target RPi. Also check "Require the user to enter this password", and enter a password to thwart any unauthorized access.

Click close and reboot the RPi. This will ensure that vino-server will start. In future boots, vino-server will start automatically. Again, the goal here is to set up these RPis so you can do everything remotely and not have to log in to start basic services.

Once the RPi reboots, repeat the vino package install and setup on any additional RPis you're using, then go to your desktop and install a VNC-compatible remote-desktop client. There are many options for all platforms, but I prefer Remmina on my Linux Mint 17 desktop. Remmina supports VNC and RDP, and it makes it easy to throttle the graphical resolution and connection speeds to customize the remote-user experience.

Use your package manager to install Remmina. It comes with VNC support out of the box, but you also can install remmina-plugin-vnc manually.

Once installed, launch Remmina from the command line by entering `remmina` (or open the graphical control panel):

```
$ remmina
```

Create a new connection with appropriate parameters (Figure 9).

Create a profile for each RPi to which you want to connect. Give each profile a name, ensure that you've selected the VNC protocol, enter the server address for the RPi, use the default RPi account (usually pi unless you've changed it), and set the color depth and quality.

I used a color depth of 256 colors and "Poor" quality, because I wanted a better remote-user experience. If you get greedy with the color and quality, the RPi will struggle a bit to redraw and refresh the VNC window, which is about four times the size of a typical desktop monitor.

