

LINUX™ JOURNAL

Since 1994: The Original Magazine of the Linux Community

Categorize Documents
with Machine Learning

The Importance
of Good Ticketing
Systems for Sysadmins

Tips for Cutting
the Cable Cord

THE SPACE ROBOTICS CHALLENGE and Linux

Solve
Physics
Problems
with Linux



Software
Updates,
Embedded
Linux and
the IoT



APRIL 2017 | ISSUE 276
<http://www.linuxjournal.com>



WATCH:
ISSUE
OVERVIEW



**Practical books
for the most technical
people on the planet.**

GEEK GUIDES



**Download books for free with a
simple one-time registration.**

<http://geekguide.linuxjournal.com>

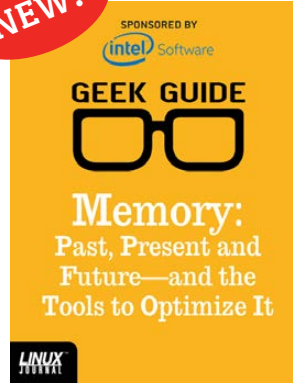
NEW!



An Architect's Guide: Linux for Enterprise IT

Author: Sol Lederman
Sponsor: SUSE

NEW!



Memory: Past, Present and Future—and the Tools to Optimize It

Author: Petros Koutoupis
Sponsor: Intel



Cloud-Scale Automation with Puppet

Author: John S. Tonello
Sponsor: Puppet



Why Innovative App Developers Love High-Speed OSDBMS

Author: Ted Schmidt
Sponsor: IBM



Tame the Docker Life Cycle with SUSE

Author: John S. Tonello
Sponsor: SUSE



SUSE Enterprise Storage 4

Author: Ted Schmidt
Sponsor: SUSE



BotFactory: Automating the End of Cloud Sprawl

Author: John S. Tonello
Sponsor: BotFactory.io



Containers 101

Author: Sol Lederman
Sponsor: Puppet

CONTENTS

APRIL 2017
ISSUE 276

FEATURES

76 Robots and Linux, a Match Made in Space

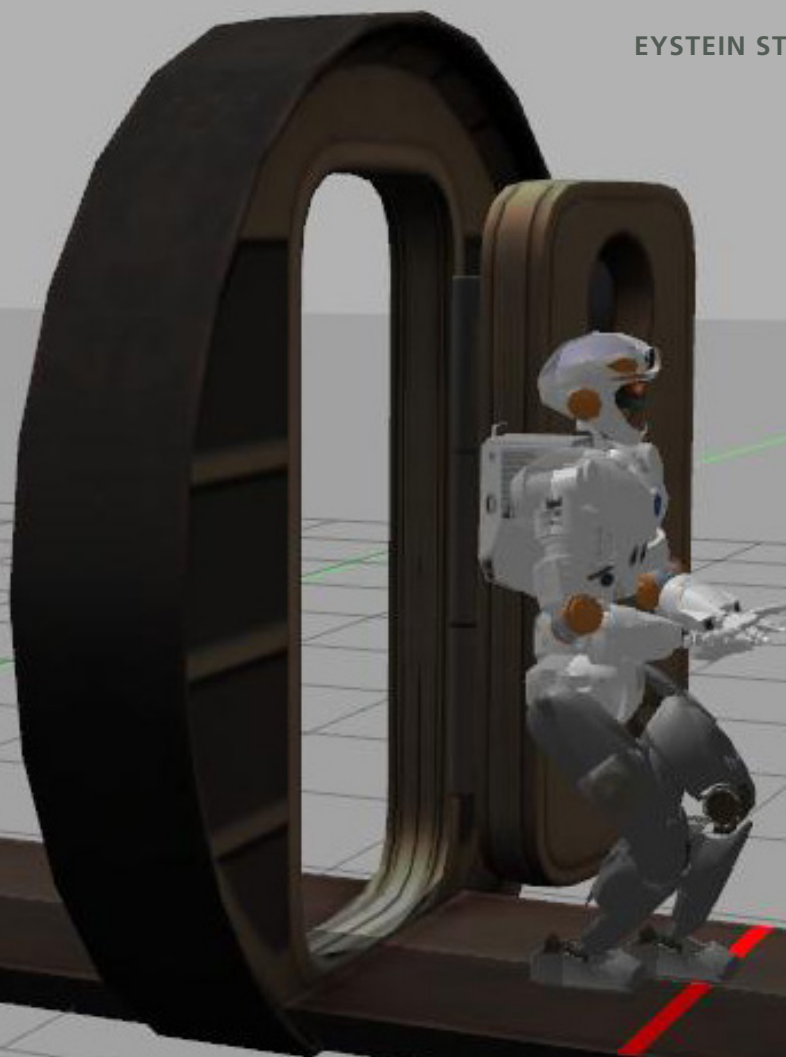
A look at NASA's Space
Robotics Challenge.

PAUL FERRETTI

88 Key Considerations for Software Updates for Embedded Linux and IoT

Don't brick the device!

EYSTEIN STENBERG



COLUMNS

32 **Reuven M. Lerner's
At the Forge**

Classifying Text

40 **Dave Taylor's
Work the Shell**

Watermarking Images—from
the Command Line

46 **Kyle Rankin's
Hack and /**

Sysadmin 101: Ticketing

54 **Shawn Powers'
The Open-Source
Classroom**

Actually Cutting the Cord

104 **Doc Searls' EOF**

How to Fix the Edge

IN EVERY ISSUE

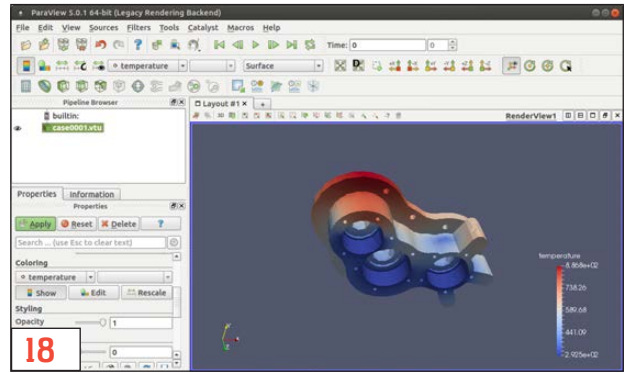
8 **Current_Issue.tar.gz**

10 **UPFRONT**

30 **Editors' Choice**

68 **New Products**

110 **Advertisers Index**



ON THE COVER

- The Space Robotics Challenge and Linux, p. 76
- Software Updates, Embedded Linux and the IoT, p. 88
- Categorize Documents with Machine Learning, p. 32
- The Importance of Good Ticketing Systems for Sysadmins, p. 46
- Tips for Cutting the Cable Cord, p. 54
- Solve Physics Problems with Linux, p. 18

LINUX JOURNAL™

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



ENJOY:

Timely delivery

Off-line reading

Easy navigation

Phrase search
and highlighting

Ability to save, clip
and share articles

Embedded videos

Android & iOS apps,
desktop and
e-Reader versions

SUBSCRIBE TODAY!

LINUX JOURNAL

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

Contributing Editors

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

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

Editorial Advisory Panel

Nick Baronian
Kalyana Krishna Chadalavada
Brian Conner • Keir Davis
Michael Eager • Victor Gregorio
David A. Lane • Steve Marquez
Dave McAllister • Thomas Quinlan
Chris D. Stark • Patrick Swartz

Advertising

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

Subscriptions

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

LINUX is a registered trademark of Linus Torvalds.

**STORAGE
REDEFINED:**

**You
cannot
keep up
with data
explosion.**

Manage data expansion with SUSE Enterprise Storage.

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

suse.com/storage



If There Was a Problem? Yo, I'll Solve It...

I'm not sure what problem Vanilla Ice solved with his DJ's sick hook, but thankfully in the Linux world, we solve problems all the time. In fact, Linux exists as a solution to a problem—25+ years later, and Linux is still solving problems all the time. This month, let's attack some problems the penguin way.

Reuven M. Lerner starts out with how to program computers to do work we'd rather not do ourselves. Specifically, Reuven shows how to automate the sorting of documents. Although the process might not be quite as boring as collating paperwork, sorting documents by subject is monotonous at best. Reuven describes how to program away this annoying task. Dave Taylor follows with the problem of potential photograph theft. With your photos on the internet, it's very easy for someone, even someone with innocent intent, to violate your copyrights. Adding a watermark to your photos is a great way to inform viewers that you own the graphic. Dave shows how to do it from the command line.

Next, we look at managing problems with Kyle Rankin. Even problem-solving systems have problems, and if you're in a big environment, it's important to track the management of issues. Kyle explains



**SHAWN
POWERS**

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



VIDEO:
Shawn Powers runs through the latest issue.

the process for implementing a successful ticketing system in your organization. In my early days of being a system administrator, we had green sheets of paper where users would write their problems and leave them in a basket for me. Those “Green Sheets” provided an incredible service to me and my users. Now, there are many digital systems that provide far more useful information than the papers in my basket.

The problem I solve this month is cable TV. I talked about cord cutting in the past, but I finally boxed up my TiVo units and went the route of the antenna. It’s a complicated endeavor to set up an antenna for broadcast channels, especially if you’re in a fringe area. Add to that the need for integrating OTA channels into my network-based entertainment system, and it’s an article worth reading if you want to cut the cord!

Paul Ferretti talks about three of my favorite topics in a single article. Linux, robots and space are pretty much the keywords I look for in any book I read. Paul talks about NASA’s work reaching out for problem solvers who want to create robots that can do significant work in the space program. If you’re looking to make reliable robots, it makes a lot of sense to use Linux as your foundation.

In fact, many embedded systems in robots use Linux. Eystein Stenberg follows Paul with a look at keeping those embedded systems updated. All too often once an embedded system is put into production, the operating system is never updated. It does a single job, and it does it well, so why upgrade? Well, if those systems are connected to a network, they’re a huge vector for attack. Eystein walks through the complicated world of keeping embedded systems secure.

Regardless of the problem you need to solve, if you’re like me, Linux is always one of the first options to consider. Whether it’s a real-world problem like your car’s embedded computer getting hacked or an intellectual issue regarding photograph copyright, this month, we focus on fixing things. We hope this issue is as useful to you as it was to all of us who put it together!■

[RETURN TO CONTENTS](#)



PREVIOUS
Current_Issue.tar.gz

NEXT
Editors' Choice



diff -u

What's New in Kernel Development

In the quest for ever-larger systems, the Linux kernel has to keep retooling itself. It's not like other software where you can just build a data structure that can keep growing without bound. At the OS level, access to RAM must be as fast as possible. Any unnecessary slowdown will be multiplied by however much work the system tries to do.

So, when **Liang Li** wanted to raise the **x86-64** limit of 46-bit physical addresses to 52-bit, he couldn't simply fix the whole problem by making address size limitless. He had to implement a highly efficient stop-gap that would tide us over until the next time it needed to be increased.

In fact, his patch cranked the limit up to 56 bits, which was what the upcoming hardware would support.

The code hit a snag, however, when **Paolo Bonzini** pointed out that Liang's implementation might interfere with migrating virtual machines from one hardware setup to another. The problem, it turned out, boiled down to a hardware bug that may be difficult to work around. In particular, using **LA57 mode** to access high memory addresses would work on newer systems but not older ones. Thus, doing even one write with LA57 mode would make a virtual system impossible to migrate.

This normally would be solved by catching the LA57 calls and rerouting

At Your Service

them to a workaround on older systems. But in this case, that would reduce efficiency on the older systems, which would make that particular change very unattractive.

It's possible that this hardware bug may slow down the adoption of Liang's patch for quite a while. There'll definitely be support for larger address spaces, but the question of how to go about it has not yet been answered.

Like security, data integrity is one of those things that trumps all other considerations. In fact, data integrity might even rank higher than security in overall importance. Fortunately, I don't think that choice has ever had to be made.

Security and data integrity do often go hand in hand, however. Recently, **Yi Zhang** described a security exploit that could allow a hostile user to corrupt data in the **ext4** filesystem. Unsurprisingly, Yi's patch is on the fast track to get in the kernel tree.

Andreas Dilger and **Valdis Kletnieks** both liked Yi's patch and had some feature requests. In particular, Valdis wanted the patch, in addition to preventing the corruption, to alert the user to exactly which filesystem had been attacked.

Yi had bigger ambitions though. He said that the ext4 exploit existed in similar form across many other filesystems. He proposed building the solution into the virtual filesystem (VFS) code itself, where all filesystems had to pass in order to communicate with the user.

Al Viro's the **VFS** guy, and he seemed open to the discussion. At that point, the conversation moved offline, but clearly, this fix will be going in somewhere very soon.

The **RCU** code in Linux (short for read/copy/update)

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

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

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

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

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

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

makes sure that certain data is available to all CPUs during the boot process, before any filesystems have been mounted. It's part of the Narnia that is boot time.

Paul McKenney recently posted some patches to make sure that RCU really made the data available all the way through the boot process, instead of only at certain specific times. In particular, data is generally transferred between CPUs at **Synchronous Grace Periods**, which couldn't be induced properly at a certain phase of the boot process. Meanwhile, the code that typically induces Synchronous Grace Periods had begun to activate during that phase, which was leading to occasional crashes.

Paul posted a patch to make Synchronous Grace Periods work throughout the whole boot process, and a variety of folks offered comments.

There was some talk from **Rafael J. Wysocki**, **Borislav Petkov** and **Lv Zheng** of possibly writing a separate version of Paul's patch for **ACPI**. The general consensus, however, was that it would overly complicate the ACPI code in exchange for not much actual benefit. So that was that.

Kirill A. Shutemov recently posted a patch to allow Linux userspace to access up to 56-bit memory addresses. This dovetails nicely with Liang Li's work mentioned previously. However, it is apparently much more controversial. There are many ways to give userspace this ability, and the impact could affect code all over the kernel. In addition to speed and security, it's important to make sure that the semantics—the ways user code can interact with the kernel—remain consistent, or at least clear.

After Kirill's patches, there was a long and wide-ranging implementation discussion, with **Linus Torvalds** ultimately weighing in on the side of an implementation that was quite different from Kirill's initial conception. There were a variety of complex proposals on the table, involving exactly when and how user code could access high address memory. Predictably, Linus favored an extremely simple solution from the user perspective, in which anyone who wanted a given binary to access high address memory simply could compile that binary with the feature in place.

It's likely that the debate will continue, but at least now there's a clear baseline. Any prospective solution should present the user with a similarly simple interface. Any solution that favors elegance or completeness over that level of simplicity is likely not to make the cut.—Zack Brown

SUPERMICRO[®]

MARKETPLACE

Powered by Silicon Mechanics



**Broad
Selection**



**Zero
Defects**



**3-Year
Warranty**

**Your Source for
Supermicro Platform Technology**

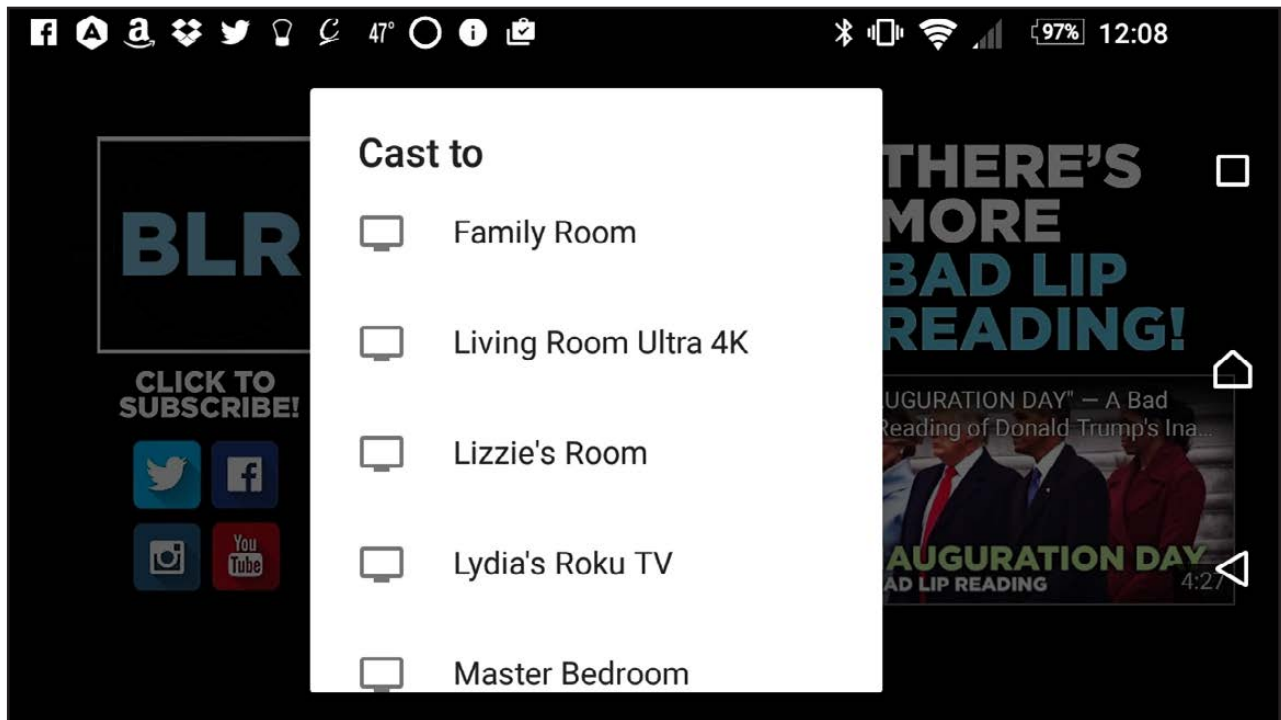
Configure Now



Talk to a Supermicro Expert!

866.352.1173

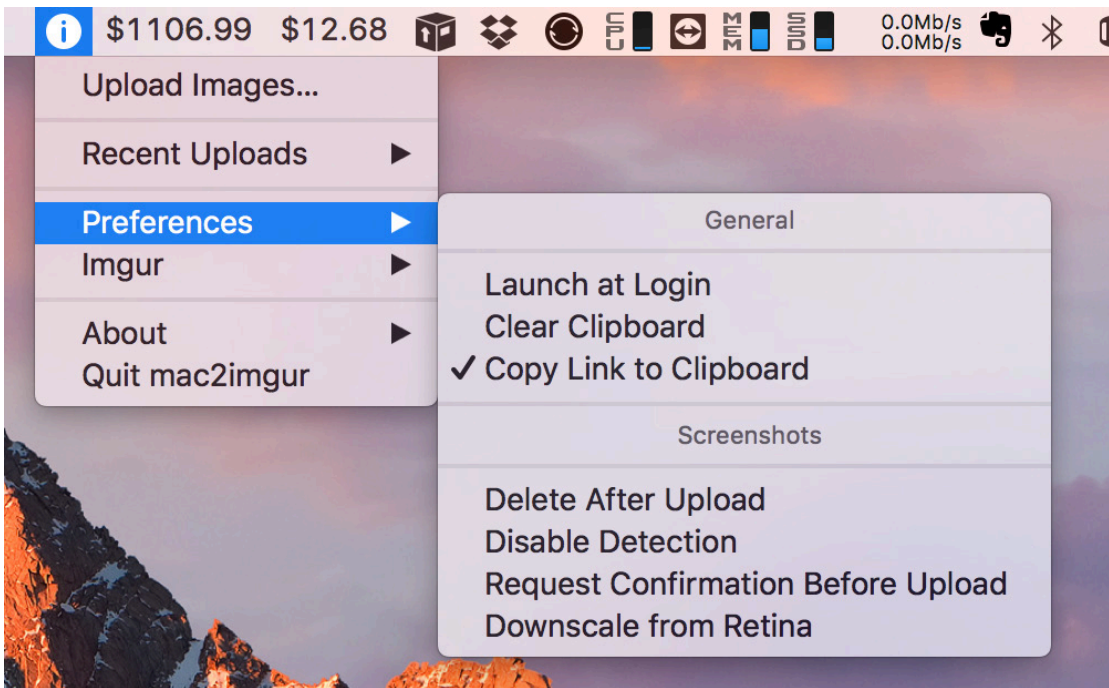
YouTube on the Big Screen



For years I've been jealous of folks with iOS devices who could just send their phone screens to their Apple TV devices. It seems like the Android screen-mirroring protocols never work right for me. My Sony Xperia has multiple types of screen mirroring, and none of them seem to work on my smart TVs or Roku devices.

YouTube is a completely different story. It doesn't matter if I'm on a laptop, iPhone, Android device or Chromebook, I can "cast" the video to any of my Roku devices or smart TVs without any problems at all. It works and works well. The great part about casting is you can shut off the connection from the sending device, and it keeps playing! Because 95% of the stuff I want to display on the TV from my phone is YouTube videos, I couldn't be happier. Plus, I can check email on my phone while the family watches the latest "Bad Lip Reading" video on the big screen! Take that Apple TV!—Shawn Powers

Non-Linux FOSS: Mac2Imgur



I love to share images with people quickly. They could be cat photos or screenshots. Usually I post those silly images to Twitter and Facebook using Buffer, but occasionally, I just want to send a quick image to a single person. (This is usually when I'm trying to show my computer via screenshot.)

When I'm on a Mac, screenshots are really simple with the Command-Shift-3 keystroke. Sharing them used to mean dragging them to my Dropbox, and then finding the image and getting a public share link. It's cumbersome. But, there's a great utility for OS X called Mac2Imgur, which uploads an image to <http://imgur.com> and provides a link in the clipboard. It even has options to remove the image after uploading, so you don't have 100 screenshots on your desktop. It's an incredibly useful utility, and it's completely open source.

If you ever share images from a Mac, head over to <https://github.com/mileswd/mac2imgur> and grab a copy (and yes, this screenshot is also on imgur at <https://i.imgur.com/wv28w31.png>). —Shawn Powers

I'll Gladly Pay You Tuesday for a Hamburger Today

My day job pays me on the 15th and last day of every month, unless those days land on a weekend, in which case I get paid the Friday before. With those rules, creating a Google Calendar event is shockingly difficult. In fact, it's not possible to create a recurring event with those rules using Google's GUI scheduling tool.

Thankfully, you can import an event from an .ical file, and Google will understand the more complex schedules. In my case, I needed two separate events. For the 15th of every month, with weekends being paid on Friday, here is the .ical file:

```
BEGIN:VCALENDAR
BEGIN:VEVENT
DTSTART:20170101
DTEND:20170101
RRULE:FREQ=MONTHLY;BYDAY=FR;BYMONTHDAY=13
RRULE:FREQ=MONTHLY;BYDAY=FR;BYMONTHDAY=14
RRULE:FREQ=MONTHLY;BYDAY=MO, TU, WE, TH, FR;BYMONTHDAY=15
SUMMARY: Get Paid
END:VEVENT
END:VCALENDAR
```

The DTSTART and DTEND dates are set to when you first want the recurring event to start. I just started on January 1, 2017. If you read through the logic, basically the event is placed on Friday if Friday is the 13th, on Friday if Friday is the 14th and on any weekday that lands on the 15th. The rules are evaluated in order, so if Friday is the 13th or 14th, the event is scheduled then.

The last day of the month except for weekends was actually a bit more complicated, but still doable. That event looks like this:

```
BEGIN:VCALENDAR
BEGIN:VEVENT
DTSTART:20170101
DTEND:20170101
RRULE:FREQ=MONTHLY;BYDAY=MO,TU,WE,TH,FR;BYSETPOS=-1;WKST=MO;
SUMMARY: Get Paid
END:VEVENT
END:VCALENDAR
```

The logic is a little harder to follow, and I actually had to adapt a post on Stack Overflow (<http://webapps.stackexchange.com/questions/10929/how-to-make-a-task-repeat-on-the-last-day-of-each-month-in-google-calendar/17531#17531>) to get it to work. They both seem to work flawlessly, so I'm not complaining.

In order to add them to your calendar, you just save two text files with the above text. I'm not sure if the .ical extension is required, but that's the type of file it is, so I recommend using it. Then you "import" a calendar, and you'll be asked to which calendar to add the event. Unfortunately, if you need to make a change to the schedule, you'll have to delete the event and re-create/import it, because it can't be adjusted with the Google GUI. But, the whole point was to have the date figured out so I wouldn't have to edit it anymore.—Shawn Powers

Solving Physics Problems on Linux

Several years ago, I wrote an article on using Elmer to solve complicated physics problems. Elmer has progressed quite a bit since then, so I thought it would be worth taking a fresh look at this simulation software (<https://www.csc.fi/web/elmer>).

