LINUX JOURNAL™

Since 1994: The Original Magazine of the Linux Community

Create a Network Backup Server with the Banana Pi

SEPTEMBER 2017 | ISSUE 281
http://www.linuxjournal.com

FULL STACK DEVELOPMENT PROJECT

+

MANAGE YOUR NETWORK WITH ANSIBLE AND SSH

THINKING CONCURRENTLY: A LOOK AT MODERN NETWORKING

GUEST EOF: HEIRLOOM SOFTWARE

Use Pandoc and Panflute for Technical Publishing

WATCH: ISSUE OVERVIEW

An Architect's Guide: Linux in the Age of Containers

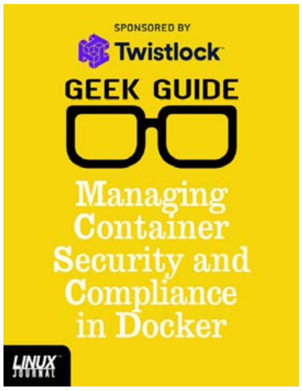
Author:
Sol Lederman

Sponsor:
SUSE

SQL Server on Linux

Author:
Reuven M. Lerner

Sponsor:
SUSE



Managing Container Security and Compliance in Docker

Author:
Petros Koutoupis

Sponsor:
Twistlock

Harnessing the Power of the Cloud with SUSE

Author:
Petros Koutoupis

Sponsor:
SUSE

DevOps for the Rest of Us

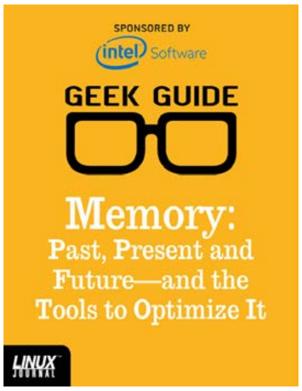
Author:
John S. Tonello

Sponsor:
Puppet

An Architect's Guide: Linux for Enterprise IT

Author:
Sol Lederman

Sponsor:
SUSE



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

Author:
Petros Koutoupis

Sponsor:
Intel

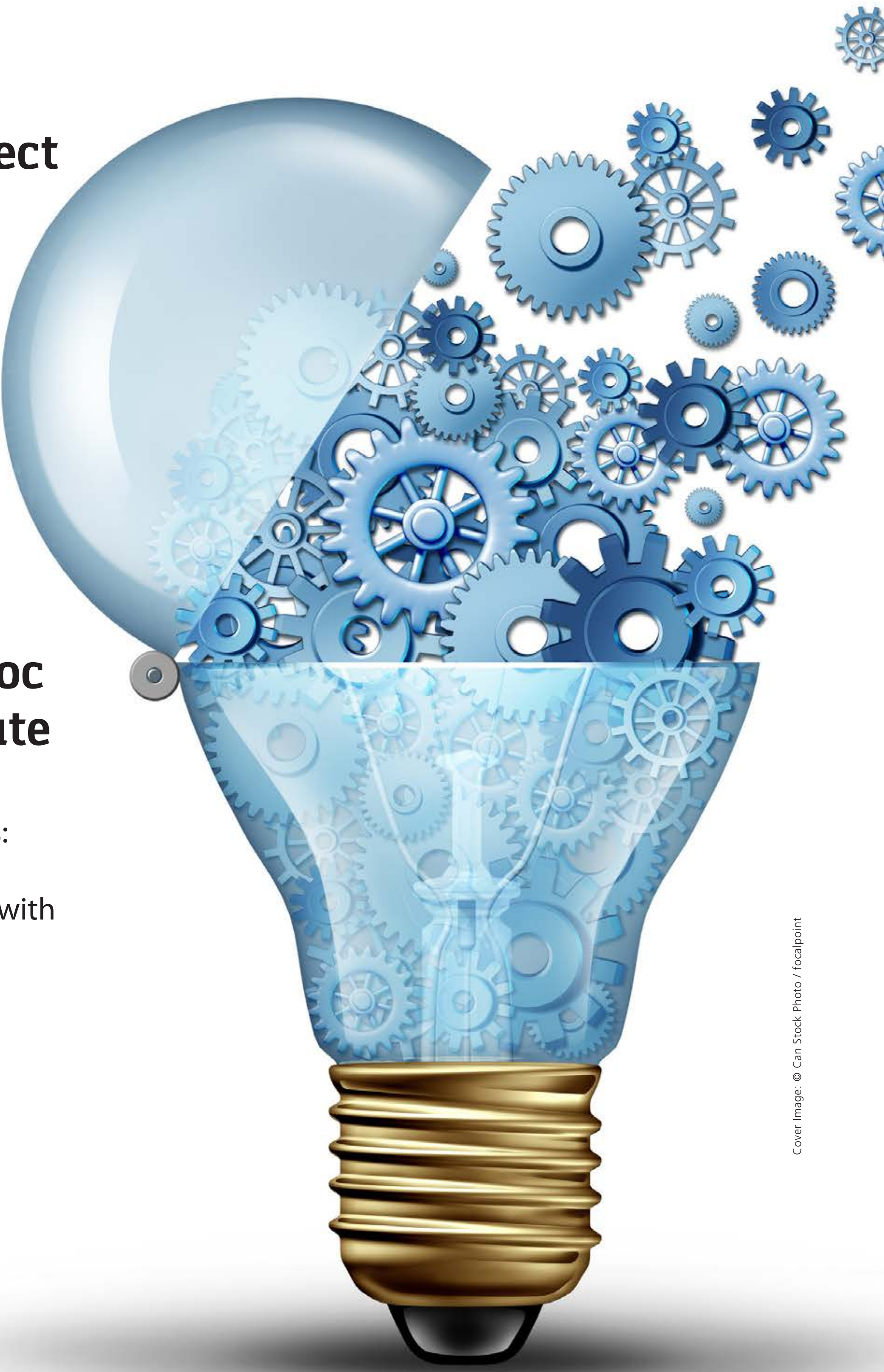
Cloud-Scale Automation with Puppet

Author:
John S. Tonello

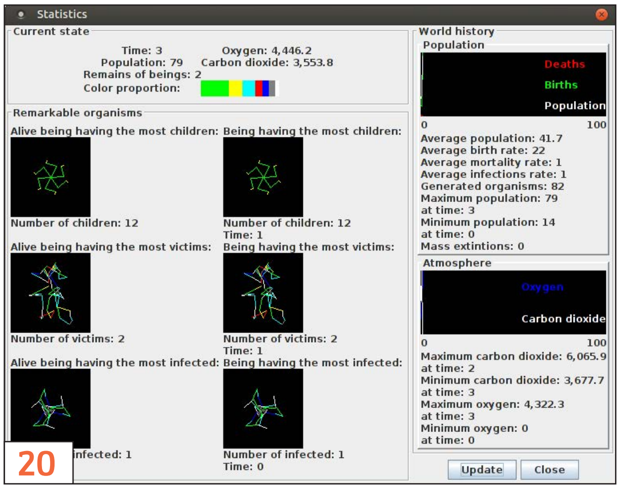
Sponsor:
Puppet

CONTENTS

SEPTEMBER 2017
ISSUE 281

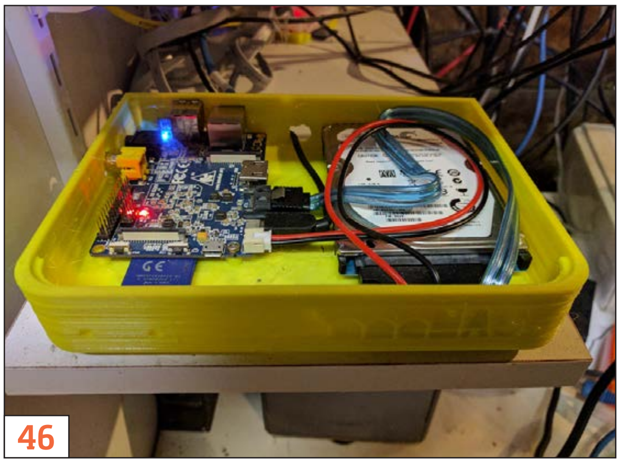


Cover Image: © Can Stock Photo / focalpoint



```
Statistics
Current state
        Time: 3          Oxygen: 4,446.2
    Population: 79    Carbon dioxide: 3,553.8
  Remains of beings: 2
  Color proportion:

Remarkable organisms
Alive being having the most children:  Being having the most children:
Number of children: 12                 Number of children: 12
                                       Time: 1
Alive being having the most victims:   Being having the most victims:
Number of victims: 2                   Number of victims: 2
                                       Time: 1
Alive being having the most infected:  Being having the most infected:
        infected: 1                    Number of infected: 1
                                       Time: 0

World history
Population
0                    100
Average population: 41.7
Average birth rate: 22
Average mortality rate: 1
Average infections rate: 1
Generated organisms: 82
Maximum population: 79
at time: 3
Minimum population: 14
at time: 0
Mass extintions: 0
Atmosphere
0                    100
Maximum carbon dioxide: 6,065.9
at time: 2
Minimum carbon dioxide: 3,677.7
at time: 3
Maximum oxygen: 4,322.3
at time: 3
Minimum oxygen: 0
at time: 0
```

20



46

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

LINUX JOURNAL

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

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

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

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

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.