Click Save or Connect, and you'll

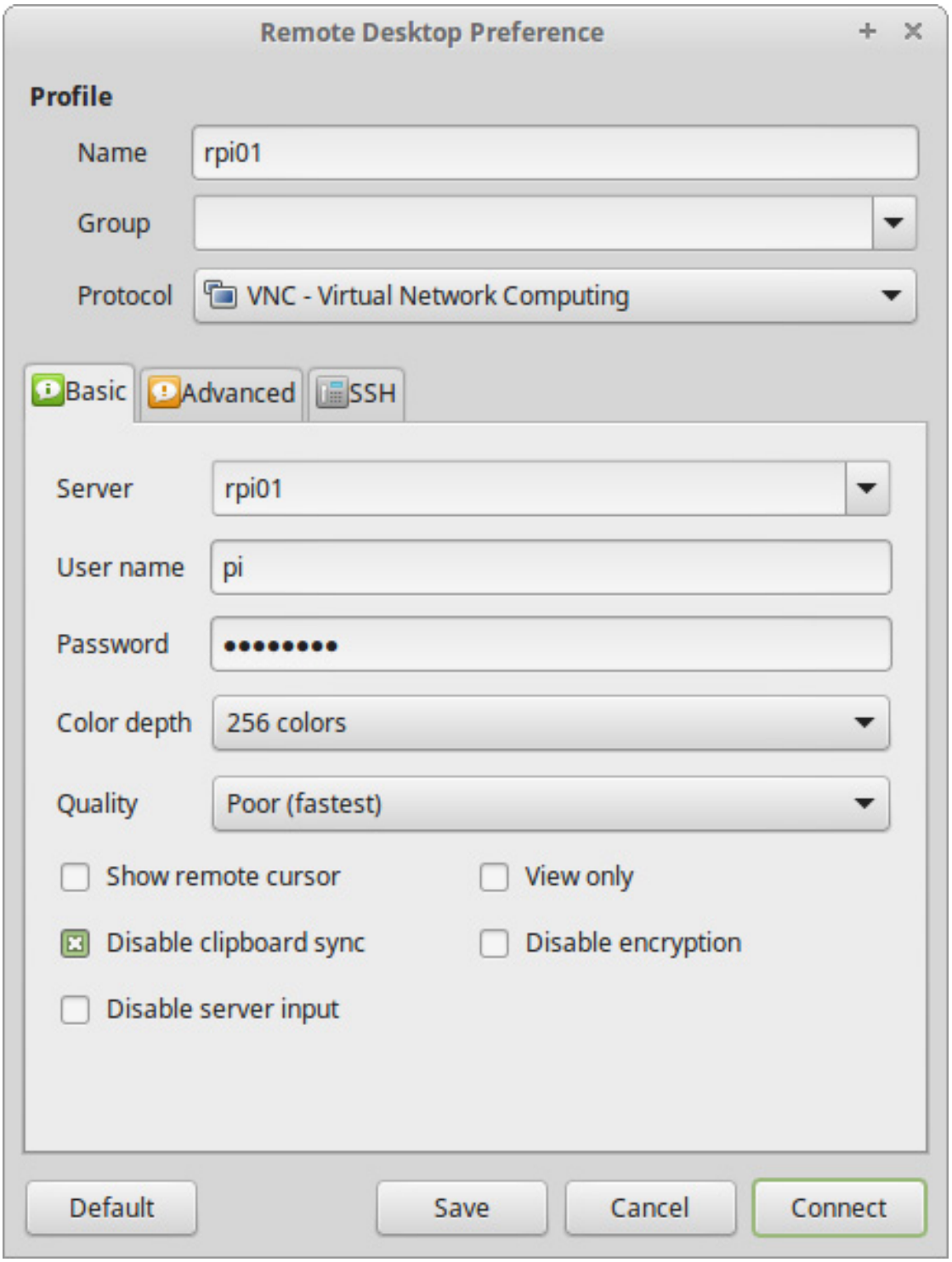


Figure 9. Creating a Profile

connect to your RPi. Be patient. I've found that this connection is not instantaneous even across a fast LAN. It takes several seconds. Once connected, you can use Remmina to show a full-screen view and use its window sliders to navigate to areas of the RPi desktop that don't fit on

your smaller desktop monitors.

Before hanging the new 50-inch TVs in our command center, I did my final testing with everything still on the ground. I booted each RPi, installed and launched the applications I needed, tested the networking and the remote connections, and left the whole thing running for a few days. Once I felt confident everything was stable, we hung the screens on low-profile wall-hangers, connected the RPis with the short HDMI cables and tucked them behind the TVs. Anyone looking sees only the monitors; the rest is neatly hidden away.

Final Thoughts

Because your RPis probably are sitting on a network, it's a good idea to secure them with a firewall. I set up some iptables rules that block almost all traffic.

Also, to make the RPis even easier to maintain, you may want to install Webmin, a free Web-based administration tool for Linux environments. It works well under Raspbian and doesn't hog resources when it's not in use.■

John S. Tonello is Director of IT for NYSERNet, Inc., in Syracuse, New York. He's been a Linux user and enthusiast since he installed his first Slackware system from diskette 20 years ago.

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

Resources

NOOBS RPi Config: https://www.raspberrypi.org/documentation/installation/noobs.md

Config.txt Documentation: https://www.raspberrypi.org/documentation/configuration/config-txt.md

RPi Config Documentation: http://elinux.org/RPiconfig

Duplicate SD Card: https://www.raspberrypi.org/forums/viewtopic.php?f=91&t=46911

Linux Journal x2x How-To: http://www.linuxjournal.com/content/share-keyboardmouse-between-multiple-computers-x2x

Synergy Mouse and Keyboard Sharing: http://www.synergy-project.org

Attend

InterDrone

The International Drone Conference and Exposition

InterDrone is Three Awesome Conferences:

Drone TECHCON

For Builders

More than 35 classes, tutorials and panels for hardware and embedded engineers, designers and software developers building commercial drones and the software that controls them.

Drone FLYER

For Flyers and Buyers

More than 35 tutorials and classes on drone operations, flying tips and tricks, range, navigation, payloads, stability, avoiding crashes, power, environmental considerations, which drone is for you, and more!

Drone BUSINESS

For Business Owners, Entrepreneurs & Dealers

Classes will focus on running a drone business, the latest FAA requirements and restrictions, supporting and educating drone buyers, marketing drone services, and where the next hot opportunities are likely to be!

The Largest Commercial Drone Show in North America

Register Today!
Meet with 80+ exhibitors!
Demos! Panels! Keynotes!
The Zipline!

September 9-10-11, 2015
Rio, Las Vegas
www.InterDrone.com

A BZ Media Event

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

Dealing with Boundary Issues

Fogged by The Cloud.

The other evening a bunch of us were sitting in a friend's living room while a series of photos scrolled across her TV. The photos were a screen saver served up by her new Apple TV box. Some of the pictures were of people, birds, flowers, cats and other typical stuff. But in the mix were also shots of price tags, bar codes, bags of mulch and other stuff she had thought about buying at some point.

"What are those doing there?" somebody asked.

"I don't know", she said. "I thought I threw those away."

"That's scary", somebody said. "What if some of your shots were nude selfies, or porn?"

"I don't have that", she replied. "But I don't like not knowing why shots I never intended to share are showing up on my TV."