The first step is to install Elmer on your system. It may exist within your package management system, but those are likely older versions. If you are running a Debian-based system, you can get the latest versions by

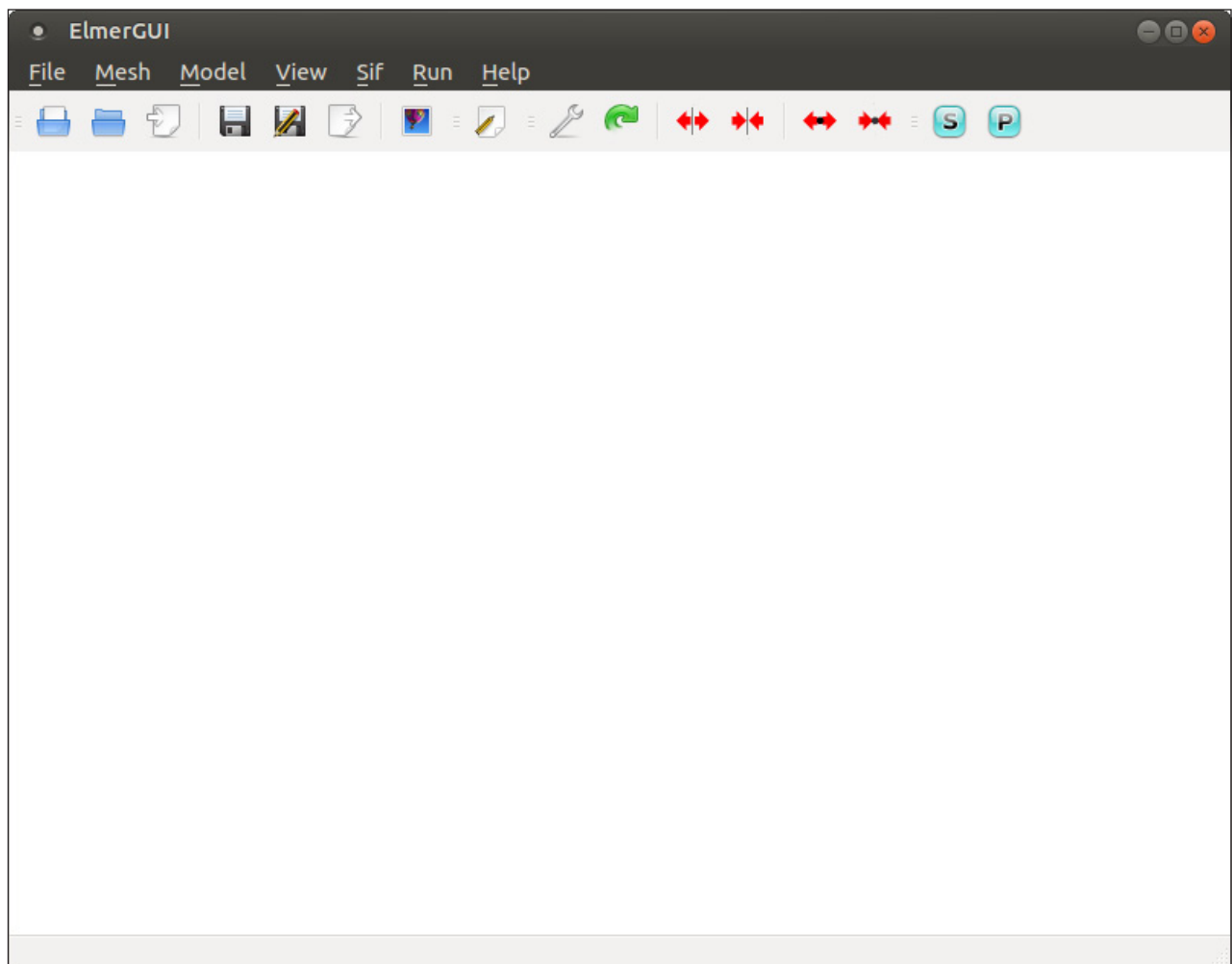


Figure 1. Executing ElmerGUI starts the main application, which controls all the other steps involved in using Elmer.

adding the Elmer repository to APT and installing it from there:

```
sudo apt-add-repository ppa:elmer-csc-ubuntu/elmer-csc-ppa
sudo apt-get update
sudo apt-get install elmerfem-csc
```

Those steps will conflict with the Elmer packages that exist in the main Debian repositories, so be sure that they haven't been previously installed on your system. The meta-package `elmerfem-csc` also will install a number of sample files that I use here as part of my description of Elmer's functionality.

The first step is to start Elmer. Depending on your desktop environment, there may be an entry within the menu system. If there isn't, open a terminal and start it with the command `ElmerGUI`. This command opens

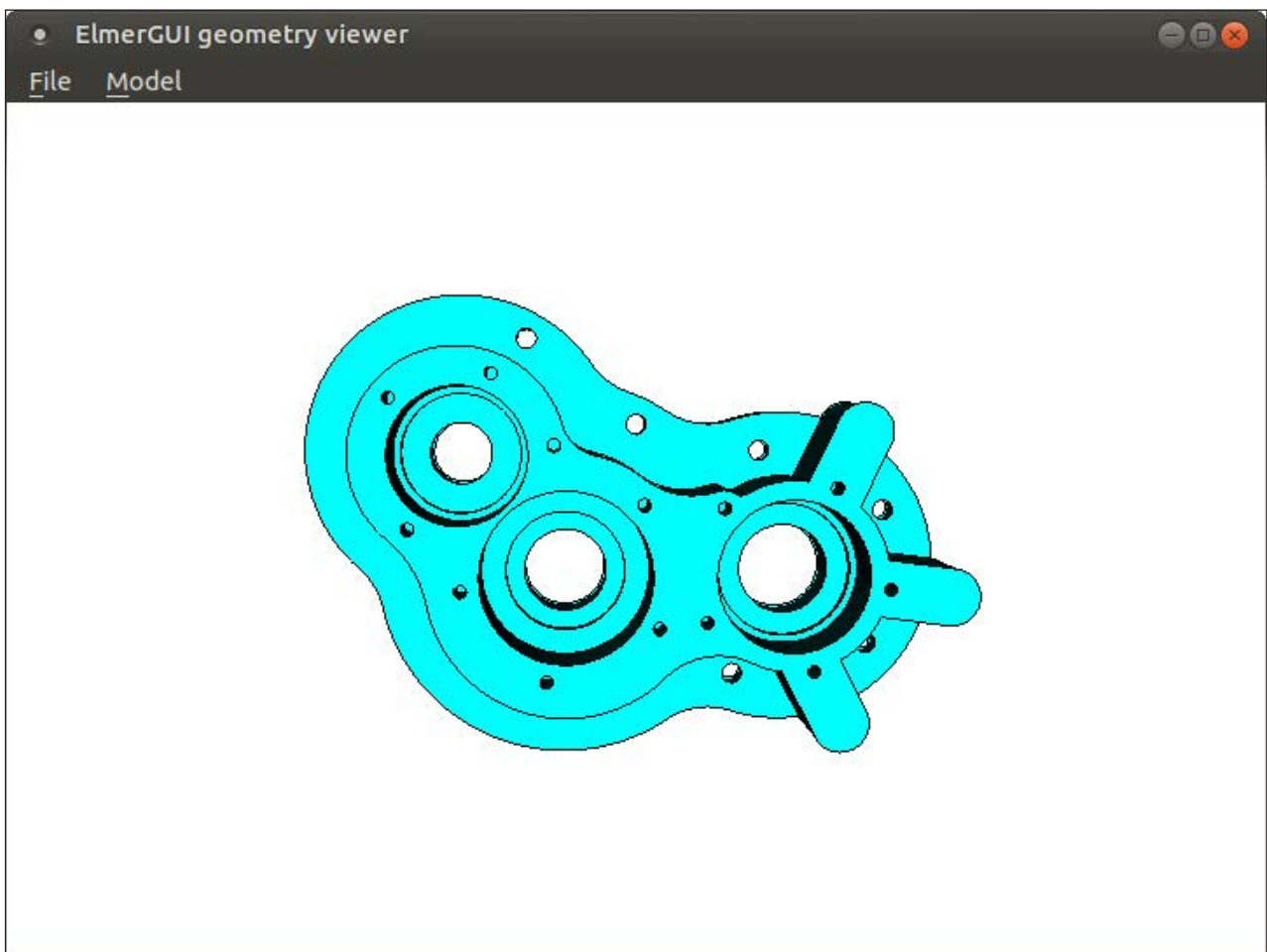


Figure 2. The ElmerGUI geometry viewer allows you to see the object you'll be using during your calculations.

the application and leaves you with an empty workspace.

Most programs that solve physics problems of this type follow three broad steps. The first, or pre-processing, step involves preparing the problem for solution. This includes defining any materials and their properties, along with any equations that describe the processes that will be taking place. This is also when you would apply a meshing function to break the geometry down into the subsections that will be used during the actual calculation phase. The second, or solver, step is when the input data is supplied to the actual solver functions that apply the equations to the materials described in your problem. The last, or post-processing, step is where you find the solution to your problem. Humans interpret graphical information most easily, so there are tools available in the post-processing step to visualize the final solution and help you see the results of your problem.

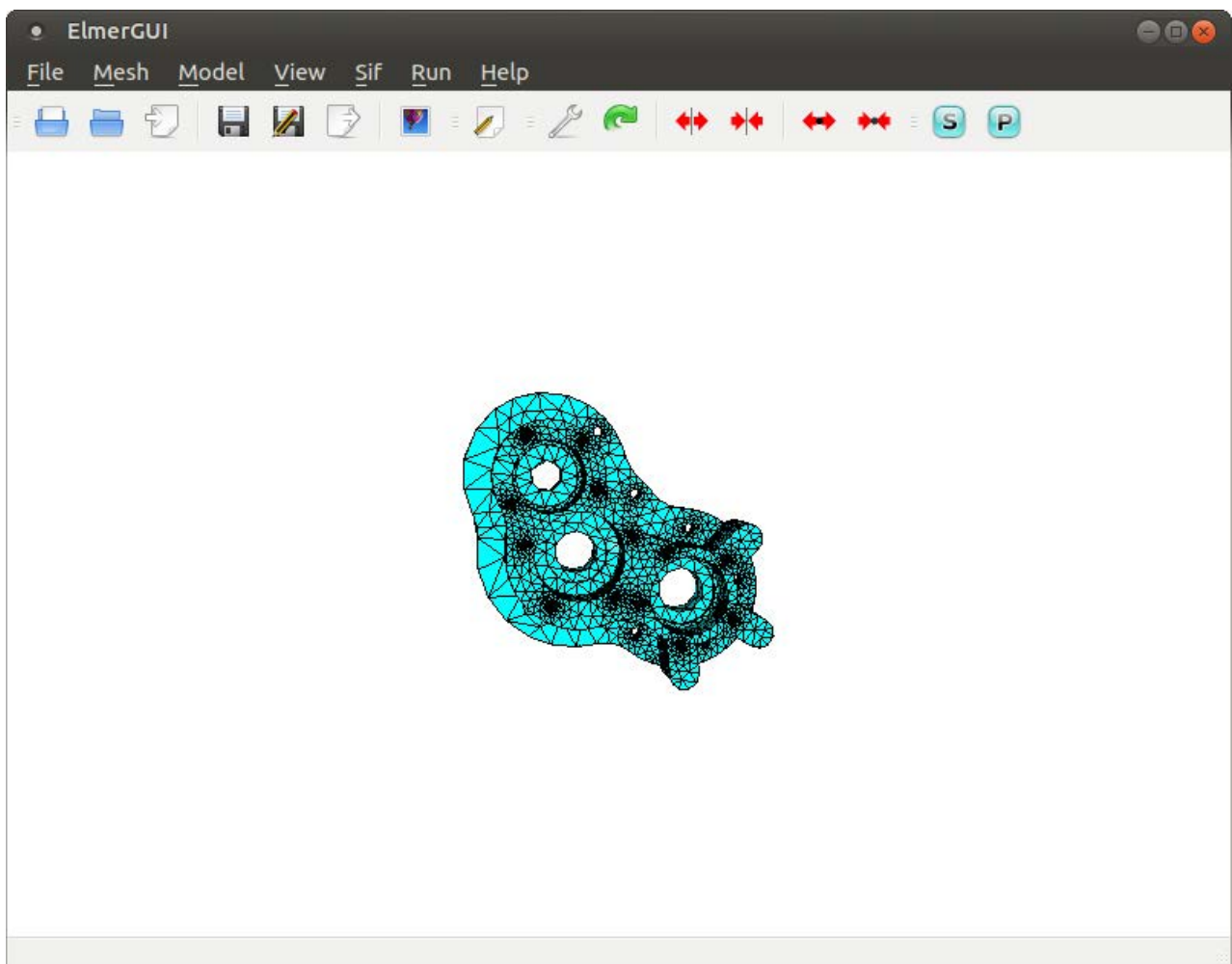


Figure 3. ElmerGUI also can handle meshing your objects in the pre-processing step.

Since Elmer's installation includes a number of sample files, let's go ahead and use those to explore what kind of work is possible with Elmer. These files should all be available in the `/usr/share/ElmerGUI/samples` directory, subdivided by the types of files contained within the samples.

Let's look at the heat distribution across a pump section and see how the distribution happens. To start, you will want to open the `pump_carter_sup.stp` step file, which is located in the `step` subdirectory of the `samples` directory. When you open this file, it gets loaded into the ElmerGUI geometry viewer.

This viewer allows you to grab the object with your mouse cursor and rotate it around, so you can get a good look at it and make sure it's structured properly. This object also is meshed automatically by ElmerGUI, so that it can be used in the solver stage.

Here, you can see the set of triangles that have been mapped onto your

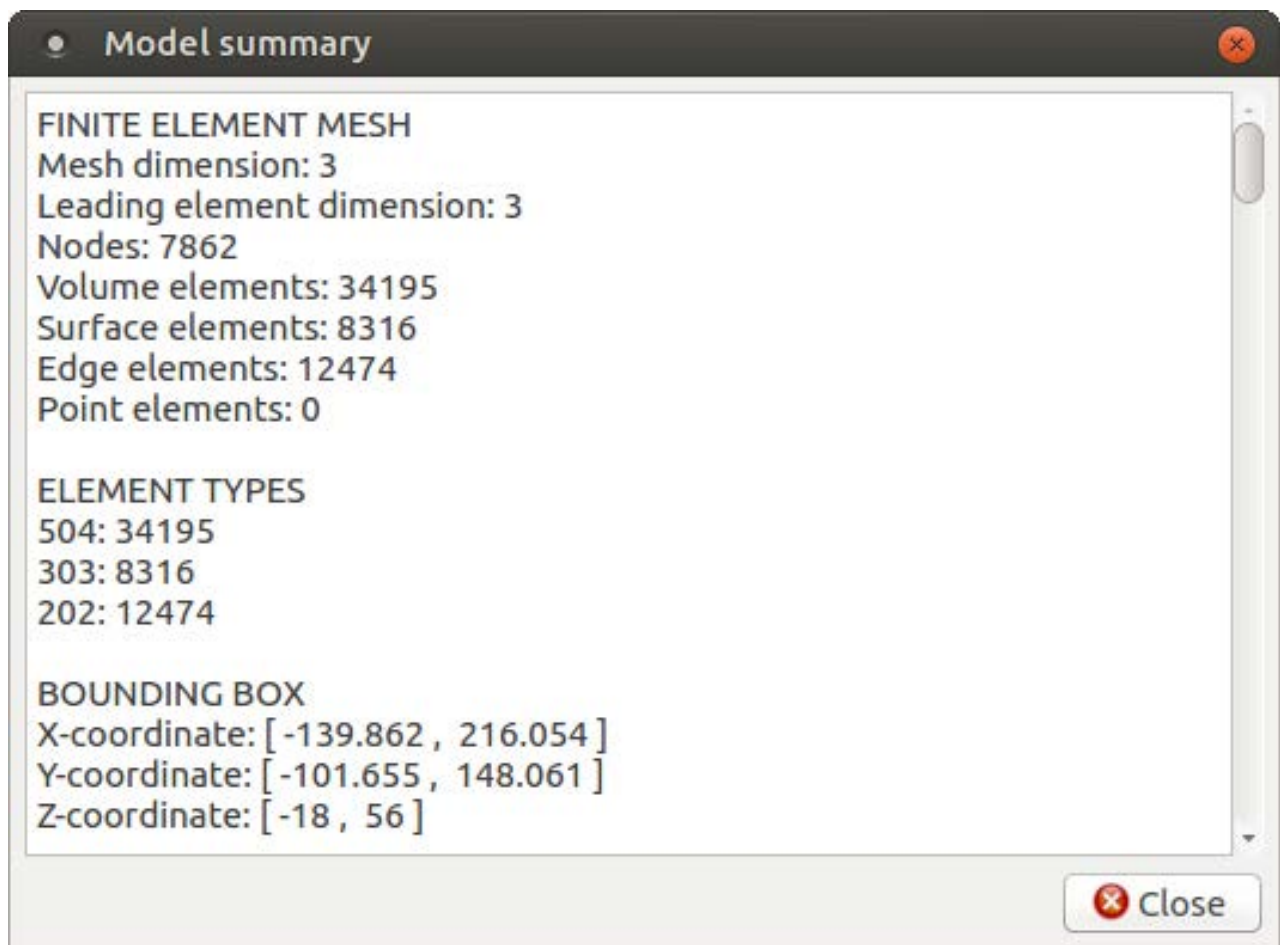


Figure 4. The model summary tells you things like the number of nodes, edges and volume elements within your model.

object to define the domains on which the solver should operate. You can verify the actual meshing by clicking the Model→Summary menu item, and look at the top of the output for the summary information.

In this case, you can see that the meshing created 7,862 nodes and 34,195 volume elements. If this isn't fine enough to handle the accuracy you need, you can click the View→Cad model menu item to bring up the geometry viewer again and then click Model→Preferences. If you do make any changes, don't forget to click Mesh→Remesh to redo the meshing process. This may take some time if you are adding a significant number of subsections.

Now that the object is loaded, let's start setting some initial conditions. For this example, you want to set the temperature on the inside surfaces of the three holes. You can select surfaces by double-clicking on them. If you need to select multiple surfaces, simply hold the Ctrl key down at the same time.

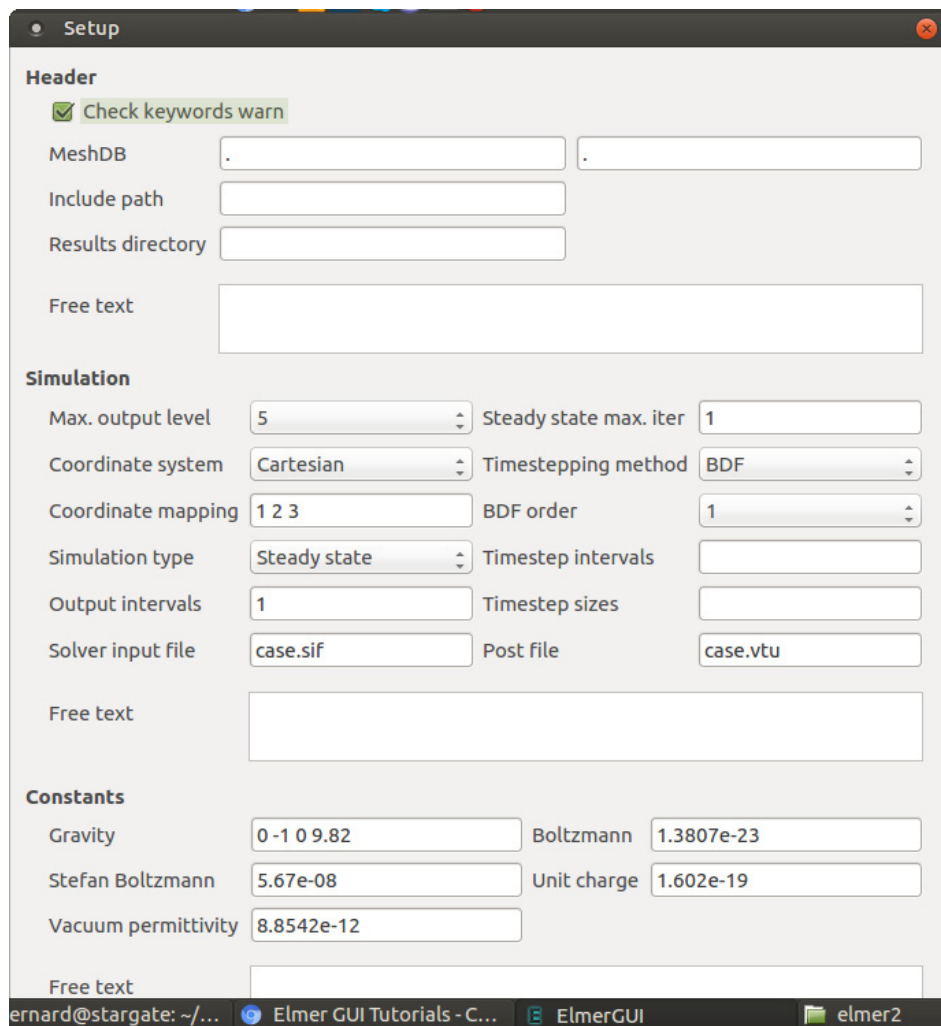


Figure 5.
The model setup window allows you to set initial conditions for the entire model as a whole.

Since you want to treat these three surfaces as a single unit, you'll need to unify them. You can do this by clicking the Mesh→Unify surface menu item. To set initial conditions for the entire problem, click the Model→Setup menu item.

Here you can set input and output locations, constants, numbers of iterations or time step procedures, among many other options. You also need to add the equations that will define the physical processes you actually are trying to solve for.

Clicking Model→Equation→Add opens an editing window to create a new equation.

In this case, let's just add a heat equation to calculate how heat flows through this pump part. Since you need to apply this to the entire object, you can click the apply box for Body 1 here in the equation editor. But, what kind of object is this? Clicking Model→Material pops up a window

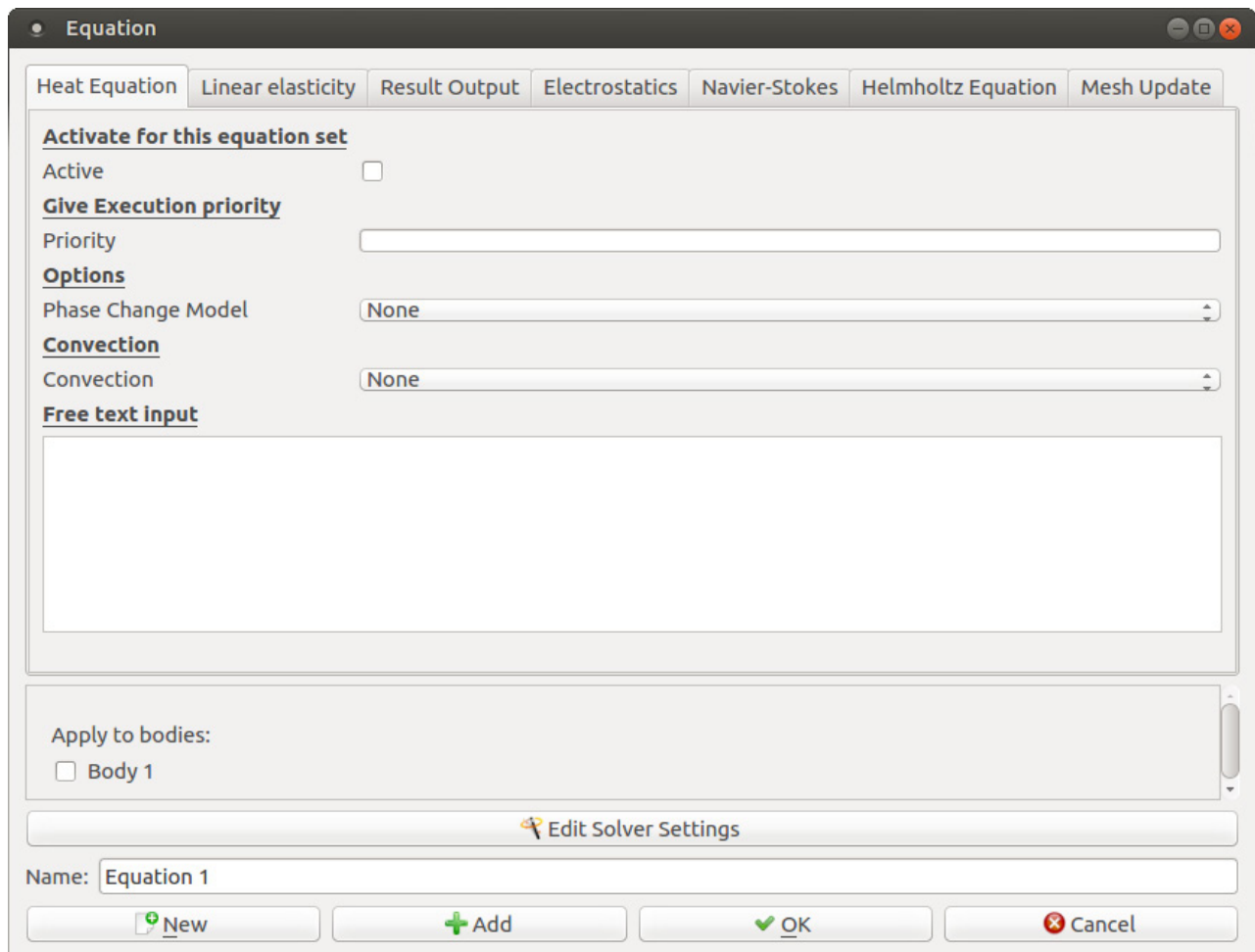


Figure 6. You can add equations to define the physical processes you are trying to model.

where you can define all the physical properties of the object.

There also is a button called Material library that allows you to select from a set of predefined materials. In this case, let's set the object to be aluminum. As with the equation editor, you can apply the material type to the object directly here.

Clicking Model→Body Force pops up a window where you can enter values that would represent the right-hand side of the equations.

Clicking the Model→Boundary Condition menu item will pop up a new window where you can set boundary conditions for your equations.