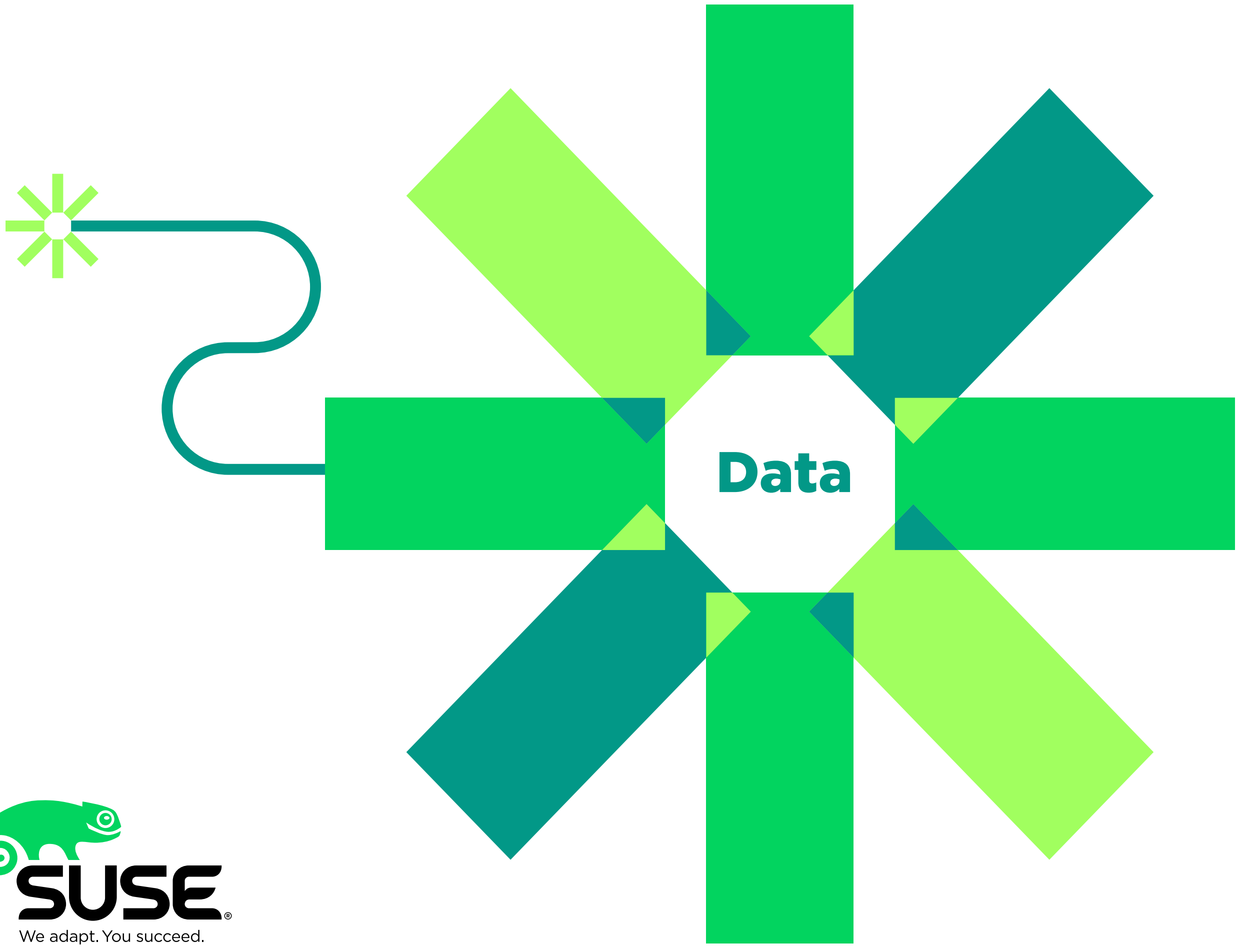
STORAGE REDEFINED:

You can~~not~~ 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



SUSE
We adapt. You succeed.

Soup to Nuts

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 IRC channel on Freenode.net.

One of my favorite things about Linux is that it has become not only the platform of choice for many projects, but it also tends to inspire an entire ecosystem of open-source solutions. It's almost as if, when the operating system is free, it only makes sense that the things built on top of it are free too. In this September issue, you'll see the all-encompassing Linux environment in action.

First up is Reuven M. Lerner, who is discussing scaling applications to handle simultaneous connections. Anyone who has waited in line to get into an amusement park knows that the only way massive throughput works is if there are multiple "lanes" working together at the same time. That concept is demonstrated perfectly in web applications, and Reuven explains how.

Dave Taylor follows with a bit of advancement on scripting dice rolls. He described a simple dice game last month, but what about dice that have more than six sides? Dave's article will expand your dice knowledge while making you better scripters at the same time.

Backups are a passion of mine, and thankfully it's a topic Kyle Rankin and I agree is vitally important. This month, Kyle demonstrates a low-power way to create a fast and effective backup system using a Banana Pi. A similar thing could be done with Raspberry Pi, but the Banana is a bit faster, which is ideal for backups. I help make things faster this month too, but rather than dealing with backups,

VIDEO: Shawn Powers runs through the latest issue.

I continue my series on Ansible. I love using Ansible, because it thinks like a system administrator, which happens to be the way I think too!

I mentioned earlier that Linux is great for "Soup to Nuts"-type projects. Kyle demonstrated that with his backup system, but John S. Tonello goes even further by explaining the "Full Stack" concept. Being able to develop an application is a desirable trait for many employers, but if you can handle every aspect of development and deployment, it makes you (and your app) far more valuable. John explains how the Full Stack idea works by walking through the development of an app you can see in action today!

Lee Phillips addresses a topic that is near and dear to my heart—specifically, writing. Most people know documentation is that thing we all value, and yet so few of us do it well. Using Pandoc and Panflute, Lee walks through creating a publishing pipeline that takes most of the frustration out of technical writing, allowing you to focus on the actual writing itself. Anything that makes technical writing easier is a benefit to our entire community, so be sure to check out his article.

Linux is the perfect platform for experimentation, application and education. Whether you want to build a web application from the ground up or literally want to create an interactive application for sharing soup and nut recipes, Linux really shines. As always, we've included product announcements, reviews, tech tips and countless other useful bits of Linux-related information. We hope you enjoy the issue as much as we've enjoyed putting it together! ■

RETURN TO CONTENTS

PREVIOUS
Current_Issue.tar.gz

NEXT
Editors' Choice

diff -u

What's New in Kernel Development

Miroslav Lichvar recently submitted some patches to improve the accuracy of the **Linux system clock**. Nowadays, the Linux clock generally is synchronized periodically with internet sources like everything else, but it's still nice to have a high degree of accuracy during the time in between synchronizations.

John Stultz liked the code, but he wanted to include Miroslav's test suite in the kernel as well, so that other developers could make sure future patches didn't spoil system clock accuracy. And, this turned out to be the biggest point of contention in the discussion thread.

Miroslav felt that his test suite—which really had been written only for his own use—was ugly, breakable and just bad news all around. However, John felt that its value outweighed all those things, and that its presence in the kernel would encourage others to improve the suite over time.

Eventually, they compromised. Miroslav wrote a new, more robust and maintainable test suite that was slower, less precise and less predictable than the original, but that still essentially did the job. It was a rare case where the quality of a feature takes a back seat to a developer's need to be able to hold his head up.

Luis R. Rodriguez, the **firmware** maintainer, noticed that bugs were creeping into the kernel firmware code, just because the folks submitting patches weren't sure who to cc. Patches, thus, were getting lost in the vast ocean of linux-kernel email. The patches themselves often would be reviewed and approved, but without the eyeballs that really could ferret out any troublesome breakages. Luis wanted to start a new mailing list, just for firmware.

The idea went nowhere, fast. **Linus Torvalds** said:

> Boutique mailing lists are generally a *bad* thing. All it means is that there's an increasingly small "in group" that thinks that they generate consensus because nobody disagrees with their small boutique list, because nobody else even *sees* that small list. We should only have mailing lists if they really merit the volume, and are big enough that there are lots of users.

And, this judgment stood even though, as Luis pointed out, many device drivers had their own mailing lists, in spite of their having extremely low traffic. As **Greg Kroah-Hartman** put it, device drivers were not part of the core kernel infrastructure that everyone relied on. Firmware support, like other parts of the kernel, needed to be broadly scrutinized.

Of course, this was in conflict with Luis' original point—that patches were not being scrutinized enough because they'd get lost in the vastness of the main mailing list.

Even so, it seems that Linus, David, Greg and others want maintainers to find other solutions to that problem—perhaps by documenting a list of people to be cc'd on all patches within a given area

LINUX™ JOURNAL

At Your Service

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.

of code—or perhaps something else. But, they definitely seem to want to avoid diluting the main linux-kernel mailing list traffic if at all possible.