Even though most people in the room were Apple customers, and some knew a bit about Apple TV, the room itself had no clear sense about how Apple TV decides what photos to show, or whether the photos on the screen come from the user's phone, laptop or iCloud (the non-place where Apple may or may not store some or all of what customers shoot with their cameras or phones). So finally, a rough agreement emerged out of collective un-knowing.

"It's probably the cloud", somebody said.

"Yeah, the damn cloud", another agreed. What else to blame than the most vague thing they all knew?

Then my wife said something that knocked me over.

"The cloud has boundary issues."

Her point was about the psychology of the cloud, not its technology. In psychology, boundary issues show up when one person doesn't respect

the boundaries set by another. And in today's networked world, personal boundaries are ignored as a matter of course, mostly because people don't have ways to set them.

In pre-Internet days, personal computers came with their own built-in boundaries. Even when they were connected to shared printers and filesystems, it was clear where the personal stuff ended and the shared stuff began.

In today's networked world, smart geeks mostly still know where local stuff ends and remote (cloud-located) stuff begins—or at least where the logical boundaries are. But for most of the rest of us, most of the time, the cloud is a hazy thing. And that haze has been troubling us ever since we chose "the cloud" over better terms that would have made more sense. (Such as calling it a "utility", which Nicholas Carr suggested in his 2010 book *The Big Switch*.)

According to Google's Ngram Viewer, mentions of "cloud computing" in books hockey-sticked in 2004 and continued to rise after that Usage of "the cloud" also rose over the same period. According to Google Trends, which looks at search terms, "the cloud" hockey-sticked in 2011 and flattened after that, because its usage became ubiquitous.

William F. Buckley, Jr., once said his purpose in life was "to stand athwart history, yelling 'stop'". In a similar way, Richard M. Stallman stood athwart history when, in September 2008, he called the cloud "worse than stupidity: it's a marketing hype campaign". "In What Does That Server Really Serve?", published in March 2010, RMS wrote:

> The information technology industry discourages users from considering … distinctions. That's what the buzzword "cloud computing" is for. This term is so nebulous that it could refer to almost any use of the Internet. It includes SaaS, and it includes nearly everything else. The term only lends itself to uselessly broad statements.
>
> The real meaning of "cloud computing" is to suggest a devil-may-care approach towards your computing. It says, "Don't ask questions, just trust every business without hesitation. Don't worry about who controls your computing or who holds your data. Don't check for a hook hidden inside our service before you swallow it." In other words, "Think like a sucker." I prefer to avoid the term.

About SaaS (Software as a Service), he adds:

> These servers wrest control from the users even more inexorably than does proprietary software. With proprietary software, users typically get an executable file, but not the source code. That makes it hard for programmers to study the code that is running, so it's hard to determine what the program really does, and hard to change it.
>
> With SaaS, the users do not have even the executable file: it is on the server, where the users can't see or touch it. Thus it is impossible for them to ascertain what it really does, and impossible to change it.
>
> Furthermore, SaaS automatically leads to harmful consequences equivalent to the malicious features of certain proprietary software. For instance, some proprietary programs are "spyware": the program sends data about users' computing activities to the program's owner....
>
> SaaS gives the same results as spyware because it requires users to send their data to the server. The server operator gets all the data with no special effort, by the nature of SaaS.

And, sure enough, spying on individuals by commercial entities is now so normal and widespread that most of us living in the networked world can hardly tell when it's not happening. This is why ad and tracking blockers have become so popular, and why survey after survey shows that more than 90% of people are concerned about privacy on-line.

But spying is just the symptom. The deeper issue is boundaries. In the networked world we don't have them, and because of that, others take advantage of us.

"Give me a place to stand and I can move the world", Archimedes said. Note that he didn't say, "Give a lot of people a place to stand." He meant himself. And his point applies to personal boundaries as well. Each of us needs to set our own. That's our fulcrum.

Free software provides a good model for approaching this problem, because its framing is personal:

> "Free software" means software that respects users' freedom and community. Roughly, it means that the users have the freedom to run, copy, distribute, study, change and

> improve the software. Thus, "free software" is a matter of liberty, not price....We campaign for these freedoms because everyone deserves them.

Most of us don't write software, but all of us create and use data. We also have containers for data in our computers. We call those containers *files*, which live in *directories*, which are visualized as *folders* in graphical user interfaces.