In this case, say you want to create a new boundary temperature of 293°Kelvin, which is named RoomTemp. To apply this boundary condition, go back to the main ElmerGUI interface and double-click

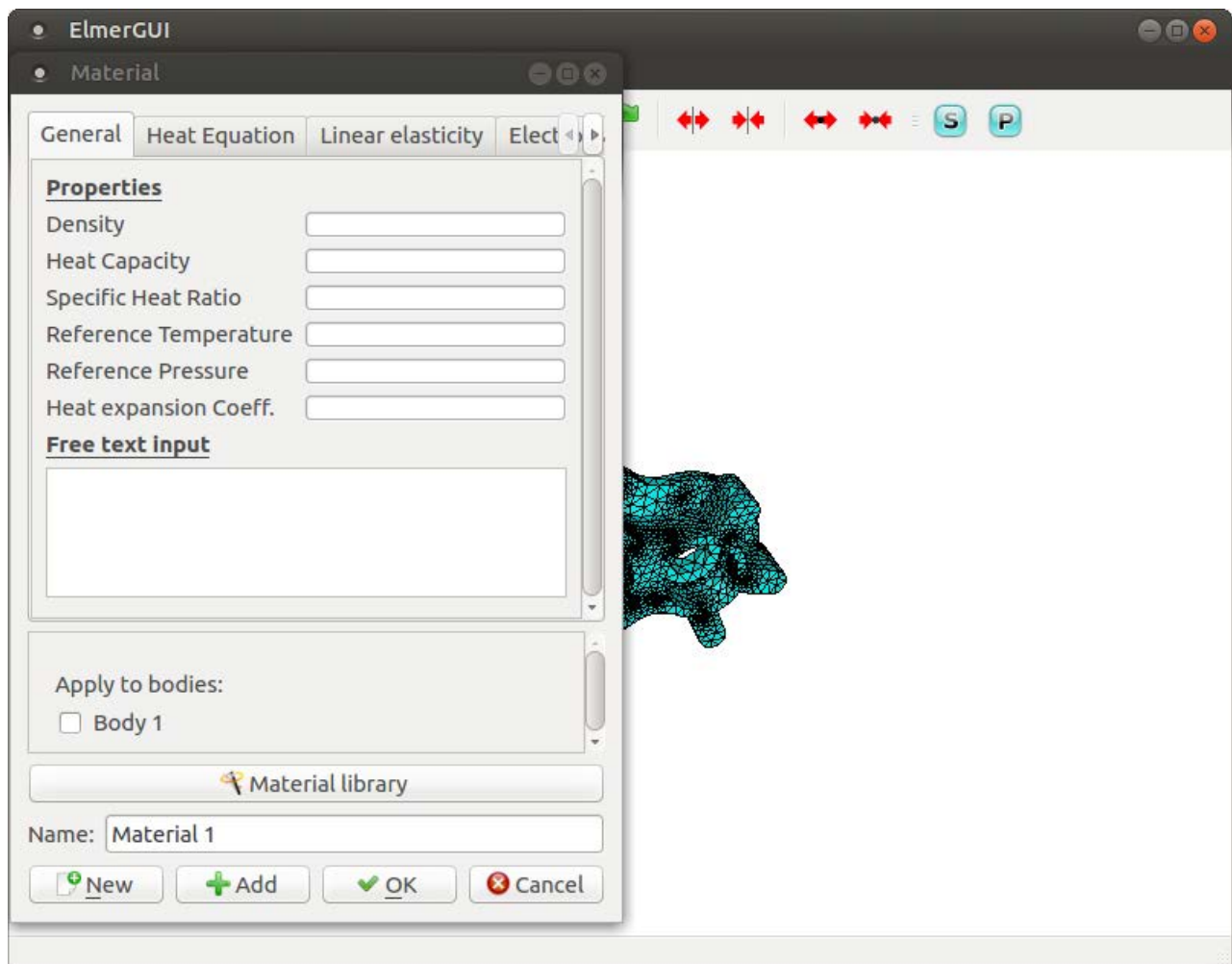


Figure 7. You can set all of the physical properties of the materials used within your object.

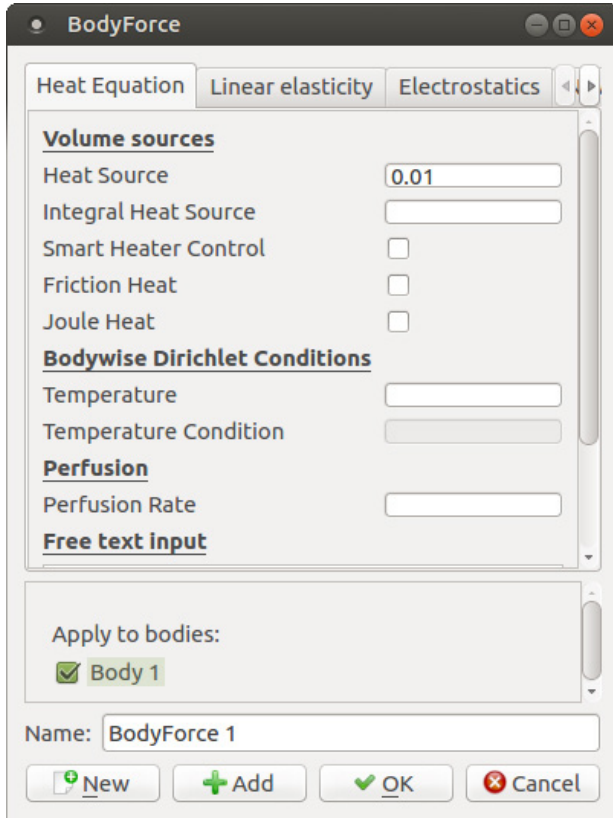


Figure 8. You can set body forces that define forcing values for the equations being used.

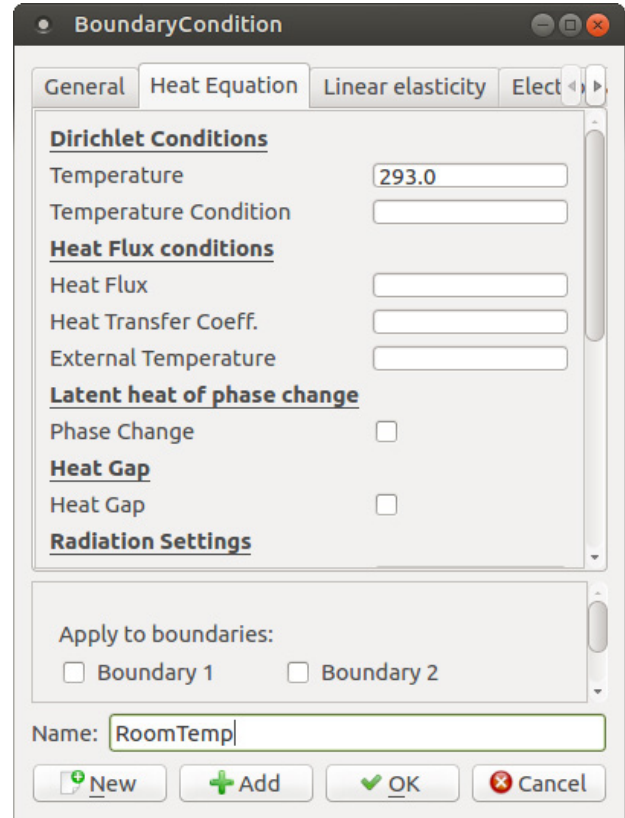


Figure 9. You can set boundary conditions for your equations to be used by the solver stage.

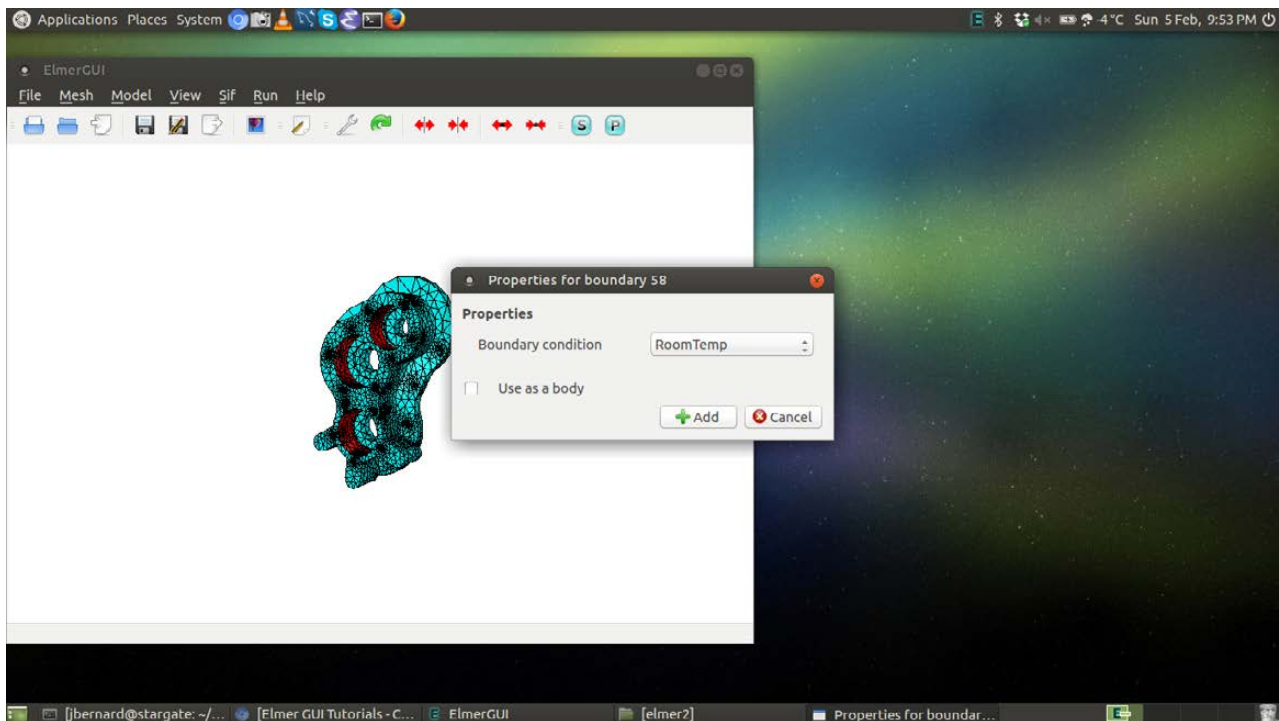
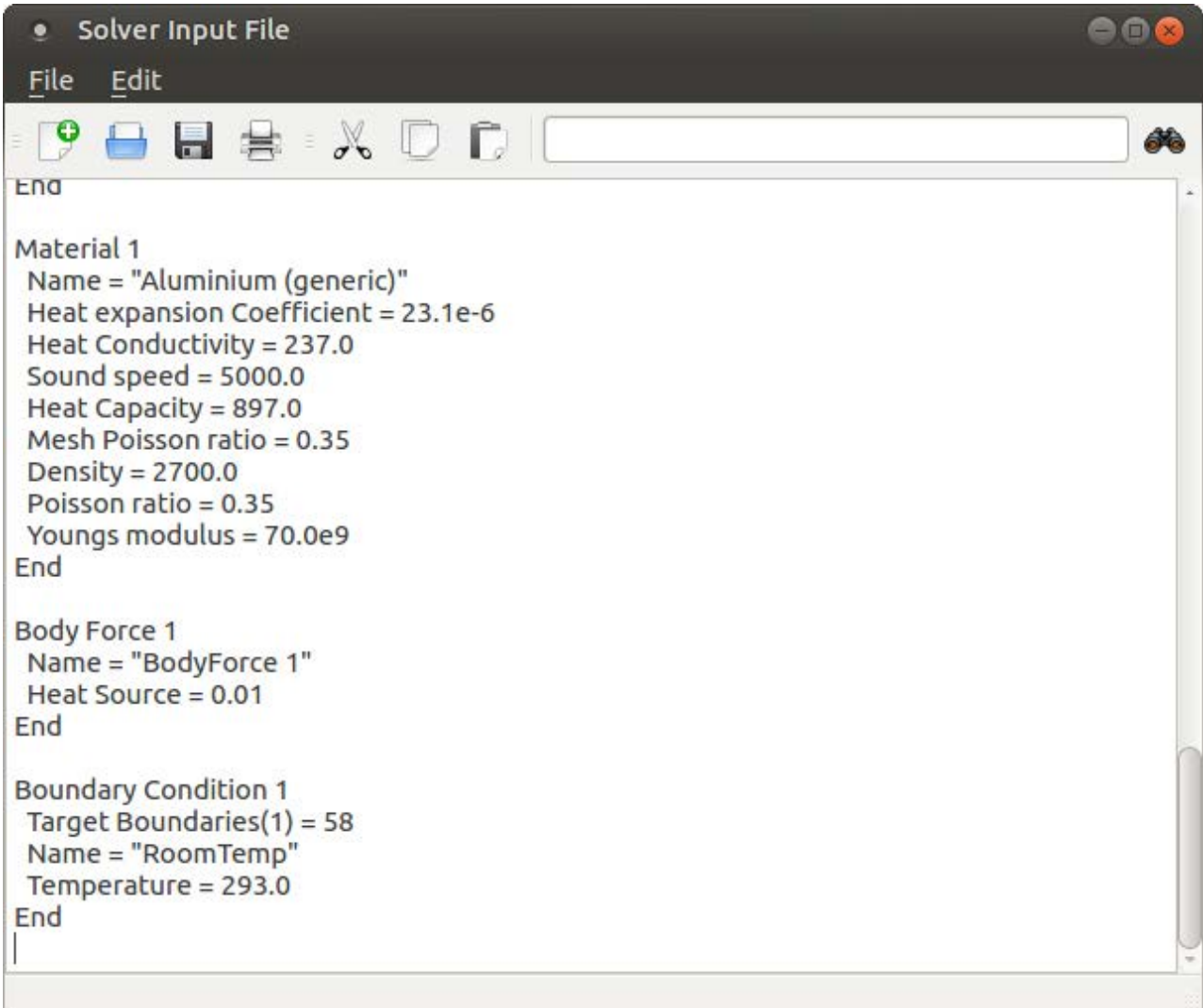


Figure 10. Select the sections that need to have a boundary condition applied to it.

the surface of the inside of the holes again. A small window will pop up asking you to set a boundary condition, where you can select the RoomTemp condition that was just set.

In order to run the solver, you need to generate the final input file. You can do this by clicking the Sif→Generate menu item. If you want to take a look at it, click Sif→Edit to pop up an editor window.

You also will need to save the entire project so that all of the required files are in a single location before starting the solver. Now, click Run→Start Solver to start the whole process. Once it's done, you'll see some new output windows. The top one is the convergence monitor, showing you how quickly the solver came to the final solution.



```
End

Material 1
Name = "Aluminium (generic)"
Heat expansion Coefficient = 23.1e-6
Heat Conductivity = 237.0
Sound speed = 5000.0
Heat Capacity = 897.0
Mesh Poisson ratio = 0.35
Density = 2700.0
Poisson ratio = 0.35
Youngs modulus = 70.0e9
End

Body Force 1
Name = "BodyForce 1"
Heat Source = 0.01
End

Boundary Condition 1
Target Boundaries(1) = 58
Name = "RoomTemp"
Temperature = 293.0
End
```

Figure 11. You can open an editor to manipulate the solver input file directly.

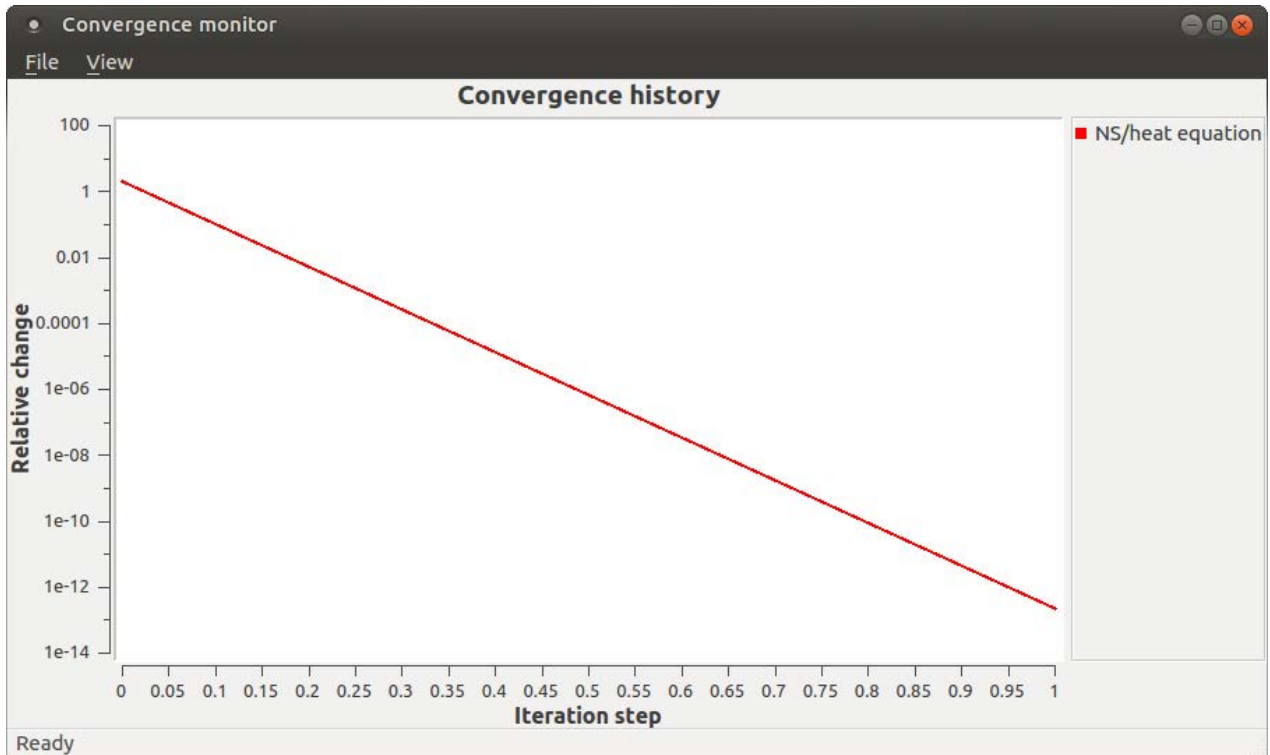


Figure 12. After the solver is done, you'll see a window showing how quickly convergence happened.

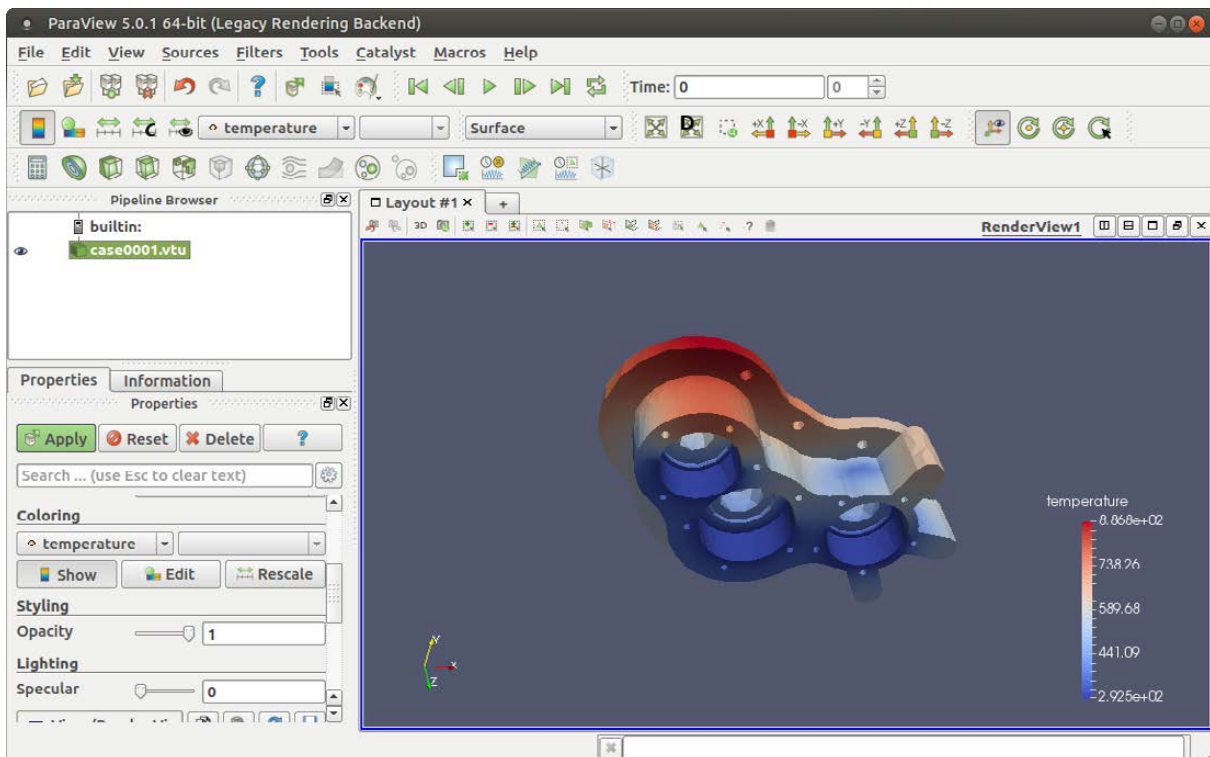


Figure 13. Paraview can provide sophisticated visualization tools for your output files.

You can visualize the results in the post-processing step. In this example, click the Run→Start ElmerVTK menu item to get one type of visualization. If you also installed paraview, you can get a very sophisticated visualization application to look at your output files.

This article was just a walk-through of one of the sample problems to show a small cross-section of the available functionality. I hope it has whetted your appetite enough to look at the other sample problems and use them as a jumping-off point for your own research interests.—Joey Bernard

THEY SAID IT

Creativity is allowing yourself to make mistakes. Art is knowing which ones to keep.

—Scott Adams

Words, once they are printed, have a life of their own.

—Carol Burnett

I don't hire people who have to be told to be nice. I hire nice people.

—Leona Helmsley

We make a living by what we get, we make a life by what we give.

—Winston Churchill

I have learned that to be with those I like is enough.

—Walt Whitman

[RETURN TO CONTENTS](#)



14th Annual 2017 HPC FOR WALL STREET – CLOUD & DATA CENTERS Show & Conference

APRIL 3, 2017 (Monday) ROOSEVELT HOTEL, NYC
Madison Ave & 45th, next to Grand Central Station

Register Today.
Low-Cost Conference at \$295.
Save \$100.
Register for Free Show

The Next Big Game Changer. New FinTech Technology Featured on April 3. All-Star Conference program for 2017.

Plan to attend the largest FinTech meeting of Cloud, Data Centers, HPC, Big Data, Networks, AI and Machine Learning, Trading, Low Latency for the Capital Markets.

- The Next-Generation FinTech Economy
- HPC Innovation for the FSI Market
- Addressing the Scalability Challenges of Deep Learning Model Training
- Optimizing New Data Technology and Developing HPC as a Service
- The AI Revolution Comes to Wall Street
- New Applications for High Performance Networking
- Enabling Financial Institutions to Build and Deploy Artificial Intelligence Applications using an Open Source Container Platform
- A New Approach to Data Storage and new Data Storage Architecture
- Hybrid Cloud, Private Cloud, Public Cloud
- Innovative Servers, Networks and Storage Applications

Plan to attend low-cost conference at \$295. Save \$100.

Qualified End Users may register as our guests, but must confirm their title and responsibility.

Register for the free show online at: www.flaggmgmt.com/linux

2017 Sponsors



Show Hours: Mon, Apr 3 8:00 - 4:00
Conference Hours: 8:30 - 4:50

Visit: www.flaggmgmt.com/linux

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

Partial List of 2017 Speakers.



Terry Roche, Principal,
Head of FinTech
Research, TABB Group



Donal Byrne
CEO,
Corvil



Lacee McGee
Sr Prod Mgr, FSI
Vertical Sols, HPE



Harvey Stein
Head of Credit Risk,
Modeling, Bloomberg



Dino Vitale
TD Securities



Moiz Kohari (Invited)
SVP Ch Tech Arch,
State Street London



Anthony Golia
FSI Chief Architect,
Red Hat



David Rukshin
CTO,
WorldQuant (invited)



Asaf Wachtel
Sr Dir, Business Dev,
Mellanox Technologies



Roman Chwył
Head of Fin Svcs
Google



Joseph George
VP Solutions Strategy,
SUSE



Andy Steinbach
Sr Dir Global FSI
Bus Dev, NVIDIA



Phil Filleul
Segment Dir, Fin
Svcs, Cray Inc



Felix Candelario
Global Fin Svcs Sol Arch,
Amazon Web Svcs (Invited)



Asif Alam
Global Bus Dir
Thomson Reuters



Natalia Vassilieva
Sr Research Manager,
Hewlett Packard Labs



Ed Turkel
HPC Strategist,
Dell EMC



Don Clegg
VP Mktg Bus Dev
Supermicro (Invited)



Pat McGinn
BBA/IB CITP VP Prod Mktg
CoolIT Systems



PREVIOUS
UpFront

NEXT

Reuven M. Lerner's
At the Forge



Android Candy: If Not This Then Stringify



I love IFTTT (If This Then That: <http://ifttt.com>), but although it usually works well, it's more and more common for triggers to fail. Sometimes they don't fail, but take several minutes to activate. When you want a light to turn on as you enter a room, several minutes of delay clearly can be a deal-breaker. I'm not sure if the problem is capacity issues or individual API problems, but I no longer feel confident that IFTTT will fire reliably. Although it's still in beta, Stringify aims to be more reliable, but also more robust.

With IFTTT, simplicity is king. You have a single trigger (If...) and a single action (then that...). With Stringify, you have "flows", which allow multiple results along with conditionals. Basically, you write a logical flow of triggers and results using the same sorts of triggers and results IFTTT offers. Granted, the number of connected services is significantly smaller, but the number is growing all the time. There's also no web-based interface for the Stringify building process, which bums me out. I would rather have a big screen to build flows, but it's only possible to manipulate your account on a mobile device (Android or iOS).