Containers are weird. The whole idea is to pretend that you've got a virtual system running on top of a Linux box, just by isolating certain resources on that box and having them all act like they're a box of their own. You even could create virtual systems that are crippled or enhanced in bizarre ways, just because they match the specific requirements of a crazy new project you've got in mind. At the same time, the virtualization process itself creates a wide range of feature constraints and security caveats that always must be navigated properly. And, the whole implementation of containers is something that has to happen gradually over time because it's so nuanced and wide-ranging throughout the full breadth of the kernel, so that Linux has been approaching better and better virtualization during a course of years.

So, when **David Howells** wanted to implement a standardized container "object" that would wrangle all aspects of containers into a single, easy-to-use set of functions, he encountered nearly universal resistance.

In theory, it's a great idea. All the little niggling details of containers can cause odd security relationships, strange communication requirements and odd filesystem structures. Trying to smooth all of that out and give it a clean face seems like a natural next step, as containers become more and more robust.

But, **James Bottomley**, **Jessica Frazelle**, **Aleksa Sarai** and others felt that David was smoothing things about a bit too much and making too many assumptions about the kind of strange uses one might have for containers.

Ultimately, folks like **Eric W. Biederman** started proposing alternatives to David's approach that would be more flexible, and the discussion ended up being less about David's patches and more about how to implement the features that the people raising their objections seemed to want.

Eventually, some sort of container management system probably will go into the kernel, though there are also projects like **Docker** that attempt to fill that role. But as David remarked at a few different points, any in-kernel solution will be highly generic, more likely to be relied on by things like Docker, than to replace them.—Zack Brown

2017

14th Annual

High Performance Computing for Wall Street- Cloud, AI and Data Centers

Show and Conference

September 12, 2017 (Tuesday) Roosevelt Hotel, NYC

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

Meetup September 12, Tuesday, for Finance, Trading, Banking, Exchanges, Brokerage, Funds, at one time and one place.

Register Today: HPC, Cloud, AI, Machine Learning, Data Centers, Big Data, Linux, Low Latency, Networks, Cost Savings.

Capital Markets, Systems, Architecture, Cloud, Machine Learning and AI is driving solutions for large data centers and HPC computing.

Go online for the full conference program and save $100. Includes general sessions, drill down sessions, an industry luncheon, $295 in advance. $395 on site.

Qualified end-users are invited to register at no charge. for the full conference.

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

Register online: www.flaggmgmt.com/hpc

HPC sponsors and exhibitors to show and demonstrate all new HPC systems at the Show.

Wall Street IT speakers and Gold Sponsors will lead drill-down sessions in the Grand Ballroom program.

Show Hours: Tues, Sept 12	8:00 - 4:00
Conference Hours:	8:30 - 4:50

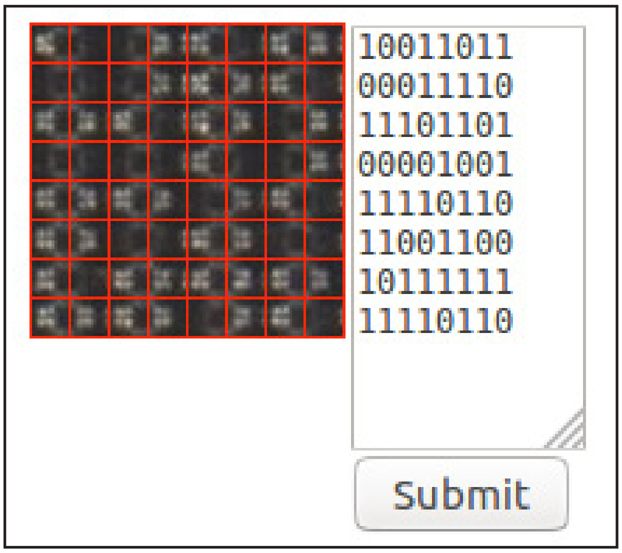
2016 Sponsors

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

Visit: www.flaggmgmt.com/hpc

Non-Linux FOSS: Caps0ff

It's no secret that I love classic video games. Fortunately, thanks to emulation, many of the classic arcade games still can be enjoyed and forever will be available via digital copies of the ROM chips. Sadly, some older systems have protection, making them impossible to dump into ROMs properly. If the chips can't be dumped, how will you ever get a digital copy of the ROM data? Well, the folks over at the http://caps0ff.blogspot.com blog actually are disassembling the original chips and painstakingly transcribing the contents one bit at a time. They're literally looking at the chips and determining the 1s and 0s burned onto them.



```
10011011
00011110
11101101
00001001
11110110
11001100
10111111
11110110
```

Figure 1. This is a sample of the interface for identifying 1s and 0s on the scanned chips.

Yes, there are a lot of chips. Yes, it takes a long time to copy the bits one by one. And yes, you can help. When a chip is stripped down literally to its bits (using various acid baths and so forth), it is scanned at high resolution. Then, pieces of the chips are put into a database over at https://cs.sipr0n.org, and people like you and me can transcribe the photos into 1s and 0s for the project!

Having the underlying code doesn't automatically make your favorite non-dumpable games work in MAME or anything, but it's a crucial and difficult first step. It's also a fairly pricey step, because the chips need to be procured, and the chemicals need to be purchased. If you are passionate enough about preserving the old chips, you might consider donating money to the Caps0ff project as well. There's a great one-time donation site on Indiegogo with an explanation of the project: https://www.indiegogo.com/projects/caps0ff#. And, if you want to support them continually, recurring donations are handled at Patreon: https://www.patreon.com/user?u=4805718.—Shawn Powers

A STRATEGIC CONFERENCE

Developing for the precision agriculture market?

This is your chance to connect with thought leaders driving the future of precision agriculture.

WHY ATTEND?

- **Networking**
  Connect with ag technology influencers across a variety of platforms. Delegates will establish senior-level contacts and foster a dialogue that endures after the conference's end.

- **Conference**
  Take home a deeper understanding of the challenges faced by growers, service providers, and colleagues working in other parts of the nation and world.

- **Get an "In the Field" Perspective with a Pre-Conference Tour**
  Match learnings and contacts gleaned from the conference with a real-world view of Arizona's agricultural technology and the University of Arizona Robotics System.

140+ COMPANIES
35 LEADING SPEAKERS
12 COUNTRIES
185+ ATTENDEES

WITH A FOCUS ON:

Water + Irrigation
Data
Labor
Energy
Water Management
Sensors + Iot
Logistics
Robotics
Labels
Traceability
Food Safety
Sustainability

SESSION TOPICS INCLUDE:

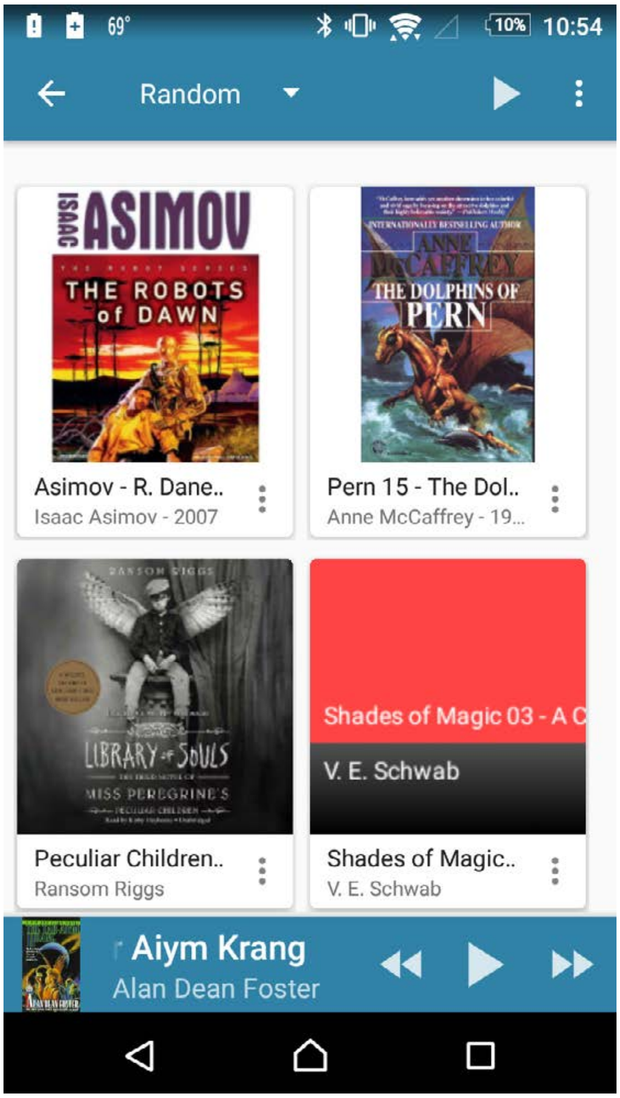
THE STATE OF THE INDUSTRY – PRECISION IN ROW AND SPECIALTY CROP PRODUCTION

EXPLORING THE CONNECTIVE TISSUE BETWEEN EXISTING AND EMERGING TECHNOLOGIES

PRECISIONAg®
VISION Conference
October 10-12, 2017 | Phoenix, AZ

#PRECISIONAGVISION
PRECISIONAGVISION.COM

Android Candy: BookSonic Mobile



This month's Editors' Choice award goes to the cool book streaming system BookSonic by Patrik Johansson. Johansson also created an Android app that connects to the BookSonic server, and it has some extra features the web interface lacks. Namely, it can save your place in a book, and it appears to have multiple speed playback.