As a mental concept, containers could hardly be more important. In "The Metaphorical Structure of the Human Conceptual System", George Lakoff and Mark Johnson say, "the nature of the human conceptual system...is fundamentally metaphorical in character" and involves "ontological concepts arising in physical experience, among which is the container". They provide examples:

> When you have a good idea, try to capture it immediately in words. Try to pack more thought into fewer words. His words carry little meaning. Your words seem hollow. The ideas are buried in terribly dense paragraphs. I can't get the tune out of my mind. He's empty-headed. His brain is packed with interesting ideas. Do I have to pound these statistics into your head? I need to clear my head.

All those statements presume containers: spaces with boundaries. We have those with files, directories, folders, drives and storage systems. We don't with clouds, beyond knowing that something resides "in the cloud". We also don't with cookies and tracking beacons that get injected (from clouds) into who knows where inside our computers and mobile devices.

In the absence of boundaries we set, controlling entities (such as Amazon, Apple, Facebook and Google) assume that boundary-setting is their job and not ours, because so much of the world's work can only be done by big systems that enjoy degrees of "scale" not available to individual human beings.

Yet personal freedom has scale too, if it is grounded in our nature as sovereign and independent human beings. This is why free software was such a powerful idea in the first place, and why it remains the world's most rigorous guide for software development. We may not always obey it, but we always know it's there, and why it is valuable.

While free software respects code and what can be done with

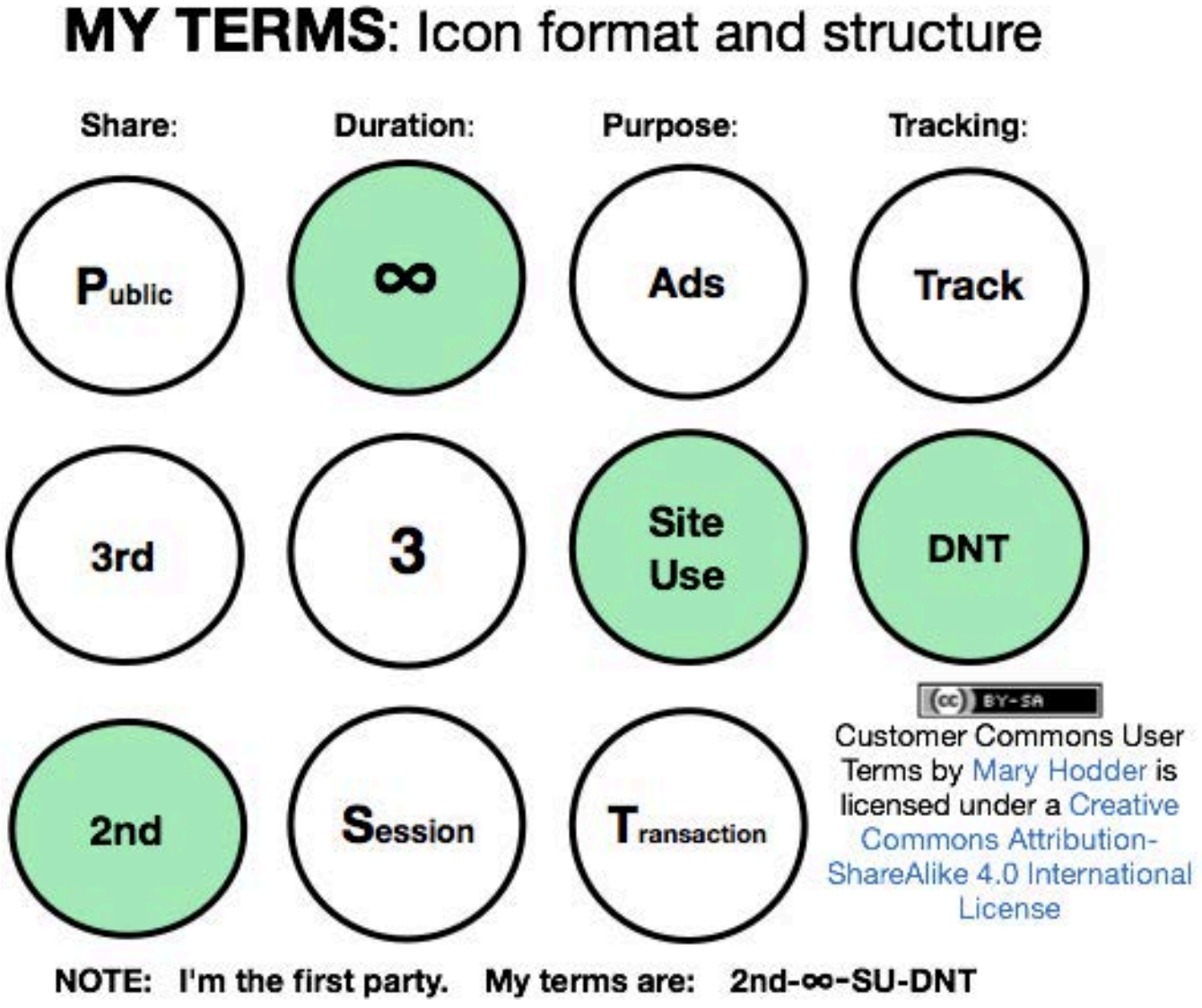


Figure 1. User Terms (from Mary Hodder at the 19th Internet Identity Workshop)