It's not clear whether Stringify will end up being more popular

and/or more reliable than IFTTT. It certainly has the promise to surpass the usefulness of IFTTT though. At the very least, the idea of competition likely will make both services strive for excellence. (Perhaps IFTTT will need to add another "T" and become "If This Then These Things!")

Thanks to using the idea of building "awesomer" ideas on top of already awesome ideas, Stringify gets this month's Editors' Choice award. Give it a try, and see if you can push the service hard enough to stress its reliability. I'm hoping it proves to be a viable alternative to the already awesome IFTTT!

—Shawn Powers

Flows Connect Your Things To Create Automation

Automatically jumpstart your morning with your Alexa, Hue and Nest

(Screenshot from Google Play Store)

[RETURN TO CONTENTS](#)

Classifying Text

How can you categorize documents using machine learning? It's simpler than you might think.



REUVEN M. LERNER

Reuven M. Lerner, a longtime Web developer, offers training and consulting services in Python, Git, PostgreSQL and data science. He has written two programming ebooks (*Practice Makes Python* and *Practice Makes Regexp*) and publishes a free weekly newsletter for programmers, at <http://lerner.co.il/> newsletter. Reuven tweets at @reuvenmlerner and lives in Modi'in, Israel, with his wife and three children.



PREVIOUS
Editors' Choice

NEXT

Dave Taylor's
Work the Shell



IN MY LAST FEW ARTICLES, I've looked at several ways one can apply machine learning, both supervised and unsupervised. This time, I want to bring your attention to a surprisingly simple—but powerful and widespread—use of machine learning, namely document classification.

You almost certainly have seen this technique used in day-to-day life. Actually, you might not have seen it in action, but you certainly have benefited from it, in the form of an email spam filter. You might remember that back in the earliest days of spam filters, you needed to “train” your email program, so that it would know what your real email looked like. Well, that was a machine-learning model in action, being told what “good” documents looked like, as opposed to “bad” documents. Of course, spam filters are far

more sophisticated than that nowadays, but as you'll see over the course of this article, there are logical reasons why spammers include innocent-seeming (and irrelevant to their business) words in the text of their spam.

Text classification is a problem many businesses and organizations have to deal with. Whether it's classifying legal documents, medical records or tweets, machine learning can help you look through lots of text, separating it into different groups.

Now, text classification requires a bit more sophistication than working with purely numeric data. In particular, it requires that you spend some time collecting and organizing data into a format that a model can handle. Fortunately, Python's scikit-learn comes with a number of tools that can get you there fairly easily.

Organizing the Data

Many cases of text classification are supervised learning problems—that is, you'll train the model, give it inputs (for example, text documents) and the "right" output for each input (for example, categories). In scikit-learn, the general template for supervised learning is:

```
model = CLASS()  
model.fit(X, y)  
model.predict(new_data_X)
```

`CLASS` is one of the 30 or so Python classes that come with scikit-learn, each of which implements a different type of "estimator"—a machine-learning algorithm. Some estimators work best with supervised classification problems, some work with supervised regression problems, and still others work with clustering (that is, unsupervised classification) problems. You often will be able to choose from among several different estimators, but the general format remains the same.

Once you have created an instance of your estimator, you then have to train it. That's done using the "fit" method, to which you give `X` (the inputs, as a two-dimensional NumPy array or a Pandas data frame) and `y` (a one-dimensional NumPy array or a Pandas series). Once the model is trained, you then can invoke its "predict" method, passing it `new_data_X`, another two-dimensional NumPy array or Pandas data frame. The result

is a NumPy array, listing the (numeric) categories into which the inputs should be classified.

One of my favorite parts of using scikit-learn is the fact that so much of it uses the same API. You almost always will be using some combination of “fit” and “predict” on your model, no matter what kind of model you’re using.

As a general rule, machine-learning models require that inputs be numeric. So, you turn category names into numbers, country names into numbers, color names into numbers—basically, everything has to be a number.

How, then, can you deal with textual data? It’s true that bytes are numbers, but that won’t really help here; you want to deal with words and sentences, not with individual characters.

The answer is to turn documents into a DTM—a “document term matrix” in which the columns are the words that were used across the documents, and the rows indicate whether (and how many times) that word existed in the document.

For example, take the following three sentences:

- I’m hungry, and need to eat lunch.
- Call me, and we’ll go eat.
- Do you need to eat?

Let’s turn the above into a DTM:

	i'm	hungry	and	need	to	eat	lunch	call	me	we'll	go	do	you
1	1	1	1	1	1	1	1	0	0	0	0	0	0
0	0	0	1	0	0	1	0	1	1	1	1	0	0
0	0	0	0	1	1	1	0	0	0	0	0	1	1

Now, this DTM certainly does a good job of summarizing which words appeared in which documents. But with just three short sentences that I constructed to have overlapping vocabulary, it’s already starting to get fairly wide. Imagine what would happen if you were to categorize a large

number of documents; the DTM would be massive! Moreover, the DTM would mostly consist of zeros.

For this reason, a DTM usually is implemented as a “sparse matrix”, listing the coordinates of where the value is non-zero. That tends to crunch down its size and, thus, processing time, quite a lot.

It’s this DTM that you’ll feed into your model. Because it’s numeric, the model can handle it—and, thus, can make predictions. Note that you’ll actually need to make two different DTMs: one for training the model and another for handing it the text you want to categorize.

Creating a DTM

I decided to do a short experiment to see if I could create a machine-learning model that knows how to differentiate between Python and Ruby code. Not only do I have a fair amount of such code on my computer, but the languages have similar vocabularies, and I was wondering how accurately a model could actually do some categorization.

So, the first task was to create a Python list of text, with a parallel list of numeric categories. I did this using some list comprehensions, as follows:

```
from glob import glob

# read Ruby files
ruby_files = [open(filename).read()
              for filename in glob("Programs/*.rb")]

# read Python files
python_files = [open(filename).read()
                for filename in glob("Programs/*.py")]

# all input files
input_text = ruby_files + python_files

# set up categories
input_text_categories = [0] * len(ruby_files) + [1]
                       + [0] * len(python_files)
```

After this code is run, I have a list (`input_text`) of strings and another list (`input_text_categories`) of integers representing the two categories into which these strings should be classified.

Now I have to turn this list of strings into a DTM. Fortunately, scikit-learn comes with a number of “feature extraction” tools to make this easy:

```
from sklearn.feature_extraction.text import CountVectorizer

cv = CountVectorizer()
cv_dtm = cv.fit_transform(input_text)
```

`CountVectorizer` isn’t the only way to create a DTM. Indeed, there are different strategies you can use. Among other things, the granularity of one word, rather than multiple words, might not be appropriate for your text.

Notice that I use `cv.fit_transform`. This both teaches the vectorizer the vocabulary (“fit”) and produces a DTM. I can create new DTMs with this same vocabulary using just “transform”—and I will indeed do this in a little bit, when I want to make a prediction or two.

Creating a Model

Now I have my inputs in a format that can be used to create a model! You potentially can use a number of algorithms, but one of the most common (and surprisingly accurate) is Naive Bayes. Scikit-learn actually comes with several different versions of Naive Bayes. The one that I use here is called `MultinomialNB`; it works well with this sort of textual data. (But, of course, it’s generally a good idea to test your models and even tweak the inputs and parameters to squeeze better results out of them.) Here’s how I create and then train my model:

```
from sklearn.naive_bayes import MultinomialNB
nb = MultinomialNB()
nb.fit(input_text_dtm, input_text_categories)
```

Notice that I’ve used “fit” twice now: once (on `CountVectorizer`) to

train and create a DTM from the input text and then (on `MultinomialNB`) to train the model based on that DTM.

The model is now all set! Now I can make some predictions. I'll create some new documents:

```
docs_new = ['class Foo(object):\nprint "Hello, {}".format(self.name)\n',
            'x = [10, 20, 30]\n',
            '10.times do {|i| puts i}']
```

The `docs_new` variable contains three strings: the first is in Python, the second could be either Ruby or Python, and the third is in Ruby.

To see how the model categorizes them, I'll first need to create a DTM from these documents. Note that I'm going to reuse `cv`, the `CountVectorizer` object. However, I'm not going to use the "fit" method to train it with a new vocabulary. Rather, I'm going to use "transform" to use the existing vocabulary with the new documents. This will allow the model to compare the documents with the previous ones:

```
docs_new_dtm = cv.transform(docs_new)
```

Now to make a prediction:

```
nb.predict(docs_new_dtm)
```

The output is:

```
array([1, 1, 0])
```

In other words, the first two documents are seen as Python, and the third is seen as Ruby—not bad, for such a small training set. As you can imagine, the more documents with which you train, the more accurate your categorization is likely to be.

I tried a slight variation on the above code with the "20 newsgroups" data set, using 20,000 postings from 20 different Usenet forum postings. After using `CountVectorizer` and `MultinomialNB` just as I did here, the model was able to predict, with a surprisingly high degree of accuracy, the

most appropriate newsgroup for a variety of sentences and paragraphs.

Of course, as with everything statistical—including machine learning—the success rate never will be 100%. And indeed, you can (and probably will want to) update the model, tuning the inputs and the model's hyperparameters to try to improve it even more.

Summary

Document categorization is a practical application of machine learning that a large number of organizations use—not just in spam filters, but also for sorting through large volumes of text. As you can see, setting up such a model isn't especially difficult, and scikit-learn provides a large number of vectorizers, feature extraction tools and estimators that you can use to create them. ■

RESOURCES

I used Python (<http://python.org>) and the many parts of the SciPy stack (NumPy, SciPy, Pandas, Matplotlib and scikit-learn) in this article. All are available from PyPI (<http://PyPI.python.org>) or from ScyPy.org (<http://scipy.org>).

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

[RETURN TO CONTENTS](#)



BALTIMORE

— DRUPALCON 2017 —

APRIL 24-28

Watermarking Images—from the Command Line

It's pretty darn easy to analyze and manipulate images from within a shell script.



DAVE TAYLOR

Dave Taylor has been hacking shell scripts on UNIX and Linux systems for a really long time. He's the author of *Learning Unix for Mac OS X* and *Wicked Cool Shell Scripts*. You can find him on Twitter as @DaveTaylor, or reach him through his tech Q&A site: <http://www.AskDaveTaylor.com>.

PREVIOUS

◀ Reuven M. Lerner's
At the Forge

NEXT

Kyle Rankin's
Hack and / ▶

US GEEKS MOSTLY THINK OF THE COMMAND LINE AS THE BEST PLACE FOR TEXT MANIPULATION. It's a natural with `cat`, `grep` and shell scripts. But although you can't necessarily view your results from within a typical terminal window, it turns out to be pretty darn easy to analyze and manipulate images from within a shell script.

In my last article, I introduced the splendid open-source ImageMagick suite that offers more features and functionality than you can shake a tree's worth of branches at. Why you would be shaking a tree at a piece of software escapes me, but, hey, I'm just

writing this stuff, not thinking about what I'm saying.

Um...wait a sec.

Anyway, ImageMagick includes a variety of programs that let you analyze, manipulate and even convert image files in a remarkable number of different ways.

My last article described setting things up and a few easy ways to confirm that the suite was working correctly on your computer. Now let's start with a simple task that can be useful for web servers: a script that checks image files and flags any that are more than a specific size.

In fact, let's use 8MB because that's the maximum size allowed in Facebook Open Graph, a fact of which many webmasters are already well aware.

Finding Those Big Darn Image Files

Identifying big files can be done with a simple `find`, but the goal here is to do something more sophisticated, so let's pair it with the ImageMagick command `identify`.

Here's a loop to identify files bigger than a specified size:

```
for name in `find *{png,jpg} -size +8M -print`  
do  
    echo file $name is bigger than 8MB  
done
```

That's a good start, but for those image files that match, more detail would be helpful, and I'm not talking `ls -l` output! Instead, let's replace the rudimentary `echo` statement with something more advanced:

```
dimensions=$(identify $name | cut -d\ -f3)  
size=$(identify $name | cut -d\ -f7)  
echo "File $name ($dimensions) has file size $size"
```

Now let's have a quick test:

```
$ sh bigimages.sh  
File screenshot-6.png (1800x490) has file size 9.639MB  
$
```

WORK THE SHELL

It's definitely more informative than just doing an `ls -l` and particularly so if you were to put this in an automatic email report for hosting clients!

Why? Because lots of people who aren't tech-savvy upload images directly from digital cameras—images that can be enormous. They resize the displayed image in their blog posts, but the original files are way larger than necessary, slowing down their sites unnecessarily.

Before moving on, let me make a few comments. First, notice the size test with `find: -size +8M`. `find` has a weird view of numbers and comparisons, and the test `-size 8M` would match only files that are exactly 8MB in size—not useful. Without the “M” suffix, the test would be comparing to 512-byte blocks, so `+8` would match a lot of files (because eight 512-byte blocks is only 4K, and that's tiny for an image file). `find` also knows “K” for kilobytes, “G” for gigabytes, “T” for terabytes and, yes, “P” for petabytes.

The second observation to make is the use of quotes with the `echo` statement. Try it. Without quotes, the shell would complain about the use of parentheses, and with single quotes, it would show the variable names, not expand them. It's a good real-world reminder of the subtle, but important quote nuances in the shell!

Add a Watermark

One of the most popular uses of ImageMagick isn't to identify image dimensions but to add watermarks. You've seen watermarks on images all over the web. Sometimes it's a little copyright notice, a URL or even a tiny graphical bug that identifies the source or ownership of the image.

Nice effect, but how do you do it? Use the `convert` command. In fact, if you want to just add a text overlay on the bottom of an image, the command is simple:

```
convert $source label:'LinuxJournal.com'-append $output
```

The default isn't very glamorous, however, so you'll inevitably want to customize it a bit. To make the font a bit larger, use `-pointsize`, and to move the watermark text to be in the lower right instead of its default position, use `-gravity`.

WORK THE SHELL

This is a bit more sophisticated and shows some of the weirdness of ImageMagick:

```
convert $source -pointsize 80 -gravity east \  
    label:'LinuxJournal.com' -append $output
```

This easily can be poured into a script, of course, and either you can have the output images in a different directory or you can rewrite the source filename appropriately. My favorite way to accomplish the latter is this:

```
predot=$(echo $name | rev | cut -d. -f2- | rev)  
postdot=$(echo $name | rev | cut -d. -f1 | rev)  
newname=$(echo ${predot}-wm.$postdot)
```

Since there's no "last field" option in `cut`, the way to grab just the filename suffix, even if the base filename contains dots, is to reverse the name, grab the *first* field, then reverse it again. This way, you can take a filename like "red.rose.png" and rewrite it as "red.rose-wm.png" to denote the version that has the watermark added.

But, what if you have a small graphic bug you want to overlay on the lower left corner of the image instead?

Let's assume the watermark graphic is specified with the variable `$copy`. With that in mind, a nice blend of the two images can be achieved with a 100% dissolve:

```
composite -dissolve 100 $copy $source $output
```

That will put the graphic on the top left corner of the resultant image, "above" the underlying photograph or graphic.

You can move it to the lower right by using `-gravity`—literally. Imagine a compass overlaid on your image: "north" is up, "east" is the right side, "west" is the left side and so on. With the `convert -label` command shown earlier, the default position was the bottom of the resultant image, so gravity of "east" moved the label to the lower right.

With `composite`, however, there's a lot more flexibility, so positioning

the copyright bug in the lower right involves using a gravity of “southeast”—like this:

```
composite -dissolve 100 -gravity southeast $copy $source $output
```

Wrap that in a for loop, and you’ve got a handy script that can add text watermarks along the bottom or overlay a graphic watermark or copyright instead.

In both cases, notice that the ImageMagick commands always create a new output file. If you want to *replace* an image with a version that includes a watermark, you’d need to do a bit of file shuffling:

```
composite -dissolve 100 $copy $source $output  
mv $output $source
```

Ideally though, you’d check to ensure that the `composite` command worked properly (rather than potentially deleting files or producing a cryptic error). Two ways come to mind: check the error status of the `composite` command (the sequence `$?`) or just test for the existence of `$output`. Here’s the latter, since then you aren’t reliant on `composite` having accurate return codes:

```
composite -dissolve 100 $copy $source $output  
if [ -x $output ] ; then  
    mv $output $source  
else  
    echo "Couldn't replace $source, $output wasn't created?"  
fi
```

Is that good? Maybe not. The `-x` test checks for the existence of the file, but what would be better would be to ensure that the file exists and is non-zero size. That’s the `-s` flag. It’s a simple switch, and you’ve got the basis for a good script.

Magic with Images

The ImageMagick suite contains a number of commands that have dozens

WORK THE SHELL

to hundreds of parameters. They can be very quirky, as the gravity setting shows—be glad I didn't delve into the geometry parameters—but fortunately, there are lots of online tutorials and help pages.

Remember, it all starts at <http://www.imagemagick.org>.■

Send comments or feedback via

<http://www.linuxjournal.com/contact>
or to ljeditor@linuxjournal.com.

[RETURN TO CONTENTS](#)

LINUX[™]
JOURNAL



LINUX JOURNAL
ARCHIVES

Issues 1–272

The First 23 Years of Linux Journal (1994–2016)

Archive
1994–2016

NOW
AVAILABLE!

SAVE \$10.00
by using
discount code
2016ARCH
at checkout.

Coupon code
expires 4/28/2017

www.linuxjournal.com/archive

Sysadmin 101: Ticketing

Learn why tickets aren't busywork, but rather an important part of a professional system administrator's toolkit.



KYLE RANKIN

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

PREVIOUS

◀ Dave Taylor's
Work the Shell

NEXT

Shawn Powers'
The Open-Source
Classroom ▶

THIS IS THE THIRD IN A SERIES OF ARTICLES ON SYSTEM ADMINISTRATOR FUNDAMENTALS where I focus on some lessons I've learned through the years that might be obvious to longtime sysadmins, but news to someone just coming into this position.

In the first article, I discussed how to approach alerting and on-call rotations as a sysadmin. The second article covered how to automate yourself out of a job. In this article, I explore something that on the surface may seem boring or mundane but is absolutely critical to get right if you want to be an effective sysadmin: ticketing.

By ticketing, I'm referring to systems that allow sysadmins to keep track of tasks both internally and those requested by their coworkers or customers.

There are many ways to get ticketing wrong so that it becomes a drain on an organization, so many sysadmins avoid or it use it begrudgingly. Also, ticketing approaches that work well for developers may be horrible for sysadmins, and vice versa. If you don't currently use a ticketing system, I hope by the end of this article, I've changed your mind. If you do use tickets, but you wish you didn't, I hope I can share how to structure a ticketing system that makes everything easier, not more difficult.

Why Tickets Are Important

Like documentation, tickets are one of those important things in a mature organization that some administrators think are unnecessary or even a waste of time. A ticketing system is important no matter the size of your organization. In a large organization, you have a large volume of tasks you need to keep track of distributed among a group of people. In a small organization, you often have one person taking on many roles. This leads me to the first reason why tickets are important.

Tickets Ensure That Tasks Aren't Forgotten Sysadmins are asked to do new tasks constantly. These days, there are any number of ways a coworker might ask for your help, from an email, to a phone call, to a message in a chat program, to a tap on the shoulder. If you weren't doing anything else, you immediately could start working on that task, and everything would be fine. Of course, usually sysadmins have to balance needs from many different people at the same time. Even requests that come in through email have a tendency to fall through the cracks and be forgotten. By storing every request in a ticket, no matter how you got the request, it is captured, so that even if you do forget about it, you'll remember it the next time you look at your ticketing system.

Tickets Make Sure the Task Is Done Right Even if you can remember what someone wants you to do, you may not remember on Monday all of the details that someone told you in person on Friday. A ticket lets you capture exactly what people want done in their own words and provides a way for them to confirm that you completed the task the way they wanted before you close the ticket.

Tickets Help You Prioritize Tasks Every request is important to the person who makes it. Every request may not be as urgent to you or your team compared to your other tasks, however. When all of your

Every time you change a system, you create an opportunity for something to break.

tasks are captured in tickets, the team lead or manager can go through and re-prioritize tasks so they are worked on in the right order. This ends up being more fair for everyone; otherwise, new tasks have a way of cutting in line, especially when the person asking for something is standing over your shoulder.

With a ticketing system, team leads or managers have a full list of important tasks they can point to when they need to explain why they aren't dropping everything for a new request. At the very least, it will help direct the conversation about why a particular task *should* be put at the head of the line.

Tickets Distribute the Work If you have only one sysadmin, distributing tickets and projects is easy. Once your team grows though, it's important to distribute the work so no member of the team gets burned out. Coworkers have a tendency of finding that senior member of your team who is most productive and going to them directly when they have any issue. Of course, that team member is probably already working on plenty of other tasks or may be trying to focus on an important project.

When a task is captured in a ticket, the team lead or manager can assign and reassign tickets to different members of the team to make sure no one gets burned out, and also to ensure that everyone learns how to do things. Otherwise, you end up cultivating specialists within the team that always take tickets related to certain systems, which leads to problems later when that team member goes on vacation.

Tickets Provide an Audit Trail for Changes Every time you change a system, you create an opportunity for something to break. If you are lucky, things break immediately after you make the change. More often, you'll find that it takes some time for a change to cause a problem. You'll discover two weeks later that something stopped working, and with a ticketing system, you can pull up all of the tasks that were worked on around that time. This makes it much easier to pinpoint potential causes of a problem.

Tickets also provide an audit trail for tasks that require approval or proof of completion, like creating or revoking accounts, granting new privileges or patching software. When someone asks who said it was okay for Bob to get access to production and when it happened, you can answer the question. If you need to prove that you applied a security patch, you can point to command output that you capture and then store in the corresponding ticket.

Qualities of an Effective Ticketing System

Many different ticketing systems exist, and sometimes when you hear people complain about tickets, what they are really complaining about is a bad ticketing system. When choosing between ticketing systems, you should look for a few things.

Some systems that developers use to track code through the development process result in very complicated workflows. For a sysadmin though, the simpler the ticketing system the better. Because you already are asking a sysadmin to take time out of solving a problem to document it in a ticket, it helps if the ticketing process is fast and simple. I prefer very simple ticket workflows for sysadmins where there may be only a few states: open, assigned, in progress, resolved and closed. (I'll talk more about how I treat each of those states in the next section.)

The fewer required fields in a ticket, the better. If you want to add extra fields for tags or other information, that's fine, just don't make those fields mandatory. The goal here is to allow sysadmins to create tickets based on someone walking up and tapping them on the shoulder in less than a minute.

Ideally, the ticketing system would allow you some other way to generate tickets from a script, either from sending an email to a special address or via an exposed API. If it has an API that lets you change ticket state or add comments, all the better, as you potentially can integrate those into your other automation scripts. For instance, I've created a production deployment script that integrates with my ticketing system, so that it reads the manifest of packages it should install from the ticket itself and then outputs all of the results from the deployment as comments in the ticket. It's a great way to enforce a best practice of documenting each of your software releases, but it does it in a way that

makes it the path of least resistance.

Favor ticketing systems that allow you to create dependencies or other links between tickets. It's useful to know that task A depends on task B, and so you must complete task B first. These kinds of ticketing systems also make it easier to build a master ticket to track a project and then break that large project down into individual tickets that describe manageable tasks. These kinds of systems often show all of the subordinate tickets in the master ticket, so a quick glance at the master ticket can give you a clue about where you are in a project.

How to Manage and Organize Tickets

Each ticketing system has its own notion of ticket states, but in my opinion, you should, at a minimum, have the following:

- **Open:** a task that needs to be completed, but hasn't been assigned to anyone.
- **Assigned:** a task that's in a particular person's queue, but they haven't started work on it yet. Tasks in this state should be safe to reassign to someone else.
- **In progress:** a task that has been assigned to someone who is currently working on the task. You definitely should communicate with the assignee before you reassign tickets in this state.
- **Resolved:** the sysadmin believes the task has been completed and is waiting for confirmation from the person who filed the ticket before closing it.
- **Closed:** the task has been completed to everyone's satisfaction.

A well run ticketing system should provide the team with the answers to a few important questions. The first question is "What should I work on now?" To answer that question, each member of the team should be able to claim tickets, and team leads or managers should be able to assign tickets to individual members of the team. It's important

I approach ticket priority as a way for users to help inform the team about how important the ticket is to them, but not how urgent it is for the team.

for people to claim tickets and start work only after they are claimed; otherwise, it's easy (and common) for two members of the team to start working on the same task without realizing it. Then everyone on the team can start working on tickets in their personal queue, starting with the highest-priority tasks.

The next question a good ticketing system should answer is "What should I work on next?" Once sysadmins' personal queues are empty, they should be able to go to the collective queue and see a list of tasks ordered by priority. It should be clear what tasks they should put on their queue, and if there's any question about it, they can go to the team lead or manager for some clarity. Again, ticket priority helps inform everyone on the team about what's next—higher-priority tasks trump lower-priority ones, not necessarily because they are less important (a ticket is always important to the person who filed it), but because they are less urgent.

I approach ticket priority as a way for users to help inform the team about how important the ticket is to them, but not how urgent it is for the team. The fact is, there's no way every employee in the company can know all of the other important tasks the sysadmin has to perform for other people nor can they be expected to weigh the importance of their need against everyone else's needs.

A good manager should reserve the right to weigh the priority assigned to a ticket against the other tickets in the queue and change the priority up or down based on its urgency relative to the other tasks. It also may be the case where a task that was low urgency two weeks ago has become urgent now because of how long it was in the queue, so a good manager would be aware of this and bump the priority. If you are going to start the practice of changing ticket priorities though, be sure to inform everyone of your intentions and how you will determine the urgency of a ticket.