Because it's based on the SubSonic system, there already are some great features included, such as caching for offline playback. I personally couldn't get the multi-speed playback to work, but it might be something I configured incorrectly on the server. Nevertheless, to experience BookSonic truly in all its glory, you must download the mobile client. It is a $3 purchase from the Google Play Store, but the source code is available on the author's GitHub page (https://github.com/popeen/Popeens-DSub), if you want to compile it yourself. Check it out!

Note: if you're using the Docker image to run the BookSonic server, be sure to use the following server URL when adding it to the Android app: http://your.docker.server.address:8080/booksonic.

The documentation doesn't mention the need for the trailing /booksonic, but without it, you'll get a connection error.—Shawn Powers

InterDrone

The International Drone Conference and Exposition

Discover the Future – at the World's Largest Commercial Drone Conference & Expo

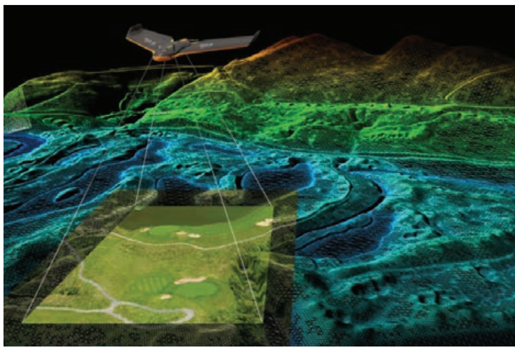


Image credit: Future Aerial Innovations

"If you want to see the state-of-the-art and expand your knowledge about the drone industry, InterDrone is the place to be."

—George Gorrill, Structural Engineer, Thomas Engineering Group

September 6-8, 2017

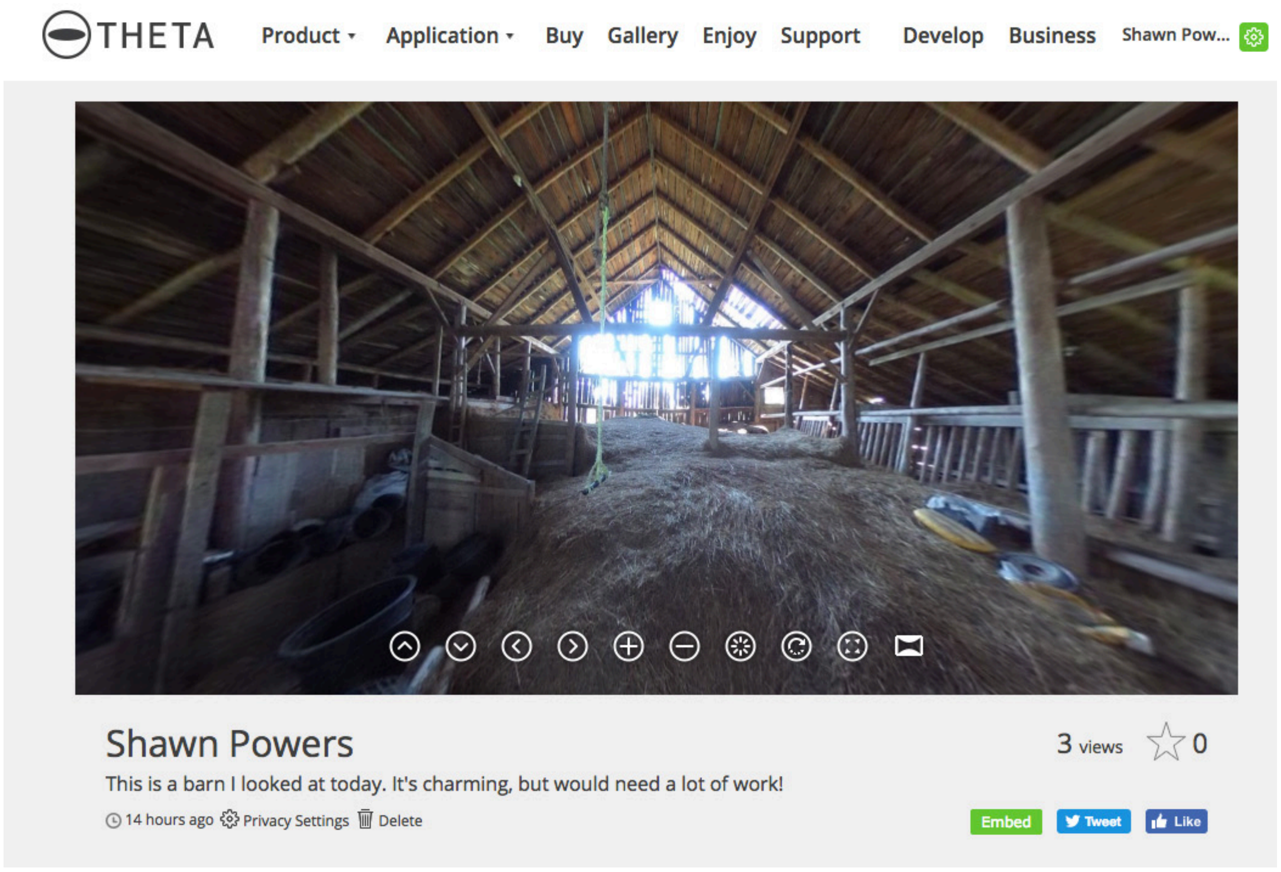
Las Vegas

www.InterDrone.com

Register Early for the Biggest Discount!

BZ Media EMERALD EXPOSITIONS

Produced by BZ Media for Emerald Expositions



Shawn Powers

This is a barn I looked at today. It's charming, but would need a lot of work!

14 hours ago Privacy Settings Delete

3 views 0

Stand Right...Anywhere!

360° photos aren't a new concept, but if you're on Facebook, you'll notice they're more and more common. In fact, Facebook now will convert panoramic photos into a sort of 180° experience where you can drag the photo to see left and right. With a true 360° photo, you literally can spin the photo in a circle to see everywhere. If you're on a mobile device, the experience becomes a sort of VR thing, where if you turn while looking at the phone, it will pan the image for you, as if you're peering through into another world. Honestly, it's pretty cool.

In order to get the really nice 360° photos, you need a camera, or set of cameras, able to accomplish it. The easiest cameras are simple point (well, pointing is sort of moot) and shoot ones. The more complex multi-camera setups require photo stitching after the fact. I like the Ricoh Theta SC camera. It's a happy medium when it comes to quality, and the photos it takes are high enough resolution to appreciate the experience. There is a more expensive Ricoh Theta S, but the photo quality is similar, so unless you want to record 360° video, the SC is perfect.

Admittedly, connecting Android via Wi-Fi to the Theta camera is a pain in the butt, and it rarely works. There are several apps in the Play Store, and it's not clear which app to get. I ultimately skipped using my phone as a viewfinder and just stuck the camera in the air and snapped. Afterward, I downloaded the photos via USB to my computer and uploaded them from there. You can create a free account on http://theta360.com to host your photos, or you can upload them to Twitter, Facebook and so on. If you head to https://theta360.com/s/jCOnQ36A8nmw03lzQKgmQw7km, you can see a barn my wife and I looked at. The 360° experience really makes looking at photos more immersive, and I'm excited to see where the technology goes from here!

—Shawn Powers

Evolving Your Own Life

Much of the software I've covered through the years in this column has been focused on engineering, chemistry or physics. However, there is a growing number of software packages that are being written to apply computational resources to problems in biology. So in this article, I'm looking at one particular package for biology named Biogenesis. Biogenesis provides a platform where you can create entire ecosystems of lifeforms and see how they interact and how the system as a whole evolves over time.