it, we need a similar set of guiding principles that respect personal data and what can be done with that. I believe these should be terms that we assert, and to which others can agree, preferably automatically.

In fact, there is a lot of work going on right now around personal terms. I am involved with some of it myself, through Customer Commons. Figure 1 shows one straw-man example of terms, drawn up by Mary Hodder at the 19th Internet Identity Workshop (IIW) last year.

Terms and privacy policies (that we assert, as individuals) will also be on

the table at two events this month.

The first is the 21st IIW, which takes place twice per year at the Computer History Museum, in the heart of Silicon Valley. It's a three-day unconference, with no keynotes and no panels—just breakouts on topics of the participants' choosing. It costs money, but it's cheap compared to most other conferences, and it's far more intense and productive than any other conference I know. The dates are October 27–29, 2015. Register here: http://iiwxxi-21.eventbrite.com.

The second is VRM Day, which happens on October 26, 2015, the day before IIW, in the same location. The purpose is to discuss VRM (Vendor Relationship Management) development, which is all about freedom and independence for individuals in the commercial world. It's free, and you can register here: http://vrmday2015b.eventbrite.com.

See you there.■

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

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

Advertiser Index

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

Resources

Mentions of “cloud computing” in books according to Google’s Ngram Viewer (2000–2008): https://books.google.com/ngrams/graph?content=cloud+computing&year_start=2000&year_end=2008&corpus=15&smoothing=3&share=&direct_url=t1%3B%2Ccloud%20computing%3B%2Cc0t1;,cloud%20computing;,c0

Usage of “the cloud” in books according to Google’s Ngram Viewer (2000–2008): https://books.google.com/ngrams/graph?content=the+cloud&year_start=2000&year_end=2008&corpus=15&smoothing=3&share=&direct_url=t1%3B%2Cthe%20cloud%3B%2Cc0t1;,the%20cloud;,c0

Google Trends “the cloud” search term (2004 to present): https://www.google.com/trends/explore#q=%22the%20cloud%22

William F. Buckley, Jr.: https://en.wikipedia.org/wiki/William_F._Buckley,_Jr.

Richard M. Stallman: https://stallman.org

“Cloud computing is a trap, warns GNU founder Richard Stallman” by Bobbie Johnson: http://www.theguardian.com/technology/2008/sep/29/cloud.computing.richard.stallman

“What Does That Server Really Serve?” by Richard M. Stallman: http://www.bostonreview.net/richard-stallman-free-software-DRM

The 2015 Ad Blocking Report by the PageFair Team: http://blog.pagefair.com/2015/ad-blocking-report

“Americans’ Attitudes About Privacy, Security and Surveillance” by Mary Madden and Lee Rainie (Pew Research Center): http://www.pewinternet.org/2015/05/20/americans-attitudes-about-privacy-security-and-surveillance

2015 TRUSTe US Consumer Confidence Index: https://www.truste.com/resources/privacy-research/us-consumer-confidence-index-2015

“Lying and Hiding in the Name of Privacy” by Mary Hodder and Elizabeth Churchill (Customer Commons): http://customercommons.org/2013/05/08/lying-and-hiding-in-the-name-of-privacy

Philosophy of the GNU Project: http://www.gnu.org/philosophy/philosophy.en.html

The Free Software Definition: http://www.gnu.org/philosophy/free-sw.en.html

“The Metaphorical Structure of the Human Conceptual System” by George Lakoff and Mark Johnson: http://www.fflch.usp.br/df/opessoa/Lakoff-Johnson-Metaphorical-Structure.pdf

Customer Commons: http://customercommons.org

Vendor Relationship Management (VRM): https://en.wikipedia.org/wiki/Vendor_relationship_management

peer1 hosting

Where every interaction matters.

break down
your innovation barriers

power your business to its full potential

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

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

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

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

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

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!