Another key to managing tickets is to make sure all of your requests are captured in the ticketing system. Sometimes a coworker can be guilty of trying to skip ahead in line by messaging you with a request or walking directly to your desk to ask you to do something. Even in those cases where you really are going to drop everything to work on their request, you should insist on capturing the request in a ticket so you can track the work. This isn't just so you can prioritize it based on other tasks or so you don't forget it, it's so in a week when some problem crops up based on this urgent change, you'll see this ticket along with other tasks completed that day and it will help you track down the cause.

Finally, as a manager, be careful to distribute work fairly among your team. Even if one member of the team happens to be an expert on a particular service, don't assign that person every task related to that service; it's important for everyone on the team to cross-train. Pay attention if employees try to get tickets assigned to their favorite member of the team, and don't be afraid to reassign tasks to spread the work around evenly. Finally, every ticket queue has routine, mundane grunt work that must be done. Be sure to distribute those tasks throughout the team so no one gets burnt out. ■

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

[RETURN TO CONTENTS](#)

The Best SharePoint 2016 and Office 365 Training!



April 2-5, 2017 • AUSTIN, TEXAS

SPTechCon offers classes and tutorials for IT professionals, business decision makers, information workers, developers and software and information architects. Each presenter at SPTechCon is a true SharePoint expert, with many drawn from Microsoft's tech teams or holding Microsoft MVP status.

Whether you're looking to upgrade to a more current version, making a move to the cloud, or simply need answers to those daunting problems you've been unable to overcome, SPTechCon is the place for you! Come join us!

- Choose from more than 80 classes and panel sessions
- Improve your skills and broaden your knowledge of Microsoft's collaboration and productivity software
- Learn about SharePoint 2016, the latest on-premises server release from Microsoft
- Tips and tricks for working with SharePoint 2013 and 2010, and Office 365
- Practical information you can put to use on the job right away!
- The most knowledgeable instructors working in SharePoint today

www.sptechcon.com

Actually Cutting the Cord

Local broadcast channels on Roku? Yes!



SHAWN POWERS

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

◀ PREVIOUS
Kyle Rankin's
Hack and /

NEXT ▶
New Products

IN MY LAST ARTICLE, I TALKED ABOUT VARIOUS STREAMING OPTIONS AVAILABLE FOR FOLKS WHO WANT TO GET RID OF THEIR CABLE BILLS. You might remember that although I found lots of options, I hadn't actually canceled my cable service because I couldn't get local channels. Since writing that article, I did in fact cancel my subscription and managed to get local channels anyway. But, it wasn't as easy as putting rabbit ears on my television, that's for sure! If getting those local channels is something you're interested in doing, but struggle with the logistics, I encourage you to read on.

Finding Towers

Figuring out what channels are available in your area can be the most difficult part of cutting the

THE OPEN-SOURCE CLASSROOM

cable cord. Thankfully, with the advent of digital TV, several wonderful websites and apps were created that can help you determine your free TV potential. It's interesting that not all websites agree on signal strength. If you recall, as I wrote in my last article, I was certain I couldn't tune in any channels. That certainty was based on reports from several websites. Thankfully though, I expanded the search and decided to try it anyway. And, I currently get all the major stations even though I expected to get none!

Before we moved to the city of Petoskey, Michigan, we were in a rural area 30 miles east. Our house was in a valley, but thanks to AntennaWeb (<http://antennaweb.org>), I was able to determine that we could get stations with an antenna on a mast (Figure 1). So when we moved to our new house, I used the same website, and it informed me that we'd be able to get zero channels (Figure 2), regardless of what sort of antenna we used. That's when I gave up—at least temporarily. I tried another website, TV Fool (<http://www.tvfool.com>), and was surprised to see quite a few channels that were available, but just were weak (Figure 3). My suspicion is that AntennaWeb cuts off the stations that are really weak. Still, I'd like to know what's available.

In the end, I found a few apps in the Google Play Store that were incredibly useful. Not only do the apps locate nearby towers, but if your phone has

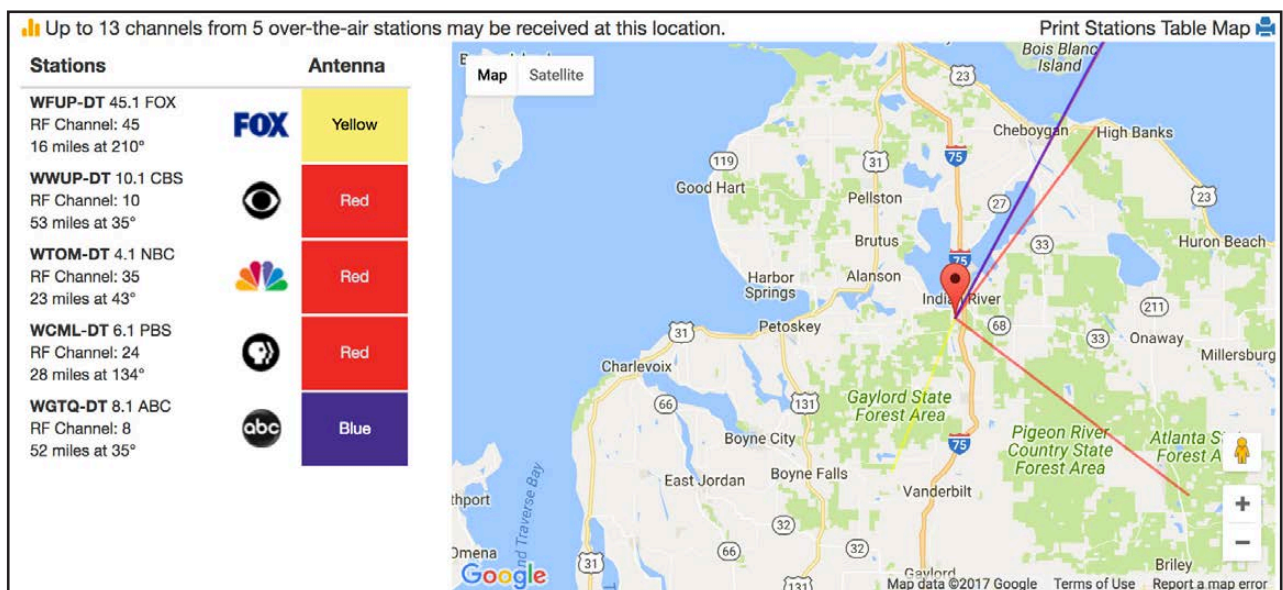


Figure 1. Who knew living in a small town would mean better reception?

THE OPEN-SOURCE CLASSROOM

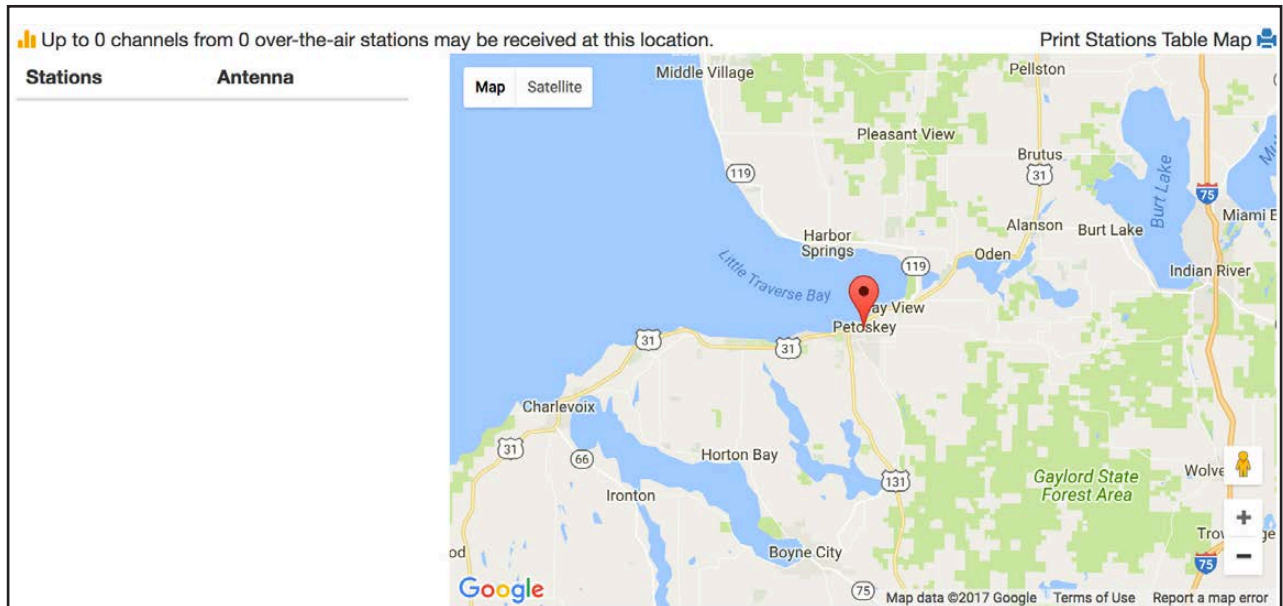


Figure 2. This was depressing.

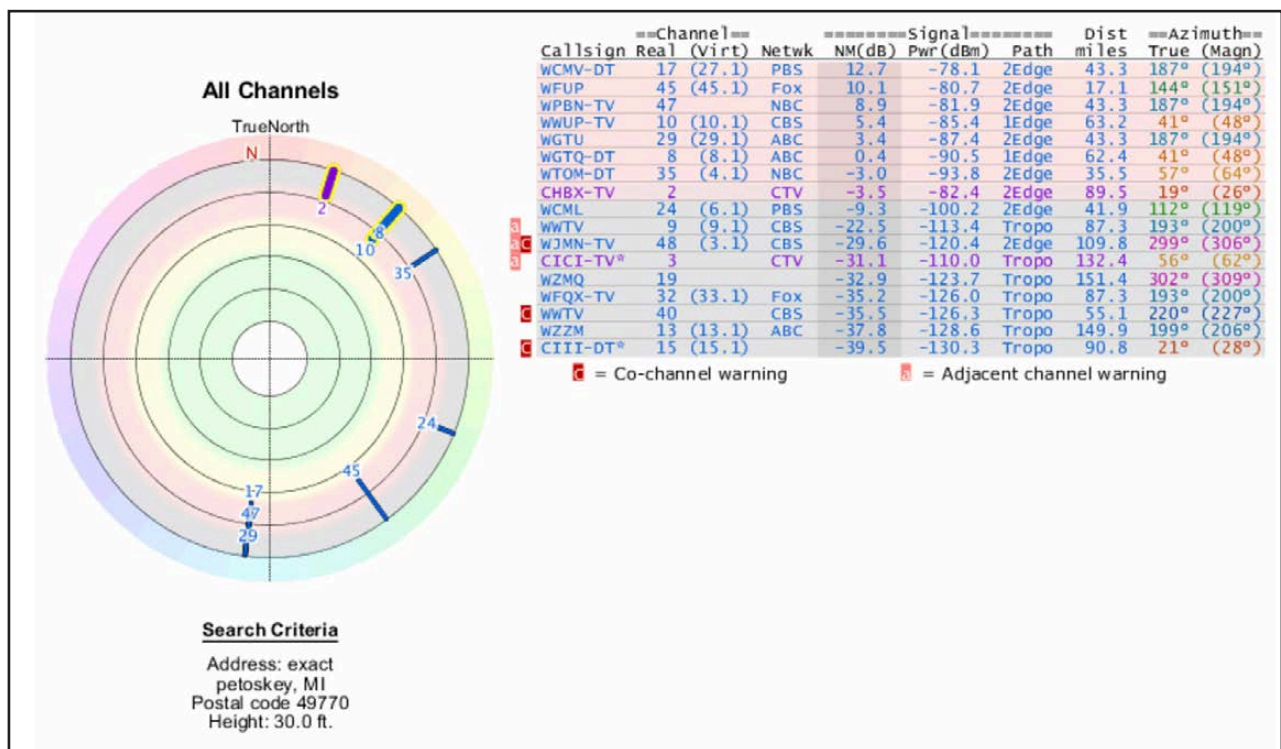


Figure 3. Thankfully, TV Fool offers information on distant stations.

a compass built in, it helps you point your antenna for pinpoint accuracy. Several apps are available, but I use Digital TV Antennas (Figure 4): <https://play.google.com/store/apps/details?id=ar.com.lichtmaier.antennas>.



Figure 4. These apps make pointing antennas so much easier.

Choosing Antennas

Once you've determined which towers you want to try, you can choose the antenna (or antennas) you want to use. If AntennaWeb shows you available towers, it also shows you the "color" antenna you'll need to tune in the various channels. Those ratings are standard, so you can look for a "blue" or "green" antenna and have a fairly good estimate of what

stations you'll get. My towers are "fringe" at best, so I needed to get the biggest, baddest antenna I could get. I also needed a preamp (more about that in a bit).

If you're in an area with strong signals, you will likely be able to use an omnidirectional antenna to tune in towers from all directions. In fact, many people do really well with rabbit ears, especially if they're in a city near a television tower. If you need to pull a station from a long distance, however, you'll need a directional antenna. And, even those antennas come in multiple flavors, and they all have various strengths and weaknesses.

Before you pick an antenna, you have to determine what type of signal you need to receive. The spectrum is divided into three parts: Low VHF, High VHF and UHF. Those translate into channels 2–6 for Low VHF, channels 7–13 for High VHF and 14–51 (used to be 69) for UHF. The majority of antennas do a poor job with Low VHF, because the wavelengths require huge antenna booms to receive. If you need to pull in a Low VHF channel, be sure that whatever antenna you end up buying is designed for Low VHF. The good news is more and more stations are moving away from Low VHF, but it's often hard to tell without looking at sites like TV Fool or AntennaWeb, because stations will use "virtual channels" to preserve their station identification.

If you're a television station that has been "NBC on channel 2" for 50 years, that channel recognition is hard to give up. So those stations end up switching to a UHF frequency, but encoding a "virtual" channel so the television shows its original branded channel number. If you look back at Figure 3, you'll see that channel 35 has a virtual channel of 4.1. When you switch channels on your television, it will actually show the station on channel 4, even though it's actually being broadcast on UHF channel 35! It's confusing, yes, but it's honestly a good thing, because it means fewer and fewer stations actually require a Low VHF antenna. In fact, every single station available to me is a UHF station, so I can get a UHF-only antenna.

Antenna Angle

If you need a Low VHF channel, or even a High VHF channel, your options are a bit more limited. In fact, you'll likely want to get a

THE OPEN-SOURCE CLASSROOM

traditional television antenna like the Channel Master 3020 (<http://a.co/83uaKeY>), which is around 15-feet long and 8-feet wide! Obviously I'm going to the extreme size, and if you don't need to pull in weak stations, you can get away with something smaller. With an antenna this big, an outdoor roof installation makes sense, but I've installed something similar in my attic (Figure 5) and had fairly good luck. Unfortunately, this style of antenna is fairly directional, so they work best if your towers are in the same direction. They can certainly pull a bit of signal from towers not directly in front of them, but they do best while pointed at the tower, especially VHF towers.

Using the Android app mentioned previously, pointing an antenna like this is trivial. If you're trying to find a sweet spot between two towers, however, the best thing to do is connect a television and make



Figure 5. This photo doesn't do the size of the antenna justice. It's enormous!

THE OPEN-SOURCE CLASSROOM

minor adjustments until you get the most stations to come in clearly. Keep in mind that weather affects reception, so a station that comes in clearly one day might not come in at all on another day. It can be frustrating. I've taken far too many trips to my attic attempting to get the best compromise between quality and quantity of stations!

If you just need UHF channels, I highly recommend getting a bow tie antenna. Like the traditional-shaped antenna, they can be mounted indoors or outside, but they are more affected by wind than traditional antennas. Bow tie antennas often are worth the hassle, however,



Figure 6.
This antenna hasn't arrived yet, but I have high hopes.

If you go to a department store, you'll likely see lots of amplified antennas, which are marketed as magic bullets to enhance even the weakest signal. That's not what amps do for television signal.

because they do an incredible job with UHF channels, plus they have a wider area of reception. Every antenna is rated for a different angle of reception, but if you have a few towers that a unidirectional antenna can't get at the same time, a bow tie might work better. Unfortunately, the math goes only so far, and you'll likely have to try various models to find the best match. I ordered the HD4228 antenna (<http://a.co/5qPRjcm>), hoping it will pick up multiple towers a bit better than my traditional directional antenna (Figure 6). I recommend unwrapping antennas carefully, because it's possible you'll need to return them when they don't work like you hope. I didn't do that, and I now have three antennas I can no longer return.

Amp? Preamp?

If you go to a department store, you'll likely see lots of amplified antennas, which are marketed as magic bullets to enhance even the weakest signal. That's not what amps do for television signal. Also, note that "amp" and "preamp" are the same thing when it comes to television antennas. (There might be a technical difference an RF pro can explain, but for all intents and purposes, "amp" and "preamp" both mean preamp when it comes to television antennas.)

A preamp does not make a weak signal stronger. If your antenna doesn't tune in the station, an amp won't help. What a preamp does help with is eliminating signal loss on the way from the antenna to the television. That's why preamps are installed as close to the antenna as possible—usually directly on the mast. The preamp takes the signal from the antenna and strengthens the signal on your in-house coax cable. Think about it like tuning in a radio station with an antenna and



Figure 7.
The preamp
is as close to
the antenna
as possible.

having an amp to make the volume louder. It's the same process. The amp makes the signal "louder", so all the televisions in the house can "hear" the signal, but it does not make the signal any clearer.

Just because preamps aren't the magic-bullet antennas marketing folks would like us to believe, they are still a vital part of retrieving a weak signal. Every connector, splitter and foot of cable introduces signal loss into the system. A preamp guarantees every bit of reception gets to the television tuners, and with fringe towers, it can make all the difference. Thankfully, preamps are relatively inexpensive and easy to



Figure 8.
The power injector can be a bit away from the actual preamp.

install. I use the Wineguard LNA-200 (<http://a.co/2xCR5rn>, Figure 7), which is powered by an injector (Figure 8) in the cable before connecting to televisions.

Multiple Antennas?

If you have towers that are in different directions, it's possible to point two directional antennas and combine the signal to get both. Logically, this makes a ton of sense, especially if you have towers that are impossible to get with a single antenna. Unfortunately, it seldom works

in practice. The problem is that along with combining reception, it also combines interference and noise. The end result is usually channels that have a ghosting effect, and instead of getting both towers, you end up with neither.

If you want to try using multiple antennas, be sure to get the smallest directional antennas you can use and still get a signal. The idea is to tune in only the tower you're pointing at, so the interference is limited. Still, it's a long shot and seldom works. There are additional steps you can do like installing channel filters so only specific frequencies come through to the antenna combiner, but it becomes complicated and expensive very quickly. I've never had luck with multiple antennas, but some people do.

Seven Televisions?

There are seven televisions in my house: one in each bedroom, one in my office, one in the upstairs family room and one in the living room. I have Ethernet or Wi-Fi to each TV, but only one has COAX. I have no desire to run COAX cables and enough preamps to get television signal to each room. So for me, there has to be another step to cutting the cord—namely, network distribution of television channels. Thankfully, a few options work really well.

The Tablo DVR (<http://a.co/7bmdB5q>) is a device that has built-in tuners for watching and recording OTA (over the air) television (Figure 9). It requires an external hard drive and a subscription for TV guide



Figure 9. I've heard good things about the Tablo, just not the price.

information. Some apps make watching television easy on devices like the Roku, but paying for a subscription feels a bit like cable television fees. The system is reported to work well, however, so if you're looking for a DVR option, it's worth looking at. I want my local stations only for watching the news and the Thanksgiving Day Parade, so DVR isn't really high on my priority list, so I chose to install an HDHomeRun CONNECT device.

The HDHomeRun CONNECT (Figure 10) has two built-in tuners and nothing else. It connects to Ethernet and acts like a network-based television tuner for any device that can connect to it. There are computer apps for watching television, but the real awesomeness comes when you use something like Plex Media Server. Plex DVR supports the HDHomeRun as a tuner, so you can use the DVR feature of Plex. Unfortunately, Plex DVR doesn't allow you to watch live television, just recordings after the shows are done.

Plex has two channels that let you watch live television from any Plex device (Figure 11)—that includes remote mobile devices! This means I can watch my local television stations even while traveling. That's awesome. Neither channel is "official", but each easily can be



Figure 10. I love the HDHomeRun, but I wish it had more than two tuners.

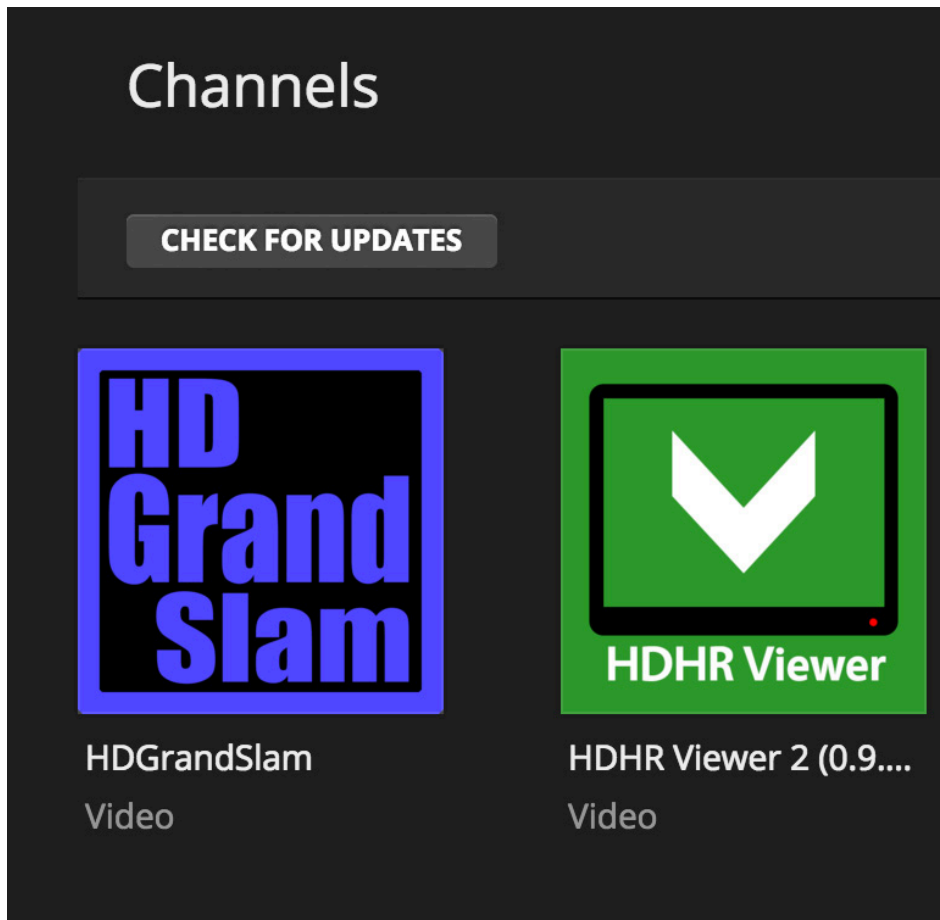


Figure 11.
HD Grand Slam
has a cooler
icon, but I prefer
HDHR Viewer 2.

installed on the Plex Media Server. HDGrandSlam (<https://github.com/jumpmanjay/HDGrandSlam.bundle>) has a more complex menu system, but it detects what shows are currently playing on your OTA channels. The other channel is HDHR Viewer 2 (<https://github.com/zynine-/HDHRViewerV2.bundle>), which has a simpler menu system, but other than that, it's similar in function. I installed both, but I find myself using HDHR Viewer 2 more often simply because it's fewer clicks to get to the television. It's interesting to note that both channels take a significant time to start playing the station. I think that's due to transcoding and buffering. It's tolerable, but just know that channel surfing isn't fun.

Am I a Happy Cord-Cutter?

If there was a magic bullet that made cord-cutting possible for me, it would be Roku. I have a few Roku TVs and some non-smart TVs with

THE OPEN-SOURCE CLASSROOM

Roku devices connected to them. Roku allows me to use PlayStation Vue, Plex, Netflix, Hulu, Amazon Prime and live TV (via Plex), with the option to switch to SlingTV if Vue ever becomes too expensive. If my family members had to switch sources every time they switched from television to Netflix, it would be frustrating. Heck, the newer Roku boxes even have HDMI-CEC, which allows volume to be controlled with a single remote, which means no more piles of remotes for each room.

I've used just about every home theater set-top-box device around, from the original Xbox running XBMC, to a Raspberry Pi running Kodi, to the Boxee Box and Popcorn Hour. With a Roku, a Linux server and a bit of elbow grease, I've finally gotten to the point where watching television doesn't require any special tech skills. Heck, I even can turn the television on and off with my Amazon Echo—but that's another article altogether.■

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

[RETURN TO CONTENTS](#)

NEW PRODUCTS

PREVIOUS



Shawn Powers'
The Open-Source
Classroom

NEXT

Feature: Robots
and Linux, a Match
Made in Space



VMKings' VPS Hosting Solution

The management team of cloud provider VMKing, as developers themselves, found standard virtual servers not to be well tailored to the developer community—too much or too little space, insufficient security and no support for their preferred Linux OS(!). To meet the specialized needs of developers everywhere, VMKings developed a fast, uncomplicated, developer-focused VPS (Virtual Private Server) hosting solution. Now developers can serve as administrators of their own maintenance-free, customizable solution that can be upgraded instantaneously and scaled to match user needs. The hosting adapts itself to the preferences of the user, not vice versa, says VMKings. Key features include the choice of distributions and frameworks that best suit the project, spin up and root access in minutes, flexible and scalable resource configurations, 99.9% SLA, blazing-fast enterprise-grade SSDs, a 40-gigabit fault-tolerant network with multi-regional data-center support and enterprise-grade hypervisor technology at the back end.