You always can get the latest version from the project's main website (http://biogenesis.sourceforge.net), but it also should be available in the package management systems for most distributions. For Debian-based distributions, install Biogenesis with the following command:

```
sudo apt-get install biogenesis
```

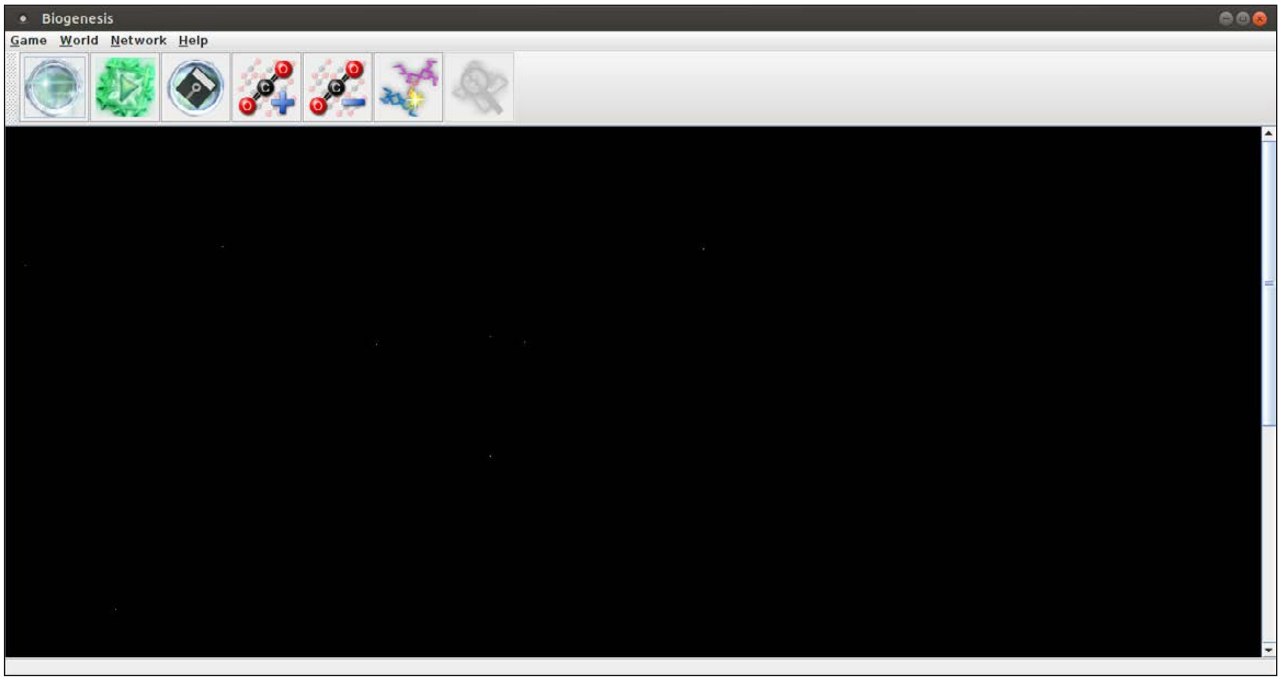


Figure 1. When you first start Biogenesis, you get a blank canvas so you can start creating your world.

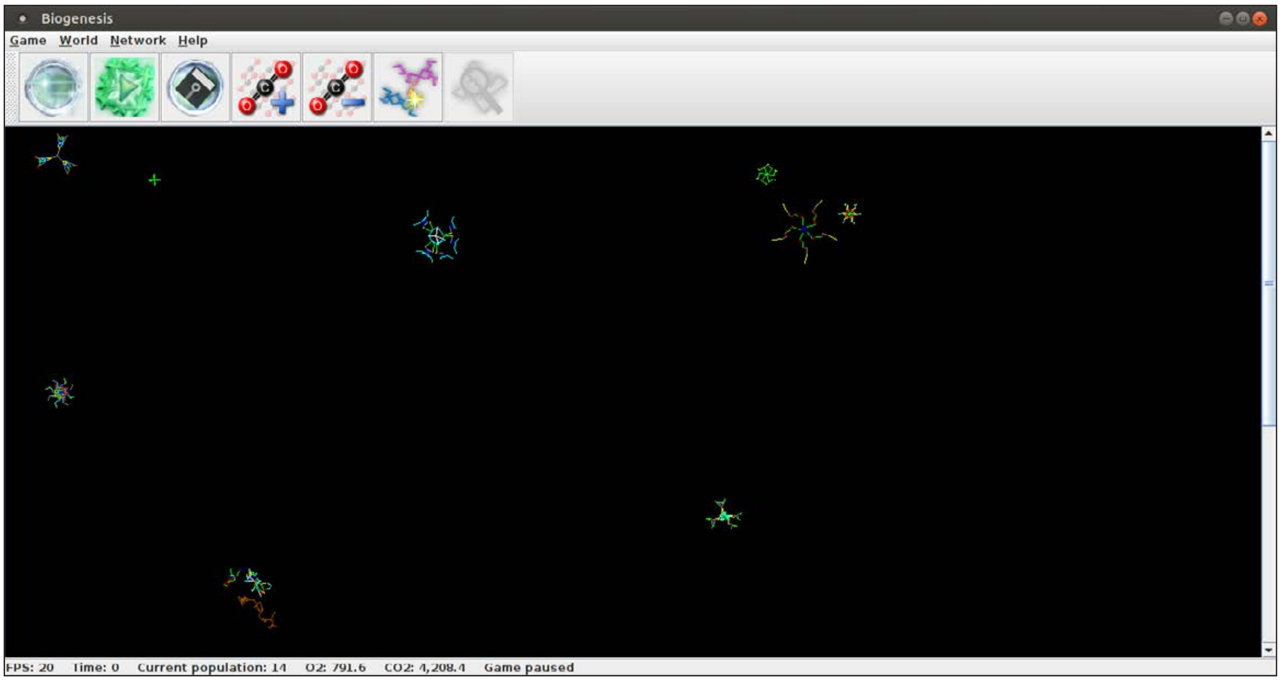
If you do download it directly from the project website, you also will need to have a Java virtual machine installed in order to run it.

To start it, you either can find the appropriate entry in the menu of your desktop environment, or you simply can type `biogenesis` in a terminal window. When it first starts, you will get an empty window within which to create your world.

The first step is to create a world. If you have a previous instance that you want to continue with, click the Game→Open menu item and select the appropriate file. If you want to start fresh, click Game→New to get a new world with a random selection of organisms.

The world starts right away, with organisms moving and potentially interacting immediately. However, you can pause the world by clicking on the icon that is second from the right in the toolbar. Alternatively, you also can just press the p key to pause and resume the evolution of the world.

At the bottom of the window, you'll find details about the world as it currently exists. There is a display of the frames per second, along with the current time within the world. Next, there is a count of the current



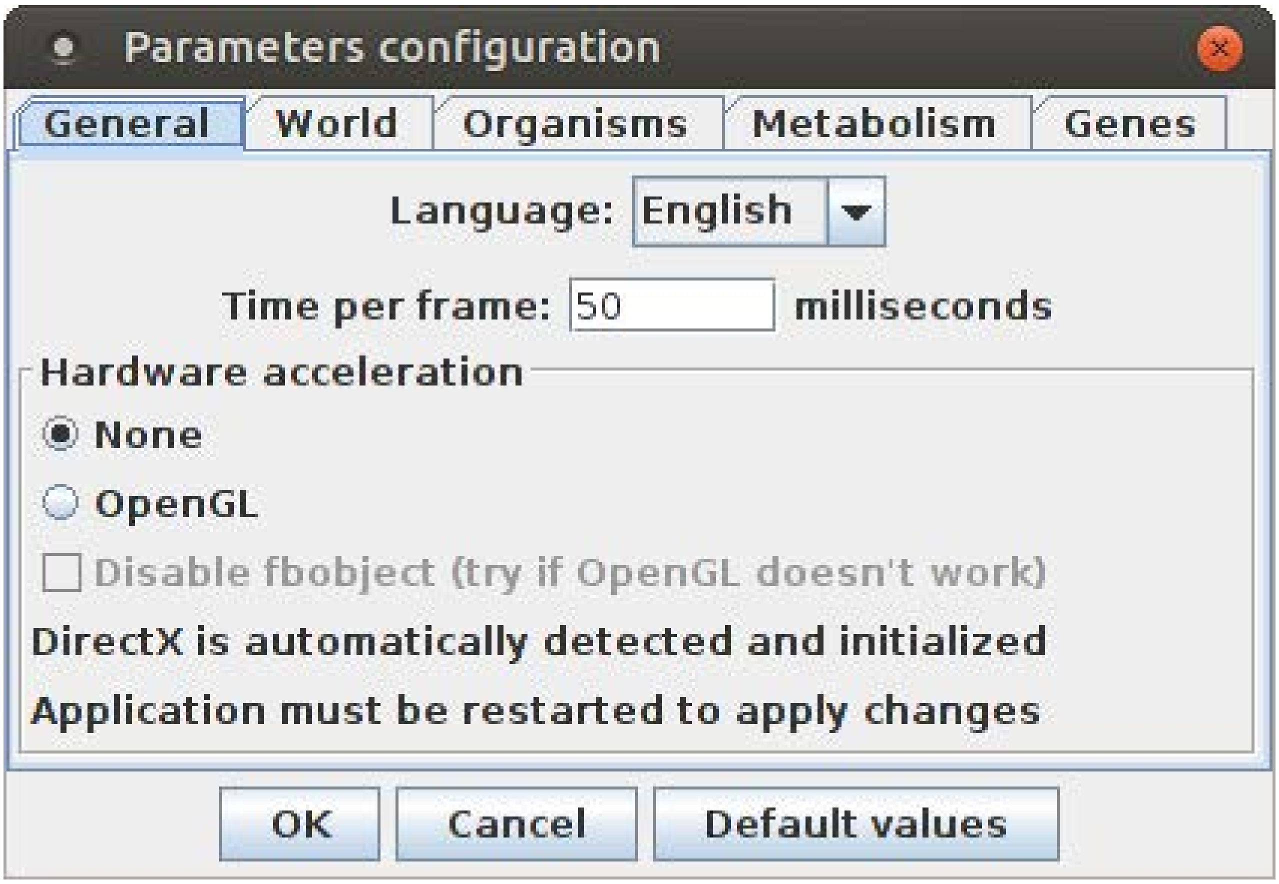
```
FPS: 20    Time: 0    Current population: 14    O2: 791.6    CO2: 4,208.4    Game paused
```

Figure 2. When you launch a new world, you get a random selection of organisms to start your ecosystem.

population of organisms. And finally, there is a display of the current levels of oxygen and carbon dioxide. You can adjust the amount of carbon dioxide within the world either by clicking the relevant icon in the toolbar or selecting the World menu item and then clicking either Increase CO2 or Decrease CO2.