<http://vmkings.com>



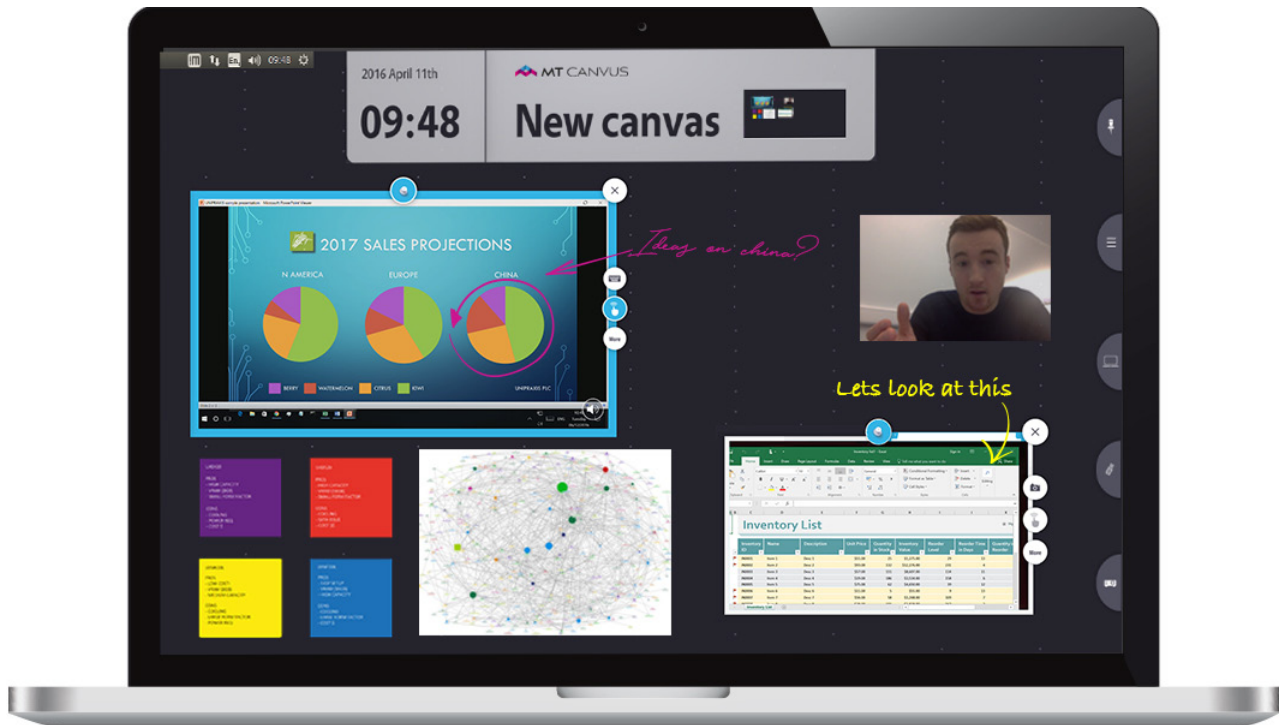
MENDER.io

Deploy Software Updates for Linux Devices

Mender

The new production release of Mender 1.0, an open-source tool for updating embedded devices safely and reliably, is now available. Mender's developers describe the tool as the "only open-source over-the-air (OTA) software updater for embedded Linux devices that integrates both an updater client and deployment management server", both of which are licensed under Apache 2.0. Using Mender, embedded development teams venturing into IoT can focus completely on developing their product instead of squandering time on building and maintaining a homegrown software updater or cobbling together a custom combination of tools or scripts to deploy software updates. Mender utilizes a dual A/B rootfs partition layout to deploy full image updates and provides automated rollback of failed updates for any reason, including power loss on the device or poor network connectivity. This feature allows users to deploy software updates confidently without the fear of bricking their devices. The Golang-based Mender runs efficiently on the embedded device and is tightly integrated with the Yocto Project and U-Boot, making Mender easy to integrate with most production-grade embedded boards. Going from a fresh system to completing your first managed deployment with Mender, including server setup, sayeth Mender folk, should take less than one hour.

<http://mender.io>



MultiTaction's MT Canvas-Connect

“A new era in visual collaboration” is the promise of MT Canvas-Connect, MultiTaction’s new real-time collaboration software that enables visual touchscreen collaboration across remote locations in real time. The platform-agnostic solution provides remote users the means to share, manipulate, draw and input information in real time wherever they are based. MT Canvas-Connect grew out of MultiTaction’s MT Canvas product, which “is already unique in the way that it helps visualize big data and encourages collaborative working”. It enables multiple sources, such as video feeds, web-based applications and content from smart devices, to run in parallel. MT Canvas-Connect extends this functionality by enabling multiple remote locations to collaborate in real time. MultiTaction is also a leading manufacturer of ultra-responsive large touch-display systems, so the software solutions are all designed from the ground up to function in touch-enabled environments. MT Canvas-Connect operates in both Linux and Windows environments and integrates with current meeting-room technologies.

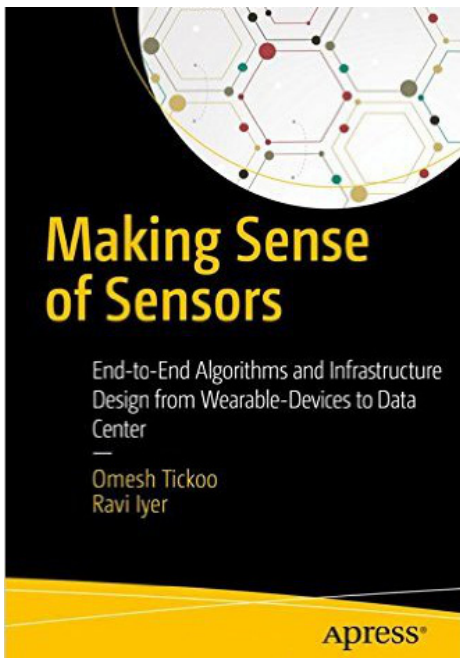
<http://multitaction.com>



CodeLathe's Tonido Personal Cloud

CodeLathe counts Dropbox and OneDrive as competitors to its Tonido Personal Cloud storage service. While the former can cost anywhere from \$50 and up per year to store a TB of data, CodeLathe's cross-platform Tonido for the desktop is fully free, and storage space is limited only by users' hard drive capacities. CodeLathe also boasts 100% private, secure file sharing due to, for instance, not storing user credentials by effecting authentication directly between the client device and the Tonido server running on a home PC. Recently, the company launched a next generation of the Tonido personal cloud software that offers remote access to all documents, photos, music and videos stored on Intel Core processor-based desktop PCs from a web browser, smartphone, tablet and DLNA-enabled devices. Seamless global access is enhanced via Tonido Relay servers strategically placed across three continents and six countries.

<http://tonido.com>



Omesh Tickoo and Ravi Iyer's *Making Sense of Sensors* (Apress)

In today's data-driven world, we are surrounded by sensors collecting various types of data about us and our world. These sensors are the primary

input devices for wearable computers, IoT and other mobile devices. Professionals seeking to better understand today's sensor-rich devices and acquire knowledge and skills to develop innovative solutions that exploit them will be pleased to learn about the new book *Making Sense of Sensors*. Written by the team of Omesh Tickoo and Ravi Iyer and published by Apress, the book is subtitled *End-to-End Algorithms and Infrastructure Design from Wearable-Devices to Data Centers*. Presented in a manner that permits readers to associate the examples with their daily lives, Tickoo and Iyer's book covers the most common architectures used for deriving meaningful data from sensors. This book provides readers with the tools to understand how sensor data is converted into actionable knowledge and provides tips for in-depth work in this field. Starting with an overview of the general pipeline to extract meaningful data from sensors, the book then dives deeper into some commonly used sensors and algorithms designed for knowledge extraction. Practical examples and pointers to more information are used to outline the key aspects of Multimodal recognition. The book concludes with a discussion on relationship extraction, knowledge representation and management.

<http://apress.com>



openHAB

Partners Canonical, openHAB Foundation and Azul Systems have collaborated hard to drive development of the new openHAB 2.0 smart-home platform as a snap package. An alternative to Apple Homekit and Samsung SmartThings, openHAB from openHAB Foundation is completely free and open source, and acts as a control hub for home IoT setups. The platform is easy to install, highly customizable and comes with great performance across a wide range of hardware, from PCs to Raspberry Pis. Furthermore, openHAB can be used to control, automate and complement smart-home setups. As an open platform, openHAB is not tied to any single brand and supports many protocols and technologies, which enables consumers to mix and match devices. Ubuntu's contribution to openHAB is snap, a secure universal Linux application format that makes applications available as a simple one-click download and install from the Ubuntu Appstore. The snap packaging of openHAB makes it simpler for home-automation creators to build, test and distribute their smart-home services. Finally, Azul System's contribution to openHAB is Zulu Embedded Java Runtime, which is available for a wide range of hardware and provides optimal performance on home gateways, PCs or ARM-compatible devices. The sum of the partnership between openHAB, Ubuntu and Azul means openHAB can be packaged and distributed as a single application, including a tested Java Runtime, without click-through licenses.

<http://openhabfoundation.org>



PasswordPing Ltd.'s Exposed Password and Credentials API Service

The typical online user has an average of 90 active and inactive online accounts. This exposure to threats, notes software and IT security specialist PasswordPing Ltd., helps to inform us why billions of credentials have been exposed in the past five years alone. To assist organizations and companies to screen their user accounts for known, compromised credentials, PasswordPing Ltd. announced the launch of its new password and credential breach notification service. Organizations can be alerted of exposed credentials and request users to update their credentials when they set up their account, reset their password or log in to their account. At its core, PasswordPing is a massive cloud database of exposed credential data. The company's various web-based APIs provide different views into this data, tailored for different use cases. The Passwords API, which can be used as part of a sign-up form or a password-change form, takes a password and returns a result indicating whether that password has been exposed. The Credentials API takes a user name/password combination and returns a result indicating whether those credentials have been exposed. If a user's credentials are exposed, the user can be notified and prompted to change the password.

<http://passwordping.com>



CyKick Labs

CyKick Labs Ltd.'s Telepath

When a shopper enters a store, the retailer doesn't know if the person will simply browse, make purchases, shoplift or hold up the register. The same goes for visitors to a website. The challenge is to prevent and stop the bad guys without hindering beneficial customer transactions. A new approach to threat protection, Telepath from CyKick Labs Ltd., leverages big data to apply behavioral analytics and machine learning to protect web applications against exploitation by individuals and bots alike. With Telepath, cyber security teams get early and actionable intelligence on emerging threats and comprehensive forensics capabilities, as well as insight into sophisticated attacks. Telepath is able to stop fraud by understanding normative site visitor behavior, inspecting enterprise web application transactions non-intrusively. It employs a hybrid approach based on rules and machine learning to identify suspect behavior. The solution protects web applications from multiple threats, such as industrial espionage information theft, account takeover, service disruption, business logic abuse and insider threats.

<http://cykicklabs.com>

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

RETURN TO CONTENTS

ROBOTS AND LINUX, A MATCH MADE IN SPACE

As robots become more prevalent, they soon will be working alongside humans, relieving some of the burdens as well as taking on more of the dangers. The systems to make that happen are being built today, and they are using Linux in a major way.

PAUL FERRETTI



PREVIOUS
New Products

NEXT

Feature: Key
Considerations for
Software Updates for
Embedded Linux and IoT



I grew up in the sixties, during the height of the space race. I vividly remember watching the first moon landing, and I have loved space and all things space since those early days. Alas, I did not become an astronaut; instead, I ended up in the tech field writing software. And as much as I loved NASA, the closest I have come to working with NASA is vicariously through my nephew who works at the Johnson Space Center in Houston. I spent many summers with my nephew building and programming robots, and it's a hobby that I still enjoy very much.

So, when I became aware that one of NASA's Centennial Challenges was called the Space Robotics Challenge, I immediately reached out to several friends to see if they were interested in forming a team. After getting commitments from four other like-minded individuals, I registered our team to compete in the challenge.

NASA AND THE SPACE ROBOTICS CHALLENGE

The Centennial Challenges program is part of NASA's Space Technology Mission Directorate (STMD). The STMD creates various challenges in order to bring together members of industry, academia and the government with the goal being the advancement of innovation in key areas of technology important to the agency. These challenges are meant to engage individuals and teams from all walks of life. Whether you are a hobbyist, such as myself, or a member of a team from a tier-one technology lab, such as MIT, NASA has made these challenges accessible to everyone.

The technical coordinator for the Space Robotics Challenge (SRC) is NineSights, and according to its website, "The SRC focuses on developing

FEATURE: Robots and Linux, a Match Made in Space

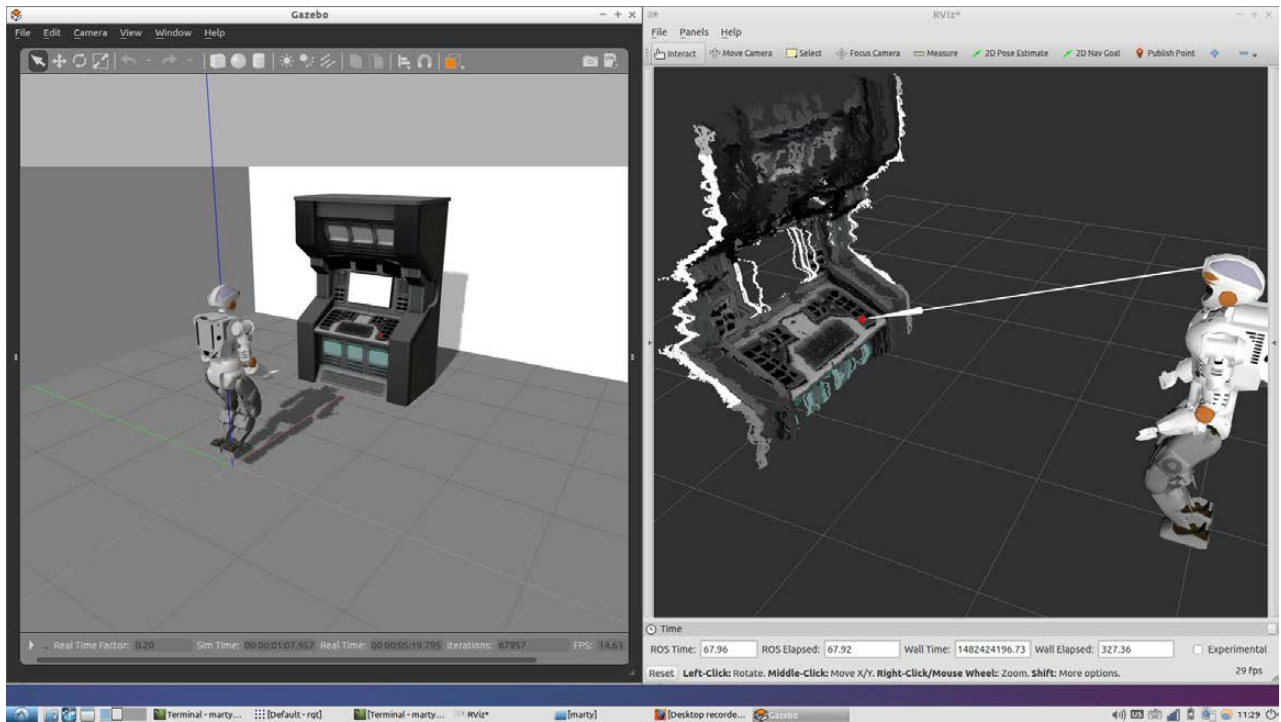


Figure 1. Using RVIZ to Check Line of Sight

software to increase the autonomy of dexterous mobile robots in humanoid format—specifically NASA’s R5 robot—so they can complete specific tasks during space travel or after landing on other planets (such as Mars), as well as on Earth.”

The SRC involved two rounds of competition: the first round, better known as the qualifying round, and the second round, referred to as the virtual competition. The qualifying round involved two tasks: a visual task and a mobility task. The visual task had the robot positioned in front of a panel that consisted of a number of LED lights. The lights would flash in a random sequence, and the task was to identify the location and color of the lights in the correct sequence (Figure 1).

The mobility task started with the robot standing behind a red line approximately four meters from a closed door. The robot needed to walk up to the door, press a button to open the door and then walk through the door for approximately one meter before ending up past a red line located on the pathway. The scoring for this task was based solely on the time it took the robot to walk from red line to red line (Figures 2 and 3).

As you can see, this challenge had two very important features: robots

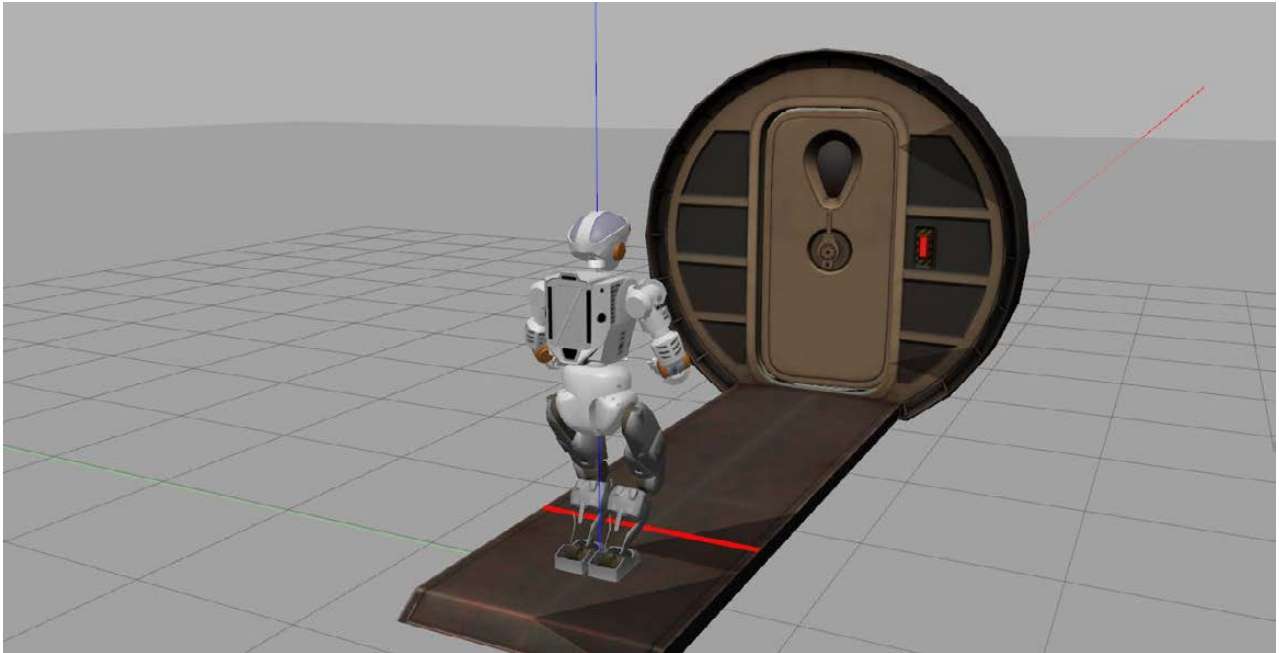


Figure 2. Robot in Starting Position

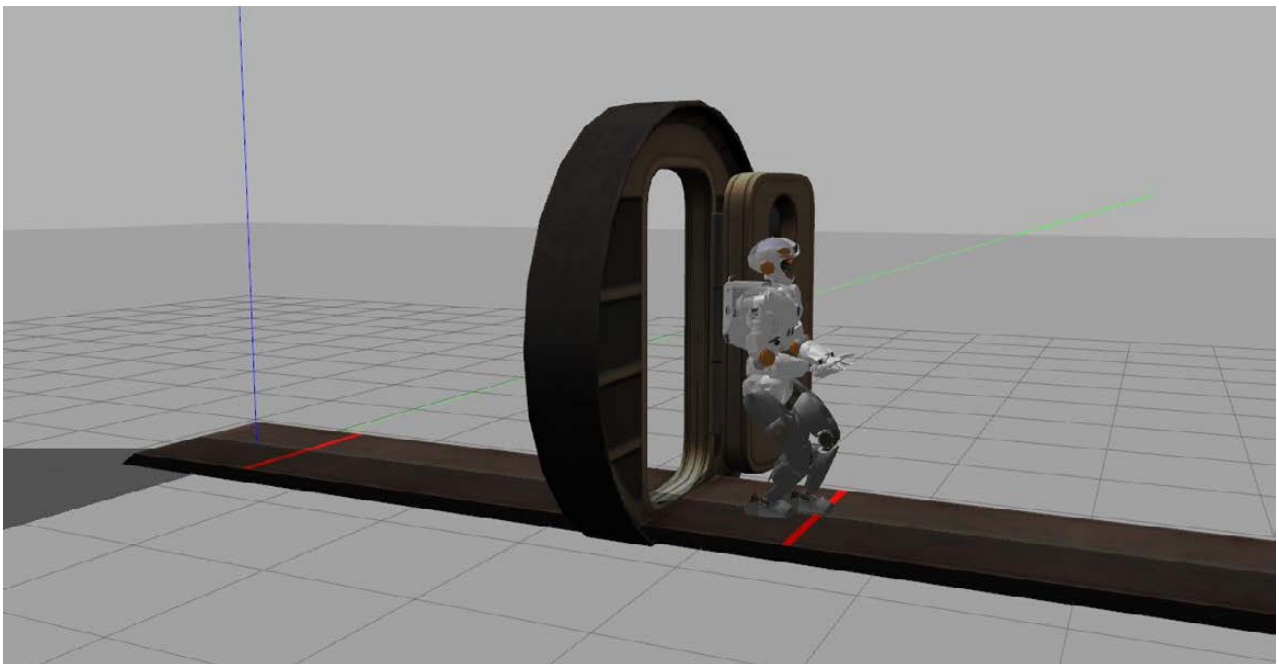


Figure 3. Robot in Finished Position

and space. That was all I needed, but I was soon to find out the challenge included a third element that was of great interest to me. The operating platform that the challenge would be working in was Linux-based. In addition, the challenge would be using the Robotics Operating System

(ROS), a simulation software called Gazebo and the Space Robotics Challenge Simulation (SRCSim), a predefined simulation that included the environment for the qualifying tasks, created for the Space Robotics Challenge by the Open Source Robotics Foundation (OSRF).

ROBOTICS AND LINUX

Robots have been around for a long time, and during most of that time, if you wanted to build your own system, you had to build the hardware, and you also had to create the software that would control the robot. This is no small task, and it required several skill sets that not everyone had. Unfortunately, the need for special skills was a deterrent that kept potential participants from jumping in to the robotics arena.

These days, there's a tremendous number of robot kits, like those made by Vex Robotics and Makeblock, which utilize an Arduino controller. Robotics kits like these greatly have reduced the need to fabricate your own robot; however, if you possess the necessary talent to fabricate a robot from scratch, you still can do so. For a long time, what the robotics community was lacking was a solid software platform with which to work.

In 2006, Microsoft was the first major player to enter the mainstream robotics software field with Robotics Studio version 1.0. Later, in 2008, Microsoft released Robotics Developer Studio (MRDS). The MRDS is a Windows-based environment for robot control and simulation. As with all Microsoft software, it is closed source. The primary programming language for MRDS is C#. Besides the closed-source nature of the MRDS, it has been described as overly complicated and requires a great deal of overhead. Along with that, the latest version, RDS4, has not had an update on its website since June 2012. A more sustainable solution was needed. Software sustainability increases when the maximum number of people are able to access it. Enter the Open Source community, led by Linux.

In 2007, Willow Garage, a robotics incubator in California, stepped in to extend work that was done by others, mainly at universities such as Stanford, to create a well tested implementation of an open-source operating environment for robotics. Linux, which has a long successful history on which to rely, was the platform utilized to provide an open-source foundation.

It makes complete sense that Linux would emerge as the dominant platform for robotics. The open-source nature of Linux, and by extension ROS, allows a

IT MAKES COMPLETE SENSE THAT LINUX WOULD EMERGE AS THE DOMINANT PLATFORM FOR ROBOTICS.

greater audience to access, develop and maintain a robust environment for the development and distribution of robotics-related software. Linux is widely used in academia as well as by home hobbyists. It's free, it's stable, and it runs on a variety of platforms, without the need for cutting-edge hardware.

REQUIREMENTS FOR THE SRC

The Technical Coordinators for the Space Robotics Challenge had specific software requirements for the participants of the challenge. The challenge organizers required teams to use the Linux-based Robot Operating System (ROS) and the 3D multi-robot simulator software package called Gazebo.

ROBOT OPERATING SYSTEM

ROS is a flexible framework for writing robot software. It is a collection of tools, libraries and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms. The ROS ecosystem now consists of tens of thousands of users worldwide, working in domains ranging from tabletop hobby projects to large industrial automation systems.

ROS is mainly targeted for the Ubuntu distribution of Linux. However, the source code is available for compilation in the many different flavors of Linux. The Ubuntu distribution provides the easiest implementation and the most support.

GAZEBO

Robot simulation is an essential component in every roboticist's toolbox. A well designed simulator makes it possible and quick to test algorithms, design robots, perform regression testing and train artificial intelligence (AI) systems using realistic scenarios.

Gazebo is a 3D multi-robot simulator that generates both realistic sensor feedback and physically plausible interactions between objects, and it

GAZEBO IS A 3D MULTI-ROBOT SIMULATOR THAT GENERATES BOTH REALISTIC SENSOR FEEDBACK AND PHYSICALLY PLAUSIBLE INTERACTIONS BETWEEN OBJECTS, AND IT INCLUDES A HIGHLY ACCURATE SIMULATION OF RIGID-BODY PHYSICS.

includes a highly accurate simulation of rigid-body physics. Gazebo also offers the ability to simulate populations of robots in complex indoor and outdoor environments accurately and efficiently. It includes multiple robust physics engines, high-quality graphics and convenient programmatic and graphical interfaces. Best of all, Gazebo is free with a vibrant user community that is constantly adding new simulations as well as extending existing ones.

Gazebo comes with many built-in robot simulations that are ready to use. Some of the more interesting ones include the Atlas Robot from Boston Dynamics; Baxter, an industrial robot, built by Rethink Robotics; and NASA's Robonaut2. Many more simulations are available, and all of them are modifiable and extendible by anyone.

INSTALLING ROS

For the SRC, the challenge rules mandated that a specific version of ROS on a specific version of Ubuntu be used. The moderators of the challenge wanted everyone to be working in the same environment and, therefore, face the same challenges and have the same benefits. Specifically, teams participating in the SRC were going to be using the ROS-Indigo version targeted for Ubuntu Desktop 14.04 (Trusty 64-bit amd64).

USING DOCKER

Complying with the SRC technical requirements meant using ROS-Indigo and the 14.04 version of Ubuntu. At the time, I was running a dual-boot Windows 10 (I know, forgive me), Ubuntu 16.04 Xenial 64-bit system.

Initially, I thought this would not be a problem, as I had figured I simply could install a Docker container running Ubuntu 14.04 and the required versions of ROS and Gazebo.

Docker containers allow you to wrap software in a container that keeps everything separate from the base operating system. The nice thing about Docker containers is that there are tons of containers already available for running various applications. I found a Docker container that was already configured with Ubuntu 14.04. All I had to do was install Docker on my 16.04 distribution and then install the 14.04 container. The install was fast and easy, and even though I had not used Docker previous to this, I found plenty of instructions on the web on how to set up my container as well as how to save various versions of the container as I installed other pieces of software. This allowed me to roll back to a working version of the software whenever I messed things up.

Unfortunately, even though I was able to install ROS and Gazebo in my Docker container, I never was able to get the SRC Simulation to run properly. I spent the better part of a weekend trying to solve the graphics problems, but in the end, simply gave up in lieu of an easier, albeit less elegant solution. I installed an additional hard drive in my computer and created a tri-boot system. Now I could boot into a separate version of Ubuntu 14.04 and install the necessary software without worrying about conflicting drivers. As a side note, I could have run Linux from a USB stick, but I figured I would get better performance using a hard drive. Plus, I had a spare 1 terabyte drive lying around, so why not use it.

Installing ROS was a breeze. I simply navigated to the ROS website and followed the instructions to install the version of ROS targeted for the indigo distribution of Ubuntu. In less than five minutes, I had a working version of ROS, complete with tutorials to get me up and running with total ease.

GAZEBO INSTALLATION

The SRC also mandated that teams use a specific version of Gazebo. The version of Gazebo packaged with ROS-Indigo is version 2.x.x; however, for the SRC, we needed to remove that version and upgrade to the latest version, which at the time of the SRC was version 7.0.x.

After loading the correct versions of ROS and Gazebo, the next step was to set up the SRC Simulation. The Open Source Robotics Foundation

THE ROS ENVIRONMENT ALLOWS YOU TO CODE IN MANY LANGUAGES. IN FACT, YOU CAN CODE IN MULTIPLE LANGUAGES AND HAVE THEM WORK TOGETHER SEAMLESSLY THROUGH THE CREATION OF SEPARATE ROS NODES.

(OSRF) created and maintained the SRCSim and they provided a web page with all of the necessary steps to perform the SRCSim installation and setup. Links to all of the software needed to run the simulation are found at the Resources section at end of this article.

QUALIFICATION ROUND

Once the install of the ROS, Gazebo and SRCSim software was completed successfully, the teams would be ready to start the actual process of coding solutions to the two qualifying tasks. The ROS environment allows you to code in many languages. In fact, you can code in multiple languages and have them work together seamlessly through the creation of separate ROS nodes.

ROS nodes are basically pieces of code that help control the robot. Robot control usually involves multiple nodes. Nodes can control things like a laser range-finder or the motors associated with wheels on the robot. Nodes also are used to perform path planning or to provide a graphical view of the robot's world. For the SRCSim, our team had nodes that controlled the robot's walking and the robot's arm movement. We also had a node that controlled the vision system for identifying LED lights.

The most often used languages for programming nodes are C/C++ and Python, although C#, Java, Ada and Assembly, along with a variety of other languages also will work. Our team chose Python mainly because most of us already had some experience with the language, and because it is a fairly easy language to pick up if you have never used it before.

The qualifying round lasted approximately three months. Three months is not a long time when you are working a full-time job and are trying to

find any amount of spare time to work on maximizing robot performance. In order to utilize our time as efficiently as possible, we split the team into two sections with each section being responsible for a single qualifying task. We also had an individual team member float between the two tasks to pitch in wherever needed.

Working with ROS in the Linux environment involves a good deal of command-line interaction. The ability to open multiple terminals and have each one dedicated to various operations was a real plus. Many times, I would have one terminal to launch the simulation, another terminal to produce output from the robot sensors and still another window to send commands via messaging to the robot. It was not unheard of to have five or more terminals open at a time all communicating via the ROS messaging network.

CHALLENGE RESULTS

More than 400 teams from 55 countries around the world pre-registered for the Space Robotics Challenge. Of those 400-plus teams, only 92 of them submitted valid solutions for both qualifying tasks. From the 92 qualifying submissions, 20 teams qualified to move on to participate in the Virtual Competition set for June 2017.

As a team, our goals for the challenge were simple. First and foremost, we wanted to have fun. Second, we wanted to learn. Each of us had a desire to become more proficient using ROS and Gazebo, while others wanted to get some experience with Linux and Python, having never used either of them. Finally, we wanted to be competitive. Our goal was to submit real solutions to the two qualifying tasks in the hope that we would be in the top 20 and move to the next round of the competition.

I am proud to say, we met all of our goals—well, almost all of them. We did submit solutions for both tasks. Unfortunately, we fell short of qualifying, but we definitely were competitive. All in all, it was a great experience, and I recommend that anyone who has an interest in robotics look seriously at utilizing the Linux stack of software mentioned in this article as a starting point.

TEAM MEMBERS

For the Space Robotics Challenge, my team, named Patriot Robotics, consisted of myself and four other talented individuals: Erica Kane, Dan

Sionov, Marty Kube and William Marrieta. They were an absolute pleasure to work with. All of us are amateurs in this arena, and although some brought a bit more experience to the team than others, we all worked together very cohesively. I want to thank each of them for making the experience both rewarding and enjoyable.

CONCLUSION

Working on the Space Robotics Challenge and collaborating with my team was great fun. The truly remarkable thing about participating in the SRC was that it cost nothing. In fact, for nothing more than the cost of your time, you can utilize the software mentioned in this article to run the same simulations that the Space Robotics Challenge used in its qualifying round. The basics of ROS and Gazebo are pretty easy to pick up. Without much effort and by using the supplied tutorials as well as other on-line

The TurtleSim Tutorial

One of the best things about ROS is that it has a very rich tutorial library. The tutorials are a great way to come up to speed with ROS and Gazebo. If you are interested in learning ROS, one of the first tutorials you should try is the turtlesim.

The turtlesim teaches you how to create ROS packages, work with multiple nodes, publish and subscribe to topics, and work with ROS services.

The tutorial provides step-by-step instructions as well as several online videos to walk you through the process of setting up and running the simulation. The simulation involves hands-on programming and command-line interaction through ROS. Learning how to make the simulated turtle move, rotate or spawn copies of itself are just some of the features of this tutorial.

The turtlesim is a fun and easy way to get ROS up and running. If for no other reason than to just control a simulated turtle, I encourage you to give it a try.

resources, anyone with little or no experience can set up and run the software. You even can try your hand at creating solutions and seeing if you can score well enough to have qualified for the Space Robotics Challenge. If you do, feel free to email me your results. Good luck! ■

Paul Ferretti has more than 20 years of experience as a software engineer. He is a Senior Member of the IEEE and a former Vice Chair for the Washington DC/Northern Virginia Chapter of the IEEE Robotics and Automation Society. He welcomes your comments sent to pferretti@ieee.org.

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

[RETURN TO CONTENTS](#)

Resources

ROS History: <http://www.ros.org/history>

About ROS: <http://www.ros.org/about-ros>

ROS.org, Installing from Source: <http://wiki.ros.org/indigo/Installation/Source>

Space Robotics Centennial Challenge:
<https://ninesights.ninesigma.com/web/space-robotics-challenge>

STMD: Centennial Challenges:
https://www.nasa.gov/directorates/spacetech/centennial_challenges/index.html

Space Center, Houston: <https://spacecenter.org>

Johnson Space Center: <https://www.nasa.gov/centers/johnson/home/index.html>

Open Source Robotics Foundation: <http://www.osrfoundation.org>

The Institute for Human & Machine Cognition (IHMC): <https://www.ihmc.us>

Patriot Robotics: <http://www.patriotrobotics.net>

Gazebo: <http://gazebosim.org>

Tutorials for Gazebo: <https://bitbucket.org/osrf/srcsim/wiki/tutorials>

turtlesim: <http://wiki.ros.org/turtlesim>

Key Considerations for Software Updates for Embedded Linux and IoT

How can we reliably update embedded Linux?

EYSTEIN STENBERG

PREVIOUS



Feature: Robots
and Linux, a Match
Made in Space

NEXT
Doc Searls' EOF



The Mirai botnet attack that enslaved poorly secured connected embedded devices is yet another tangible example of the importance of security before bringing your embedded devices online. A new strain of Mirai has caused network outages to about a million Deutsche Telekom customers due to poorly secured routers. Many of these embedded devices run a variant of embedded Linux; typically, the distribution size is around 16MB today.

Unfortunately, the Linux kernel, although very widely used, is far from immune to critical security vulnerabilities as well. In fact, in a presentation at Linux Security Summit 2016, Kees Cook highlighted two examples of critical security vulnerabilities in the Linux kernel: one being present in kernel versions from 2.6.1 all the way to 3.15, the other from 3.4 to 3.14. He also showed that a myriad of high severity vulnerabilities are continuously being found and addressed—more than 30 in his data set.

Although the processes and practices of development teams clearly have a critical impact on the (in)security of software in embedded products, there is a clear correlation between the size of the software project’s code

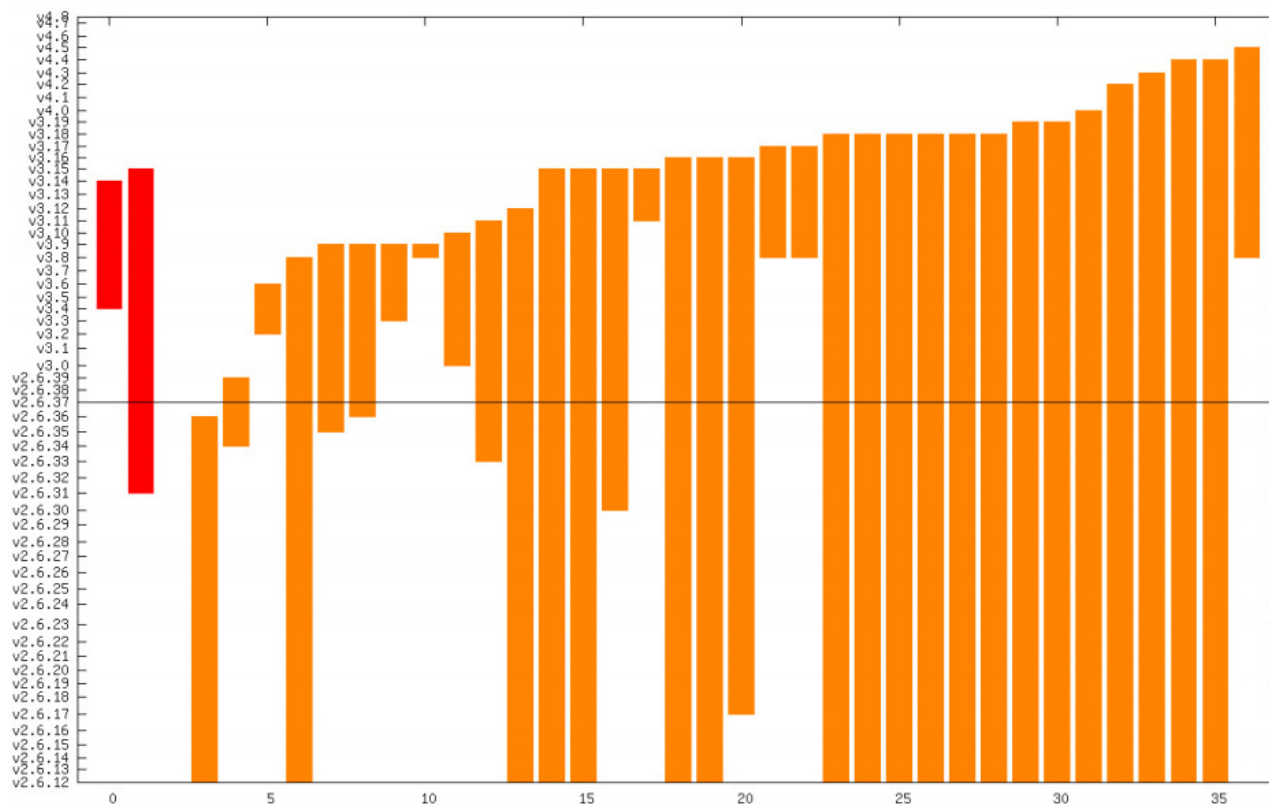


Figure 1. Linux Kernel Fix Timing

base and the number of bugs and vulnerabilities as well. Steve McConnell in *Code Complete* states there are 1–25 bugs and vulnerabilities per 1,000 lines of code, where the variable is determined by the practices of the team. Military-certified products typically end up in the lower end due to more rigid security practices and quality testing, while consumer electronics are unfortunately in the higher end of the scale due to the greater focus on features and time to market.

Seasoned software developers always seek to reduce the size of the code base through refactoring and the reuse of functionality in libraries, but with the never-ending demand for more features and intelligence in every product, it is clear that the amount of software in embedded devices will only grow. This also necessarily implies that there will be more bugs and vulnerabilities as well.

From this, it should be clear that having a way to deploy bug and security fixes, as well as new features, remotely is an obvious requirement for embedded devices with some level of intelligence, especially if they are network-connected.

This article goes further into specific requirements and solution designs on deploying software updates to embedded devices, in particular, embedded Linux.

The State of Software Updates in the Embedded Industry

As part of the work on Mender, we are conducting interviews with many software engineers and teams in order to better understand how software updates are being done in embedded products today. In one series of these interviews, we spoke with 30 embedded software engineers and the main takeaways are described below.

In the first question, we simply asked if software updates are being deployed to their embedded products today and, if so, which tools were used (Figure 2).

45.5% of the respondents said that updates were never being deployed to their products. Their only way to get new software into customers' hands was to manufacture hardware with the new software and ship the hardware to the customers.

Roughly the other half, 54.5%, said that they did have a way to update their embedded products, but the method was built in-house. This also

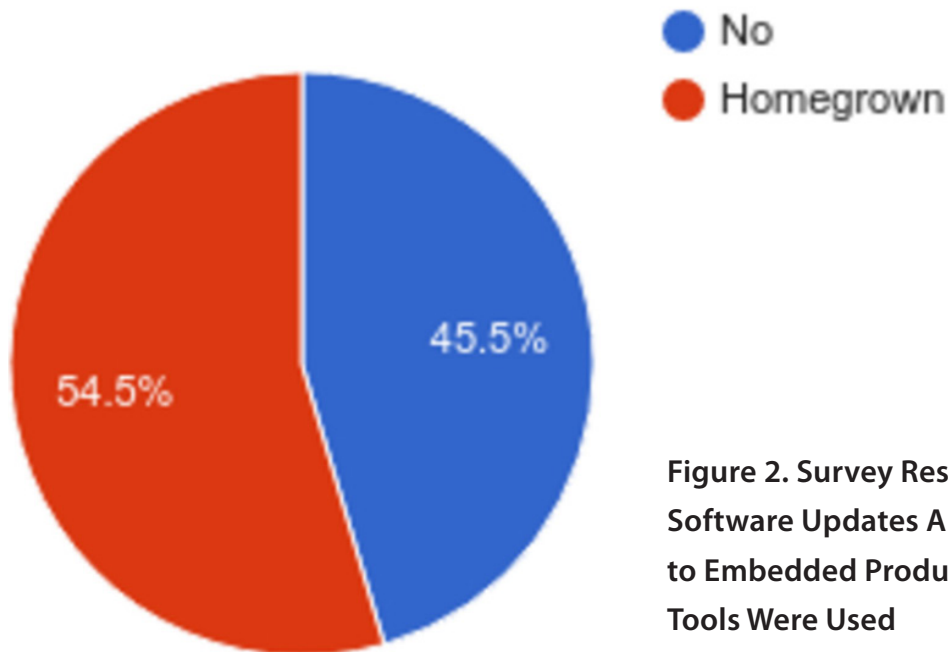


Figure 2. Survey Response to Whether Software Updates Are Being Deployed to Embedded Products, and If So, Which Tools Were Used

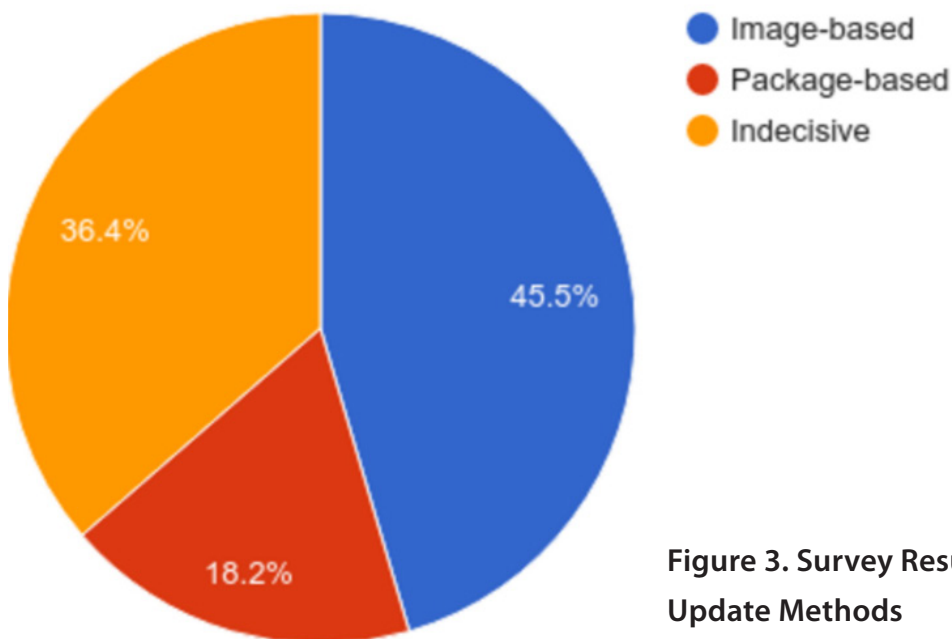


Figure 3. Survey Results on Software Update Methods

includes local updates, where a technician would need to go to a device physically and update the software from external media, such as a USB stick. Yet another category was devices enabled for remote updates, but where you could update only one at the time, essentially precluding any mass updates. Finally, some had the capability to deploy mass updates to

all devices in a highly automated fashion.

One of the key findings here was that virtually nobody reused a third-party software updater—they all re-invented the wheel!

Second, we asked what the preferred method of deploying software updates is (Figure 3). You can broadly classify embedded updaters into image- or package-based. Image-based updaters will work on the block level and can replace an entire partition or storage device. Package-based updaters work on the file level, and some of them, like RPM/YUM, are well known in Linux desktop and server environments as well.

Image-based updaters have, in general, a clear preference in the embedded space. The main reason for this is that they typically provide *atomicity* during the update process. Atomicity means that 1) an update is always either fully applied or not at all, and 2) no other component except the software updater can ever see a partial update. This property is very important for embedded updaters, because embedded devices could lose power or be rebooted at any time, and losing power in the middle of the update process should not lead to the device becoming bricked or unusable. The other stated key advantage of image-based updaters was *consistency across devices*, meaning that you can more confidently rely on behavior in test environments being the same as in production because of the 1:1 software copy.

Package-based approaches generally suffer from not being able to implement atomic updates, but they have some advantages as well. The installation time of an update is shorter, and the amount of bandwidth used also can be smaller than for image-based updates. Finally, since many develop their homegrown updater and already have packages from their build system, package-based update systems are generally faster to develop from scratch.

The Embedded Environment

People familiar with Linux desktop and server systems might ask why we are not just using the same tools and processes that we know from these systems, including package managers (such as rpm, dpkg), VMs and containers to carry out software updates. To understand this, it is important to see in which aspects an embedded device is *different* with

regards to applying software updates.

Unreliable Power I already touched on this property of an embedded system, and this is a widely known issue: an embedded device can, in general, lose power at any time. For example, a smart portable audio system can be unplugged as it is moved around in the house. The battery of a portable GPS could run out or become unreliable in certain weather conditions. In-vehicle-infotainment systems in cars can lose power intermittently as the car is started or stopped. This issue is amplified in battery-powered devices, as the update process itself can consume significant battery and cause the power to run out.

Embedded systems must be designed in such a way that they will tolerate power failure at any given time. The lack of power reliability is in strong contrast to typical data-center environments, where multiple redundant power systems will ensure that power is never lost.

Unreliable Network Embedded devices typically are connected using some kind of wireless technology. Although Wi-Fi is used in some devices, it is more common to use wireless standards that have longer range but lower data rates, for example 3G, LoRa, Sigfox and protocols based on IEEE 802.15.4 (low-rate wireless personal area networks).

It is tempting to assume that high-speed wireless networks will be generally adopted by embedded devices as technology evolves, just like what happened with smartphones where you can now stream YouTube videos in high resolution. However, keep in mind that the use cases for smartphones and typical embedded devices always will be very different. For example, an agricultural device that measures and optimizes crop yield needs a high amount of connectivity and should work even in places where there is no 3G coverage. In addition, the amount of data that needs to be sent is very low—perhaps just a few data points per day on the temperature and moisture measurements of the earth. So one should rather assume that embedded devices, especially industrial ones, always will have a limited network data rate.

In addition, wireless networks have frequent and intermittent connectivity loss—for example, when the device is moved to an area with low coverage, like underground.

Although low data rate and intermittent connectivity can be difficult

design issues, they usually are easy to identify once something is not right.

Security issues over public wireless networks are much more implicit and difficult to expose. In the context of software updates, there are countless examples of homegrown updaters that do not properly authenticate the update, allowing an attacker to inject malicious code while the update is taking place.

Expensive Physical Access Once a large-scale issue that cannot be fixed remotely occurs, the cost of remediating it is typically very high. The reason is that embedded devices are typically widely distributed geographically.

For example, a manufacturer of smart energy grid devices can install these devices in thousands of homes in several countries. If there is a critical issue with an update to the Linux kernel that cannot be fixed remotely, the cost of either sending a service technician to all those homes or asking customers to send devices back to the vendor can be prohibitive.

The 2015 Fiat-Chrysler Jeep Cherokee security breach offers a recent real-world example of wide-scale recalls. In this case, 1.4 million cars were recalled. The cost of repairing this issue was certainly in the hundred-million dollar range, perhaps even billions.

Five-to-Ten-Year Device Lifetime Technology moves very fast, and it's typical to replace common consumer electronics devices like smartphones and laptops every two to three years.

However, more expensive consumer devices like high-end audio systems and TVs are replaced less frequently. Industrial devices that do not directly interact with humans typically have even longer lifetimes. For example, robots used on factory floors or energy grid devices easily can reach a ten-year lifetime.

In conclusion, in the embedded environment, people need to be very wary of the risk of "bricking" devices. Not only can this easily happen due to the power and network properties, but it is also a very expensive situation from which to recover.

Key Criteria for Embedded Software Updaters

Now that you are more familiar with the embedded environment, let's consider the implications this has for an embedded software updater.

Robust and Secure

As you've seen, both the power and the network can be very unreliable and insecure in an embedded environment. An embedded updater must have a couple properties and features in order to tackle these challenges sufficiently.

Atomic Updates The database industry is very familiar with the concept of atomic transactions—the “A” in ACID, where a set of operations either all complete or none of them complete. The classic example for the need for this requirement in database theory is with online transactions. When one user transfers money to another, deducting money from one account should occur only if you also successfully add the money to the other account.

This same property is very important for embedded updaters, in order to handle intermittent update errors like sudden power loss. For an embedded updater, the atomic property can be defined in two parts:

- An update is always either *completed fully or not at all*.
- *No software component* (besides the updater) ever sees a *partially installed update*.

You can see that common ways of deploying software updates in the desktop environment do not meet this atomicity requirement. For example, while you are installing an rpm package, many files are written and modified across the filesystem, and they would be in an inconsistent and potentially non-recoverable state if you suddenly unplugged your desktop during the installation—the application being updated probably wouldn't start at all.

Consistent Deployments An important approach to mitigate the risks of bricking devices is to test new software updates extensively before releasing them into production. However, in order to rely on test results, you need a test environment that is as identical as possible to the production environment. It is a classic problem in general operations, be it for embedded devices or data centers, that the test environment diverges from production, so that changes work well in the test environment but cause significant downtime when released

to production. This is one of the reasons why full-image updates are so prevalent in the embedded space. If your entire root filesystem is the same, block by block, in the test and production environments, then there are guaranteed similarities. Contrast this to a deployment using rpm packages, which may depend on libraries that have different versions, or patches, on the test and production environments, and maybe even across the production environments as well. Over time, such a design typically will lead to production deployments that fail for reasons that are inconsistent and hard to diagnose.