There also are several parameters that govern how the world works and how your organisms will fare. If you select World→Parameters, you'll see a new window where you can play with those values.

The General tab sets the amount of time per frame and whether hardware acceleration is used for display purposes. The World tab lets you set the physical characteristics of the world, such as the size and the initial oxygen and carbon dioxide levels. The Organisms tab allows you to set the initial number of organisms and their initial energy levels.



Language: English

Time per frame: 50 milliseconds

Hardware acceleration

[✓ None]

OpenGL

Disable fbobject (try if OpenGL doesn't work)

DirectX is automatically detected and initialized

Application must be restarted to apply changes

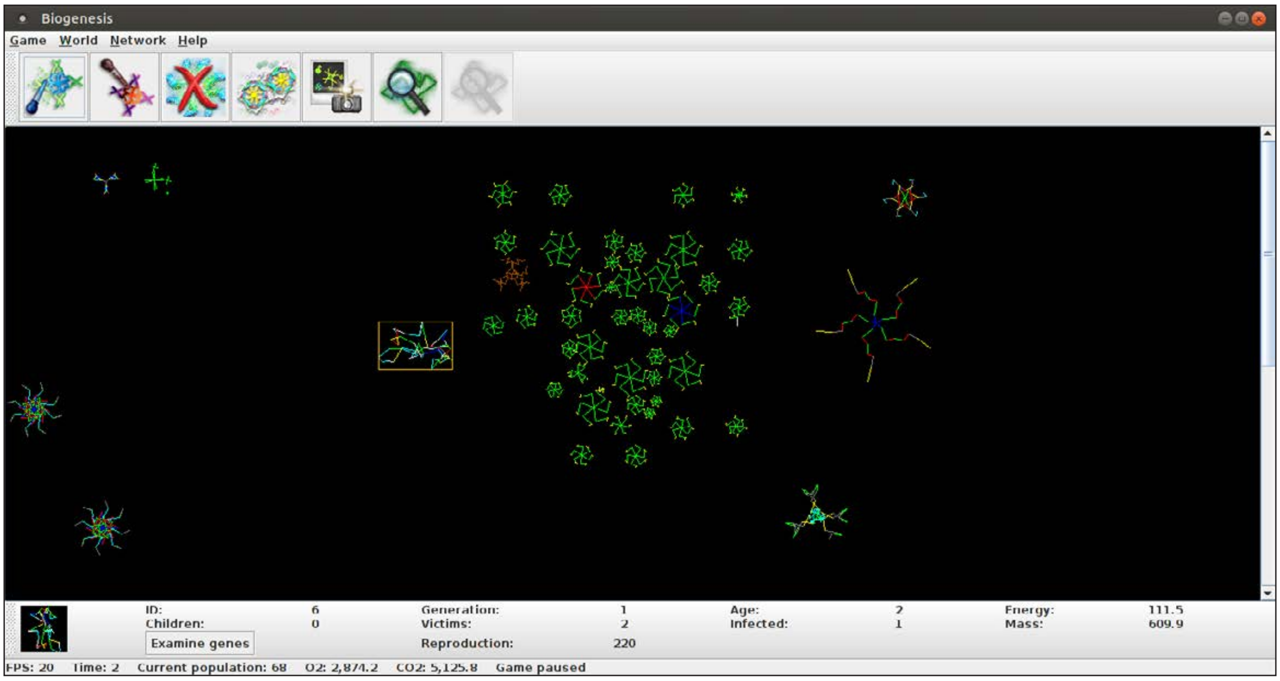
Figure 3. The parameter configuration window allows you to set parameters on the physical characteristics of the world, along with parameters that control the evolution of your organisms.

You also can set their life span and mutation rate, among other items. The Metabolism tab lets you set the parameters around photosynthetic metabolism. And, the Genes tab allows you to set the probabilities and costs for the various genes that can be used to define your organisms.

What about the organisms within your world though? If you click on one of the organisms, it will be highlighted and the display will change.

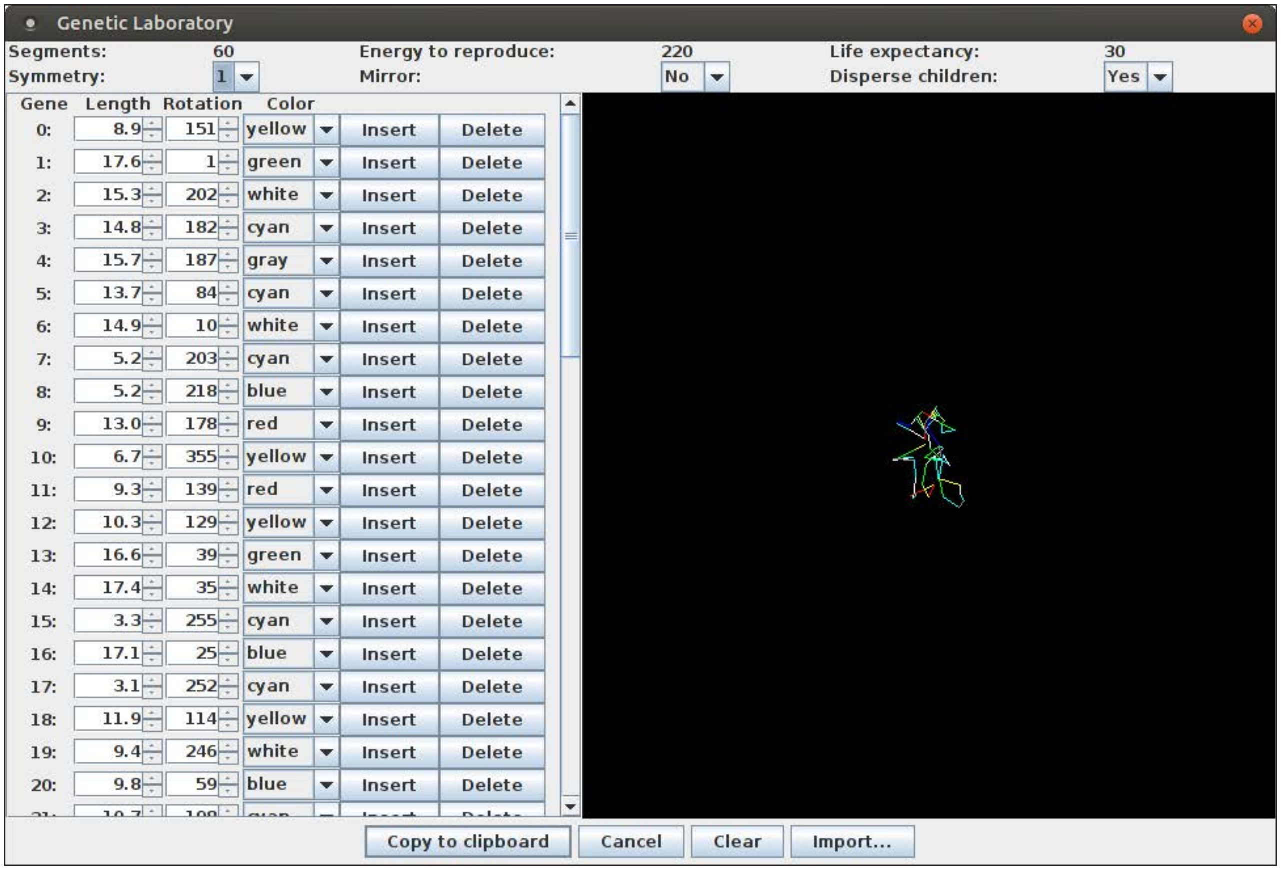
The icon toolbar at the top of the window will change to provide actions that apply to organisms. At the bottom of the window is an information bar describing the selected organism. It shows physical characteristics of the organism, such as age, energy and mass. It also describes its relationships to other organisms. It does this by displaying the number of its children and the number of its victims, as well as which generation it is.

If you want even more detail about an organism, click the Examine genes button in the bottom bar. This pops up a new window called the Genetic Laboratory that allows you to look at and alter the genes making up this organism. You can add or delete genes, as well as change the parameters of existing genes.



```
ID:        6   Generation:    1   Age:       2   Energy:  111.5
Children:  0   Victims:       2   Infected:  1   Mass:    609.9
               Reproduction:  220
FPS: 20   Time: 2   Current population: 68   O2: 2,874.2   CO2: 5,125.8   Game paused
```

Figure 4. You can select individual organisms to find information about them, as well as apply different types of actions.