Authenticity Checks before Updates From a security perspective, it is very important to know whether software comes from an authorized source or whether an attacker could have injected malicious software into the update. There have been countless cases where embedded devices are simply broadcasting their desire to install an update, and anyone who responds would be able to inject the software of their choosing into the device.

A basic approach to ensuring a level of authenticity is to leverage in-transit security protocols like TLS. If done correctly, this will ensure that the update cannot be modified while in transit from an update server to the device.

However, a more robust end-to-end approach is to embed cryptographic authenticity metadata as part of the update itself. Typically a form of *code signing* is employed, where digital signatures are created by an authority and verified at the device.

One of the key advantages of code signing over solely relying on in-transit security is that the authority that signs the update can be decoupled from the server that hosts it. For example, someone in the QA department could sign an update offline. This reduces the attack surface in cases where the update server gets compromised, because an attacker can still deploy only updates that have been signed by the QA department.

For performance-sensitive devices, cryptographic mechanisms like Message Authentication Code (MAC) or Elliptic Curve signatures should be considered, as they provide much more efficient verification than RSA or DSA at the same level of security.

Sanity Checks after Updates Embedded devices are typically

single-purpose and run only one main application, although in some cases, they could run several. In either instance, it's important to check the health of such applications after deploying an update. Are they running? Do they have network access? Can the user interact with them successfully on the device?

A software update should not be considered successful just because the device boots; there should be a way to integrate custom application sanity checks as well. Finally, a critical check that should be covered by the updater generically is this: *Is it possible to deploy another update?*

If any of these checks fail, the updater should have the capability to *roll back* to the previous known-working software, so that downtime is avoided while the issue is being diagnosed and resolved.

The general workflow for deploying software updates is shown in Figure 4.

Integration with Existing Development Workflow If you are one person starting from scratch with an embedded/IoT project, you likely can choose all the tools and processes you like the best. However, once several people are collaborating on the same project, and in particular, if

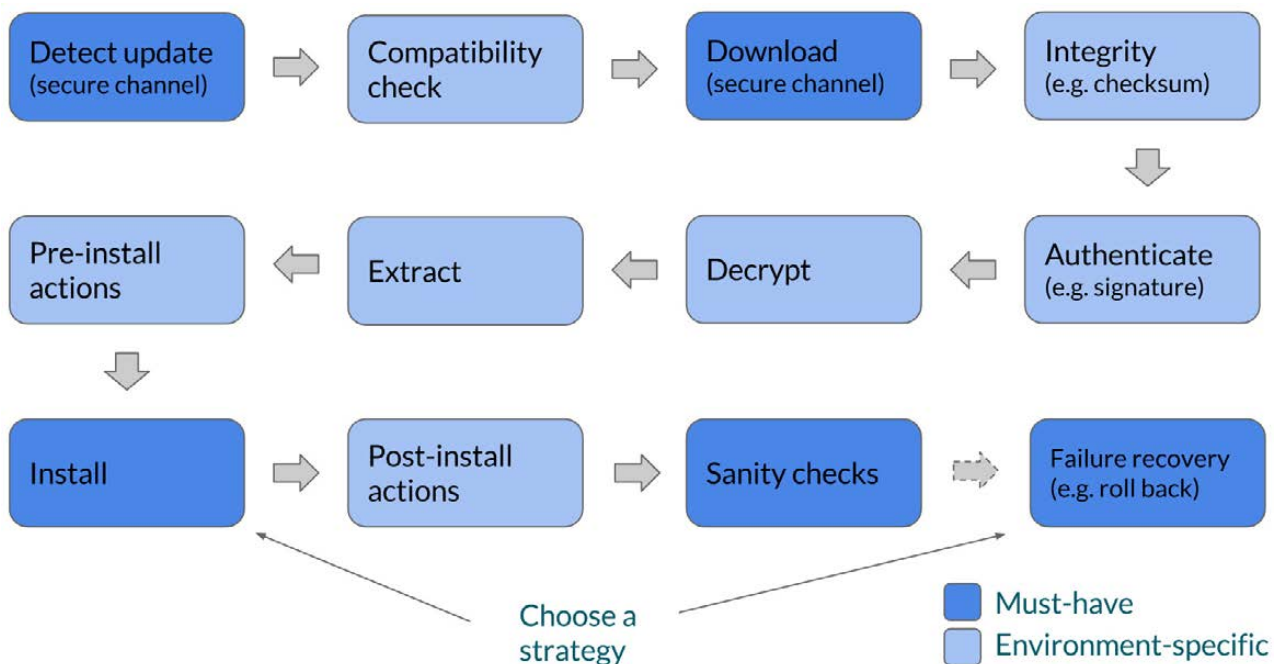


Figure 4. General Workflow for Deploying Software Updates

there is a product already being developed before software updates were taken into account, it is very important that the software update process integrates well with the development workflow.

At first glance, this may look like a strange criteria for an updater, but many approaches to software updates require a full replacement of existing development workflows. Commercial updater tools more often than not are offered as part of a “platform”, where the updater is bundled together with a full device OS, a cloud back end and other device management features. For existing products, this can pose a significant challenge, because the device OS needs to be replaced, potentially also together with the build system, version control and associated QA processes.

For homegrown updaters, this criteria is typically implicitly taken into account, because teams tend to start with what they have and see what is the shortest path to develop and integrate an updater into it. Since existing build systems tend to output packages like rpm or opkg easily, this is an approach that integrates well and is chosen by many homegrown updaters. However, package-based updates have significant drawbacks with respect to lack of robustness, as I discussed earlier.

Bandwidth As I mentioned previously, embedded devices typically are connected with some kind of low data rate wireless connection. An update process that requires less bandwidth will be favorable over one that takes more, simply because it would cost less and take less time to deploy an update.

Compression is the first feature to look at in order to reduce bandwidth, as this could cut the size of the update in half or more, depending on type and compressibility of the update. There is also a variety of delta-based update mechanisms that could be employed to reduce bandwidth usage further.

Downtime during Update While an update is being deployed, it is desirable to have as little downtime on the device as possible. How much downtime is acceptable is clearly dependent on the use case of the embedded device. Is it part of the power grid that must function 24x7, or is it a consumer audio system that isn't used at night?

The method for deploying updates impacts the required downtime the most. For example, for full image updates, it's possible to deploy the update from a maintenance mode or use a dual-A/B rootfs approach.

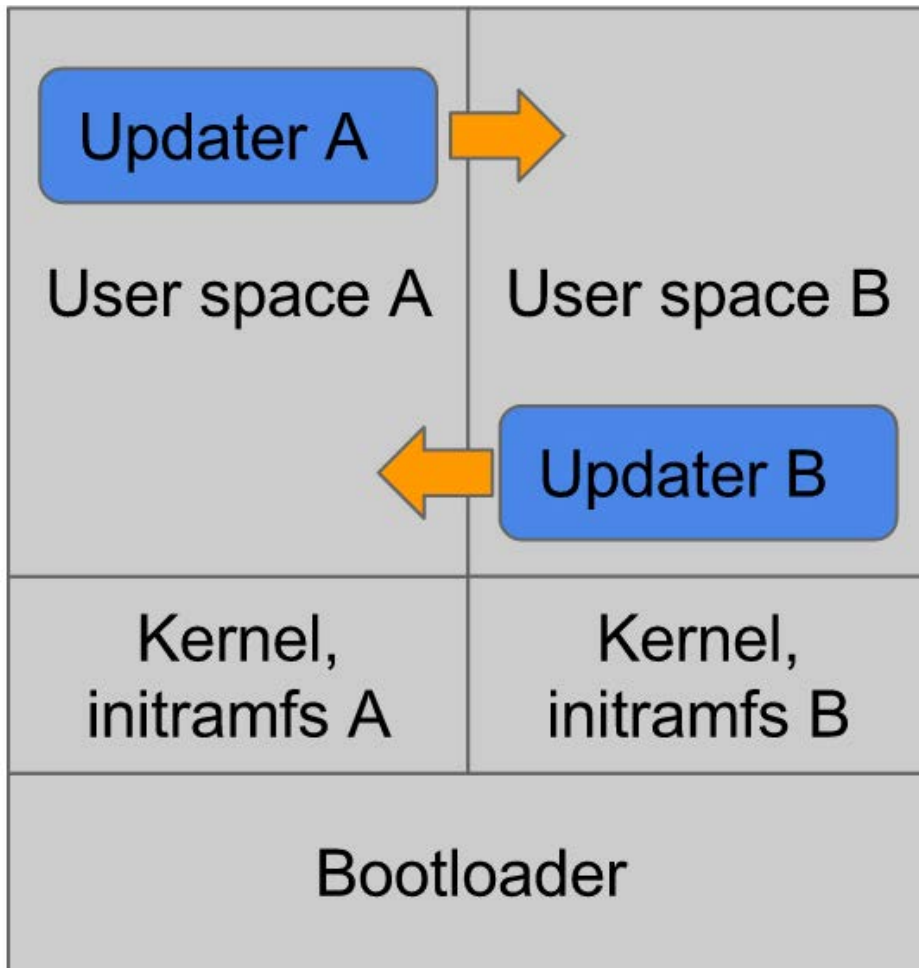


Figure 5.
Dual-A/B rootfs
Partition Update
Design

The maintenance-mode approach works by rebooting into a maintenance partition, installing the update to the root filesystem partition and then rebooting into the root filesystem partition again; the device is unusable for all of this period. In a dual-A/B rootfs approach, the update is installed to the inactive root filesystem while the device can continue to be used. The downtime in this case is only during the reboot into the updated (previously inactive) partition. The dual-A/B rootfs partition update design is shown in Figure 5.

Deployment Management As you can see, many design choices and trade-offs need to be made on the device side for an embedded updater client.

However, once an updater client is installed and working on embedded devices, the problem of managing all those clients becomes apparent. How can a new update be installed on 1,000 of these embedded devices? Which version of the software are they running?

How do you know if an update is installed successfully everywhere, and is there a log for failed updates?

These use cases typically are handled with an update management server, so that updates can be managed across a device fleet.

Conclusion

Many design trade-offs need to be considered in order to deploy software updates to IoT devices. Although historically most teams have decided to implement their homegrown updaters, the recent appearance of several open-source software updaters for embedded Linux means that we should be able to stop re-inventing the wheel. ■

Eystein Stenberg has more than seven years of experience in security and systems management software and has spoken at various conferences. You can reach him at eystein@mender.io.

Resources

SWUpdate (<https://sbabic.github.io/swupdate>) is a very flexible client-side embedded updater for full image updates, licensed GPL version 2.0+.

Mender (<https://www.mender.io>), the project the author of this article is involved in, focuses on ease of use and consists of a client updater and management server with a UI and is licensed under Apache License 2.0.

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

[RETURN TO CONTENTS](#)



Where every interaction matters.

break down your innovation barriers

power your business to its full potential

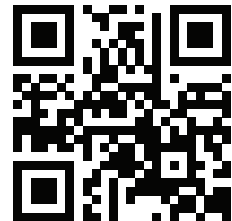
When you're presented with new opportunities, you want to focus on turning them into successes, not whether your IT solution can support them.

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.

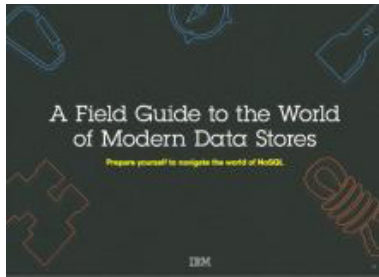
Unsurpassed performance and reliability help build your business foundation to be rock-solid, ready for high growth, and deliver the fast user experience your customers expect.

Want more on cloud?

Call: 844.855.6655 | go.peer1.com/linux | [View Cloud Webinar:](#)



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



A Field Guide to the World of Modern Data Stores

There are many types of databases and data analysis tools to choose from when building your application. Should you use a relational database? How about a key-value store? Maybe a document database? Is a graph database the right fit? What about polyglot persistence and the need for advanced analytics?

If you feel a bit overwhelmed, don't worry. This guide lays out the various database options and analytic solutions available to meet your app's unique needs.

You'll see how data can move across databases and development languages, so you can work in your favorite environment without the friction and productivity loss of the past.

Sponsor: IBM

> <https://geekguide.linuxjournal.com/content/field-guide-world-modern-data-stores>



Why NoSQL? Your database options in the new non-relational world

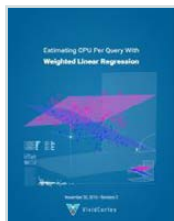
The continual increase in web, mobile and IoT applications, alongside emerging trends shifting online consumer behavior and new classes of data, is causing developers to reevaluate how their data is stored and managed. Today's applications require a database that is capable of providing a scalable, flexible solution to efficiently and safely manage the massive flow of data to and from a global user base.

Developers and IT alike are finding it difficult, and sometimes even impossible, to quickly incorporate all of this data into the relational model while dynamically scaling to maintain the performance levels users demand. This is causing many to look at NoSQL databases for the flexibility they offer, and is a big reason why the global NoSQL market is forecasted to nearly double and reach USD3.4 billion in 2020.

Sponsor: IBM

> <https://geekguide.linuxjournal.com/content/why-nosql-your-database-options-new-non-relational-world>

Estimating CPU Per Query With Weighted Linear Regression



Your database server is suddenly using a lot of CPU resources. Quick, what caused it? This is a familiar question for engineers of all persuasions. And it's often impossible to answer.

There are good reasons why it's hard to figure out what consumes resources like CPU, IO, and memory in a complex piece of software such as a database. The first problem is that most database server software doesn't offer any way to measure or inspect that type of performance data. The database server isn't observable. This problem arises in turn from the complexity of the database server software and the way it does its work, which actually precludes measuring resource consumption accurately!

Author: Baron Schwartz

Sponsor: VividCortex

> <https://geekguide.linuxjournal.com/content/estimating-cpu-query-weighted-linear-regression>



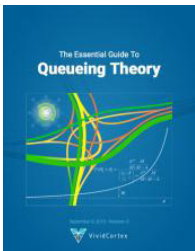
Database Performance Monitoring Buyer's Guide

More and more companies have begun to recognize database performance management as a vital need. Despite its widespread importance, good database performance management requires specialized expertise with custom approaches--yet all too often, organizations rely on one-size-fits-all solutions that theoretically "check the box" but in practice do little or nothing to help them find or prevent database-related outages and performance problems.

This buyer's guide is designed to help you understand what database management really requires, so your investments in a solution provide the greatest possible ultimate value.

Sponsor: VividCortex

> <https://geekguide.linuxjournal.com/content/database-performance-monitoring-buyer%E2%80%99s-guide>



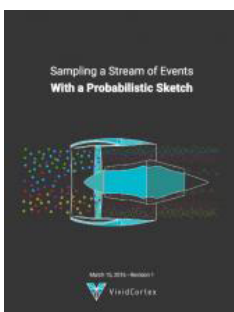
The Essential Guide To Queueing Theory

Whether you're an entrepreneur, engineer, or manager, learning about queueing theory is a great way to be more effective. Queueing theory is fundamental to getting good return on your efforts. That's because the results your systems and teams produce are heavily influenced by how much waiting takes place, and waiting is waste. Minimizing this waste is extremely important. It's one of the biggest levers you will find for improving the cost and performance of your teams and systems.

Author: Baron Schwartz

Sponsor: VividCortex

> <https://geekguide.linuxjournal.com/content/essential-guide-queueing-theory>



Sampling a Stream of Events With a Probabilistic Sketch

Stream processing is a hot topic today. As modern Big Data processing systems have evolved, stream processing has become recognized as a first-class citizen in the toolbox. That's because when you take away the how of Big Data and look at the underlying goals and end results, deriving real-time insights from huge, high-velocity, high-variety streams of data is a fundamental, core use case. This explains the explosive popularity of systems such as Apache Kafka, Apache Spark, Apache Samza, Apache Storm, and Apache Apex—to name just a few!

Author: Baron Schwartz

Sponsor: VividCortex

> <https://geekguide.linuxjournal.com/content/sampling-stream-events-probabilistic-sketch>

How to Fix the Edge

If silo builders control the edge, the distributed future can't happen.

PREVIOUS

◀ Feature: Key Considerations for Software Updates for Embedded Linux and IoT

In December 2016, Peter Levine (<http://peter.a16z.com>) of the venture firm Andreessen Horowitz published a post with a video titled "Return to the Edge and the End of Cloud Computing" (<http://a16z.com/2016/12/16/the-end-of-cloud-computing>). In it, he outlines a pendulum swing between centralized and distributed computing that goes like this:

- Mainframe/Centralized/1960-1970 ↔ Client-Server/Distributed/1980-2000
- Mobile-Cloud/Centralized/2005-2020 ↔ Edge Intelligence/Distributed/2020-

He says the "total addressable market" in that next pendulum swing will include the Internet of Things, with trillions of devices, starting with today's



DOC SEARLS

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

cars and drones. He also says machine learning will be required to “decipher the nuances of the real world”.

Thanks to bandwidth and latency issues, most of this will have to happen at end points, on the edge, and not in central clouds. Important information still will flow to those clouds and get processed there for various purposes, but decisions will happen where the latency is lowest and proximity highest: at the edges. Machines will make most of the data they gather (and is gathered for them). That’s because, he says, humans are bad at many decisions that machines do better, such as in driving a car. Peter has a Tesla and says “My car is a much better driver than I am.” In driving for us, machine-learning systems in our things will “optimize for agility over power”. Systems in today’s fighter aircraft already do this for pilots in dogfights. They are symbiotes of a kind, operating as a kind of external nervous system for pilots, gathering data, learning and reacting in real time, yet leaving the actual piloting up to the human in the cockpit. (In dogfights, pilots also do not depend on remote and centralized clouds, but they do fly through and around the non-metaphorical kind.)

The learning curve for these systems consists of three verbs operating in recursive loops. The verbs are *sense*, *infer* and *act*. Here’s how those sort out.

Sense Data will enter these loops from sensors all over the place—cameras, depth sensors, radar, accelerometers. Already, he says, “a self-driving car generates about ten gigabytes of data per mile”, and “a Lytro (<https://www.lytro.com>) camera—a data center in a camera—generates 300 gigabytes of data per second.” Soon running shoes will have sensors with machine-learning algorithms, he adds, and they will be truly smart, so they can tell you, for example, how well you are doing or ought to do.

Infer The data from our smart things will be mostly unstructured, requiring more machine learning to extract relevance, do task-specific recognition, train for “deep learning”, and to increase accuracy and automate what needs to be automated. (This will leave the human stuff up to the human—again like the fighter pilot doing what only he or she can do.)

Act As IoT devices become more sophisticated, we’ll have more

data accumulations and processing decisions, and in many (or most) cases, machines will make ever-more-educated choices about what to do. Again, people will get to do what people do best. And, they'll do it based on better input from the educated things that also do what machines do best on their own.

Meanwhile, the old centralized cloud will become what he calls a "training center". Since machine learning needs lots of data in order to learn, and the most relevant data comes from many places, it only makes sense for the cloud to store the important stuff, learn from everything everywhere, and push relevant learnings back out to the edge. Think of what happens (or ought to happen) when millions of cars, shoes, skis, toasters and sunglasses send edge-curated data back to clouds for deep learning, and the best and most relevant of that learning gets pushed back out to the machines and humans at the edge. Everything gets smarter—presumably.

His predictions:

- Sensors will proliferate and produce huge volumes of geo-spatial data.
- Existing infrastructure will back-haul relevant data while most computing happens at edges, with on-site machine learning as well.
- We will return to peer-to-peer networks, where edge devices lessen the load on core networks and share data locally.
- We will have less code and more math, or "data-centric computing"—not just the logical kind.
- The next generation of programmers won't be doing just logic: IF, THEN, ELSE and the rest.
- We'll have more mathematicians, at least in terms of talent required.
- Also expect new programming languages addressing edge use cases.
- The processing power of the edge will increase while prices decrease,

All that is good as far as it goes, which is toward what companies will do. But what about the human beings who own and use this self-educating and self-actualizing machinery?

which happens with every generation of technology.

- Trillions of devices in the supply chain will commoditize processing power and sensors. The first LIDAR for a Google car was \$7,000. The new ones are \$500. They'll come down to 50 cents.
- The entire world becomes the domain of IT. "Who will run the fleet of drones to inspect houses?" When we have remote surgery using robots, we also will need allied forms of human expertise, just to keep the whole thing running.
- We'll have "consumer-oriented applications with enterprise manageability."

His conclusion: "There's a big disruption on the horizon. It's going to impact networking, storage, compute, programming languages and, of course, management."

All that is good as far as it goes, which is toward what companies will do. But what about the human beings who own and use this self-educating and self-actualizing machinery? Experts on that machinery will have new work, sure. And all of us will to some degree become experts on our own, just as most of us are already experts with our laptops and mobile devices. But the IoT domain knowledge we already have is confined to silos. Worse, silo-ization of smart things is accepted as the status quo.

Take for example "Google Home vs. Amazon Echo—a Face-Off of Smart Speakers" by Brian X. Chen in *The New York Times* (https://www.nytimes.com/2016/11/04/technology/personaltech/google-home-vs-amazon-echo-a-face-off-of-smart-speakers.html?_r=1).

Both Google Home and Amazon Echo are competitors in the “virtual assistant” space that also includes Apple’s Siri and Microsoft’s Cortana. All are powered by artificial intelligence and brained in the server farms of the companies that sell them. None are compatible with each other, meaning substitutable. And all are examples of what Phil Windley called The Compuserve of Things in a blog post by that title exactly three years ago (http://www.windley.com/archives/2014/04/the_compuserve_of_things.shtml). His summary:

On the Net today we face a choice between freedom and captivity, independence and dependence. How we build the Internet of Things has far-reaching consequences for the humans who will use—or be used by—it. Will we push forward, connecting things using forests of silos that are reminiscent of the online services of the 1980s, or will we learn the lessons of the internet and build a true Internet of Things?

If progress continues on its current course, the distributed future Peter Levine projects will be built on the forest-of-silos Compuserve-of-things model we already have. Amazon, Apple, Google and Microsoft are all silo-building Compuserves that have clearly not learned the first lesson of the internet—that it was designed to work for everybody and everything, and not just so controlling giants can fence off territories where supply chains and customers can be held captive. For all their expertise in using the internet, these companies are blindered to the negative externalities of operating exclusively in their self interest, oblivious to how the internet is a tide that lifts all their economic and technological boats. In that respect, they are like coal and oil companies: expert at geology, extraction and bringing goods to market, while paying the least respect to the absolute finitude of the goods they extract from the Earth and to the harms that burning those goods cause in the world.

But none of that will matter, because the true Internet of Things is the only choice we have. If all the decisions that matter most, in real time (or close enough), need to be made at the edge, and people there need to be able to use those things expertly and casually, just like today’s fighter pilots, they’ll need to work for us and not just

their makers. They'll be like today's cars, toasters, refrigerators and other appliances in two fundamental ways: they'll work in roughly the same ways for everybody, so the learning curve isn't steep; and they'll be substitutable. If the Apple one fails, you can get a Google one and move on.

For an example, consider rental cars. They're all a bit different, but you know how to drive all of them. Sure, there are glitches. Every Toyota I rent plays Edith Piaf (from somewhere in my music collection) as soon as I plug in my phone to the car's USB jack. Other car brands have their own dashboard quirks. (Last month, my visiting sister rented a Chrysler 200, which had the stupidest and least useful climate control system I've ever seen, but it was easy for both of us to drive.)

Also, as the ever-more distributed world gets saturated by smart things on Peter Levine's model, we will have more need to solve existing problems that get worse every day in present time. Some examples:

- Too many login and password combinations, plus the fact that we still need logins and passwords at all. Geez, it's 2017. We can do better than that.
- Too many ways to message each other. Last I counted, Apple's App Store had something like 170 different messaging apps, and Google Play had more than a hundred. The only standard we ever had for bridging them all was XMPP, originally called Jabber, which I advocated mightily in *Linux Journal*, back around the turn of the millennium. (See "The Message" at <http://www.linuxjournal.com/article/4112>, "Talking Jabber" at <http://www.linuxjournal.com/article/4113> and "Jabber Asks the Tough Question" at <http://www.linuxjournal.com/article/5631>.) For whatever reason, XMPP stalled. (Never mind why. Make another standard protocol everyone can adopt.)
- Too many contacts and too few ways of connecting them to login/password management, to-do lists, calendars or other ways of keeping records.

- Calendar and contact apps silo'd into the bowels of Apple, Microsoft, Google and others, with too few compatibilities.

To solve all these problems, you need to start with the individual: the individual device, the individual file, the individual human being.

If you start with central authority and central systems, you make people and things subordinate dependents, and see problems only silos can solve. All your things and people will be captive, by design. No way around it.

Why do I pose this challenge here? Two reasons: 1) because Linux answered the same challenge in the first place, and it can again; and 2) because Linux geeks have the best chance of both grokking the challenge and doing something about it. ■

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

RETURN TO CONTENTS

ADVERTISER INDEX

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

ADVERTISER	URL	PAGE #
DrupalCon Baltimore	https://events.drupal.org/baltimore2017	39
Drupalize.me	http://drupalize.me	111
HPC Wallstreet	http://www.flagmgmt.com/linux	29
Peer 1 Hosting	http://go.peer1.com/linux	101
Silicon Mechanics	http://www.siliconmechanics.com	13
SPTechCon	http://www.sptechcon.com	53
SUSE	http://suse.com/storage	7

ATTENTION ADVERTISERS

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

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!