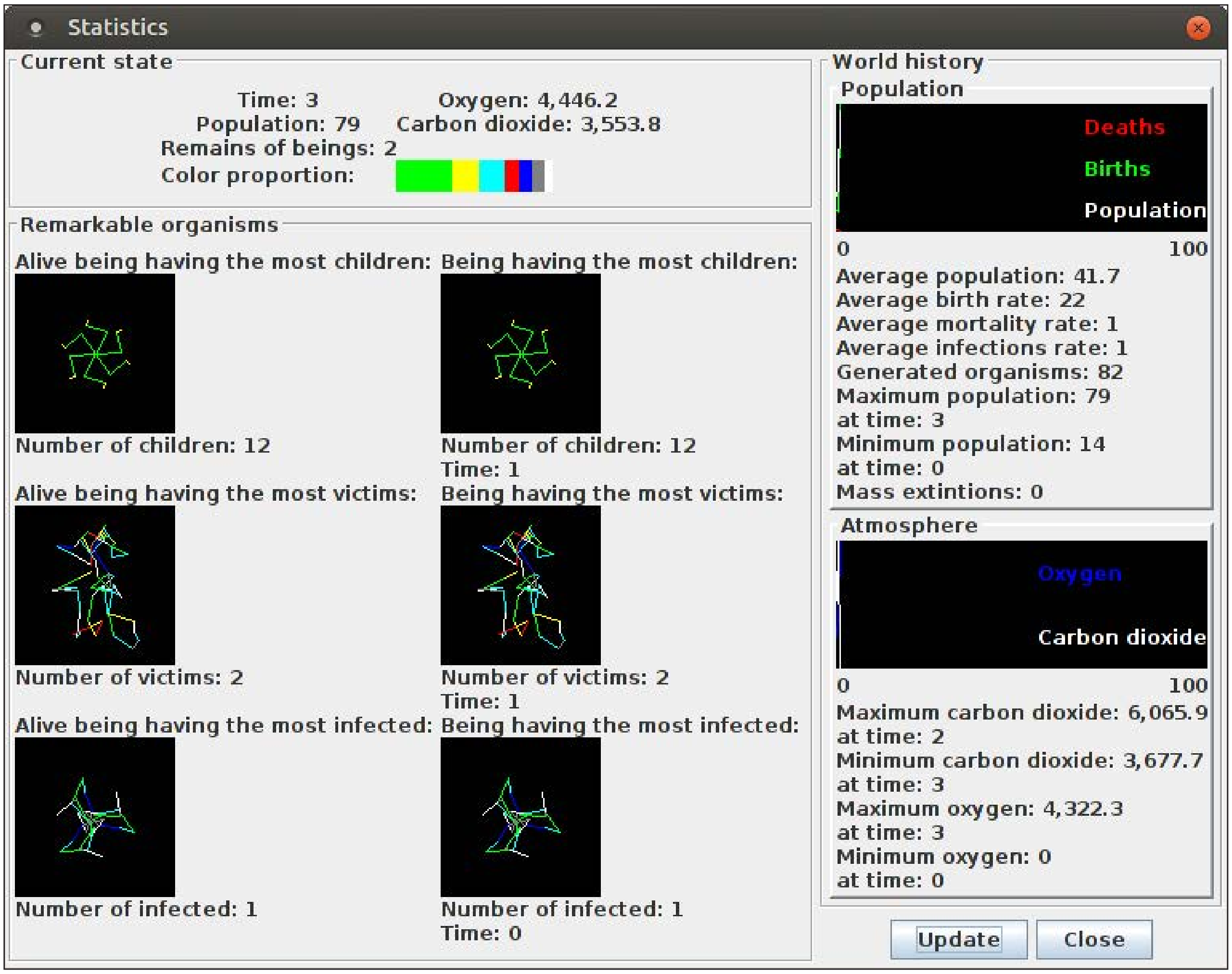
Segments: 60
Energy to reproduce: 220
Life expectancy: 30
Symmetry: 1
Mirror: No
Disperse children: Yes

Gene	Length	Rotation	Color
0:	8.9	151	yellow
1:	17.6	1	green
2:	15.3	202	white
3:	14.8	182	cyan
4:	15.7	187	gray
5:	13.7	84	cyan
6:	14.9	10	white
7:	5.2	203	cyan
8:	5.2	218	blue
9:	13.0	178	red
10:	6.7	355	yellow
11:	9.3	139	red
12:	10.3	129	yellow
13:	16.6	39	green
14:	17.4	35	white
15:	3.3	255	cyan
16:	17.1	25	blue
17:	3.1	252	cyan
18:	11.9	114	yellow
19:	9.4	246	white
20:	9.8	59	blue

Figure 5. The Genetic Laboratory allows you to play with the individual genes that make up an organism.

Right-clicking on a particular organism displays a drop-down menu that provides even more tools to work with. The first one allows you to track the selected organism as the world evolves. The next two entries allow you either to feed your organism extra food or weaken it. Normally, organisms need a certain amount of energy before they can reproduce. Selecting the fourth entry forces the selected organism to reproduce immediately, regardless of the energy level. You also can choose either to rejuvenate or outright kill the selected organism. If you want to increase the population of a particular organism quickly, simply copy and paste a number of a given organism.

Once you have a particularly interesting organism, you likely will want to be able to save it so you can work with it further. When you right-click an organism, one of the options is to export the organism to a file. This pops up a standard save dialog box where you can select the location



Current state

Time: 3 Oxygen: 4,446.2
Population: 79 Carbon dioxide: 3,553.8
Remains of beings: 2
Color proportion:

Remarkable organisms

Alive being having the most children: Number of children: 12

Being having the most children: Number of children: 12, Time: 1

Alive being having the most victims: Number of victims: 2

Being having the most victims: Number of victims: 2, Time: 1

Alive being having the most infected: Number of infected: 1

Being having the most infected: Number of infected: 1, Time: 0

World history

Population

Deaths
Births
Population
0 100

Average population: 41.7
Average birth rate: 22
Average mortality rate: 1
Average infections rate: 1
Generated organisms: 82
Maximum population: 79
at time: 3
Minimum population: 14
at time: 0
Mass extintions: 0

Atmosphere

Oxygen
Carbon dioxide
0 100

Maximum carbon dioxide: 6,065.9
at time: 2
Minimum carbon dioxide: 3,677.7
at time: 3
Maximum oxygen: 4,322.3
at time: 3
Minimum oxygen: 0
at time: 0

Figure 6. The statistics window gives you a breakdown of what's happening within the world you have created.

and filename. The standard file ending for Biogenesis genetic code files is .bgg. Once you start to have a collection of organisms you want to work with, you can use them within a given world by right-clicking a blank location on the canvas and selecting the import option. This allows you to pull those saved organisms back into a world that you are working with.

Once you have allowed your world to evolve for a while, you probably will want to see how things are going. Clicking World→Statistics will pop up a new window where you can see what's happening within your world.

The top of the window gives you the current statistics, including the time, the number of organisms, how many are dead, and the oxygen and carbon dioxide levels. It also provides a bar with the relative proportions of the genes.

Below this pane is a list of some remarkable organisms within your world. These are organisms that have had the most children, the most victims or those that are the most infected. This way, you can focus on organisms that are good at the traits you're interested in.

On the right-hand side of the window is a display of the world history to date. The top portion displays the history of the population, and the bottom portion displays the history of the atmosphere. As your world continues evolving, click the update button to get the latest statistics.

This software package could be a great teaching tool for learning about genetics, the environment and how the two interact. If you find a particularly interesting organism, be sure to share it with the community at the project website. It might be worth a look there for starting organisms too, allowing you to jump-start your explorations.

—Joey Bernard

RETURN TO CONTENTS

THEY SAID IT

Put more trust in nobility of character than in an oath.
—Solon

You make me understand how wonderful it is for little lizards when they find that one special rock that's perfect for sunning themselves on. You make me lizard-happy.
—Randy K. Milholland

It is as hard to see one's self as to look backwards without turning around.
—Henry David Thoreau

I live in the present due to the constraints of the Space-Time Continuum.
—Hank Green

All that really belongs to us is time; even he who has nothing else has that.
—Baltasar Gracian

WISTEM WOMEN IN STEM 2017

THE GLOBAL WOMEN IN STEM CONFERENCE & AWARDS.

100 SPEAKERS - HUNDREDS OF WOMEN IN SCIENCE AND TECHNOLOGY

Women are increasingly becoming the engine driving global economic growth and innovation. Join us as we celebrate the women who are making this possible in spite of all the odds. With over a hundred speakers and hundreds of attendees from all across the world, WISTEM is possibly the biggest Women in STEM conference in the World.

SEPTEMBER 10th-12th

SAN FRANCISCO

EXPERIENCE THE THREE AMAZING DAYS THAT FLY BY, BUT STAY WITH YOU FOREVER.

SESSIONS & TRACKS

START-UP PITCH

AWARDS

WWW.WOMENINSTEMCONFERENCE.COM

An initiative of
MKF, A Public Non Profit
Unlocking Potential. Igniting Ideas. Empowering Actions.

Affiliate Partner Program of

GSMA MOBILE WORLD CONGRESS

ctia Everything Wireless

PREVIOUS
UpFront

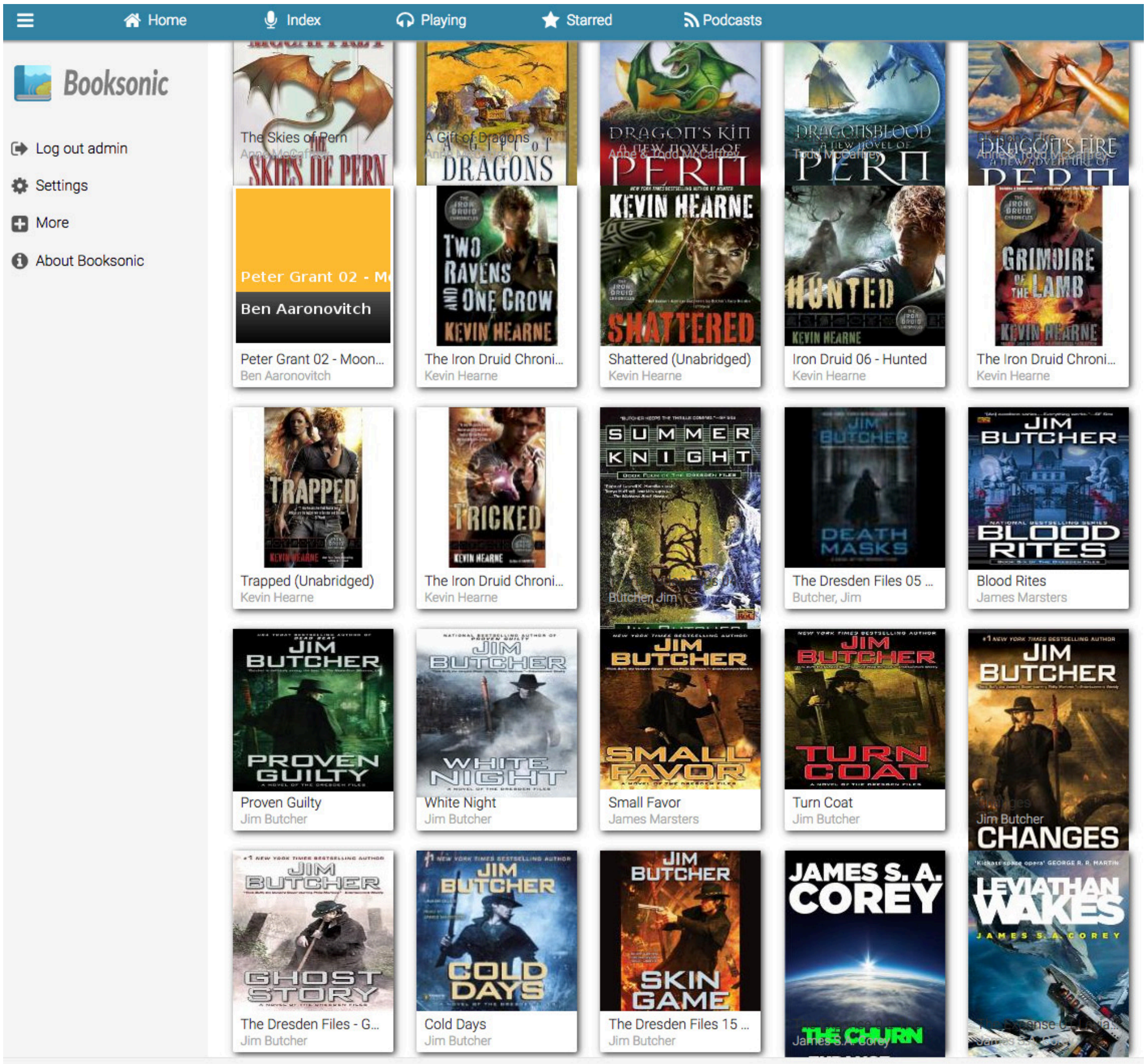
NEXT
Reuven M. Lerner's
At the Forge

Build Your Own Audible

I have audiobooks from a variety of sources, which I've purchased in a variety of ways. I have some graphic audio books in MP3 format, a bunch of Audible books in their DRM'd format and ripped CDs varying from m4b (Apple format for books) to MP3 and even some OGG. That diversity makes choosing a listening platform difficult. In order to meet my idea of perfection, I need:

- A system that plays any audio format.
- A way to play books on multiple platforms, iOS Android and web browsers.
- Current location stored and honored across platforms.
- The ability to play audiobooks at different speeds.
- An easy way to access my entire library remotely.

Several options come close. My favorite Android audiobook app, for instance, is "Listen", available in the Play Store. But, it falls short on the multi-platform front and also on accessing books remotely. Audible itself will do most of what I need, but it doesn't allow importing remote books. And, traditional music players are out.



Honestly, Plex seems like the perfect platform for audiobooks. And although some people do use it, they're just kludging things. Plex doesn't natively support the concepts behind audiobooks, so the process isn't smooth at all. I'm honestly hoping that changes in the future, because it would be a perfect addition to an already amazing system. Thankfully, in the meantime, there's BookSonic.

You've probably heard of SubSonic, which is a music streaming server that allows you to do pretty much what I'm looking for with audiobooks, but it's strictly for music. Patrik Johansson (https://github.com/popeen) has forked SubSonic and created BookSonic, specifically modified to handle audiobooks. It even handles tagging and book art. Currently, the system isn't perfect, but it's closer than any other projects come to book nirvana, and if you use Docker, it's dead simple to get installed. A simple:

```
docker -d create \
  --name booksonic \
  -p 8080:8080 \
  -v <path/to/storage/location/on/host>:/audiobooks \
  -v <path/to/configuration/on/host>:/var/booksonic \
  ironicbadger/booksonic
```

will get BookSonic running on your Docker host. Once it's installed, just head over to http://docker_host:8080 and log in as admin/admin. You can start the book scan, and fairly soon, your books will show up for you to start playing!

Many things about BookSonic do need work (syncing locations to the web client and so on), but it's a great start, and it's a wonderful way to access all your books in one place. Well, as long as you figure out how to strip the DRM from your Audible books anyway! BookSonic is such a great idea, and fills such a gaping hole, that even with its not-feature-complete release, it gets this month's Editors' Choice Award. For more details, head over to http://booksonic.org.

—Shawn Powers

RETURN TO CONTENTS

THE LARGEST OPEN SOURCE CONFERENCE ON THE EAST COAST

ALL THINGS OPEN 2017
IN PARTNERSHIP WITH OPENSOURCE.COM

October 23 & 24 | Raleigh, NC USA

FEATURING THE MOST WELL-KNOWN EXPERTS IN THE WORLD:

Jeff Atwood Stack Overflow	Sara Chipps Jewelbots	Kelsey Hightower Google Cloud	Yehuda Katz Tilde Inc	Angie Jones Twitter

More than 3,000 technologists and decision makers are expected from all over the U.S. and the world

www.AllThingsOpen.org

Thinking Concurrently

How do modern network applications handle multiple connections? Reuven explains the three main paradigms in use today.

PREVIOUS
Editors' Choice

NEXT
Dave Taylor's
Work the Shell

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.

WHEN I FIRST STARTED CONSULTING, and my clients were small organizations just getting started on the web, they inevitably would ask me what kind of high-powered server they would need. My clients were all convinced that they were going to be incredibly popular and important, and that they would have lots of visitors coming to their websites—and it was important that their sites would be able to stand up under this load.

I would remind them that each day has 86,400 seconds. This means that if one new person visits their site each second, the server will need to handle 86,400 requests per day—a trivial number, for most modern computers, especially if you're just serving up static files.

I would then ask, do they really expect to get more than 86,000 visitors per day? The client would almost inevitably answer, somewhat sheepishly, "No, definitely not."

Now, I knew that my clients didn't need to worry about the size or speed of their servers; I really did have their best interests at heart, and I was trying to convince them, in a somewhat dramatic way, that they didn't need to spend money on a new server. But I did take certain liberties with the truth when I presented those numbers—for example:

- There's a difference between 86,400 visitors in one day, spread out evenly across the entire day, and a spike during lunch hour, when many people do their shopping and leisure reading.

- Web pages that contain CSS, JavaScript and images—which is all of them, in the modern era—require more than one HTTP request for each page load. Even if you have 10,000 visitors, you might well have more than 100,000 HTTP requests to your server.

- When a simple web site becomes a web application, you need to start worrying about the speed of back-end databases and third-party services, as well as the time it takes to compute certain things.

So, what do you in such cases? If you can handle precisely one request per second, what happens if more than one person visits your site at the same time? You could make one of them wait until the other is finished and then service the next one, but if you have 10 or 15 simultaneous requests, that tactic eventually will backfire on you.

In most modern systems, the solution has been to take advantage of multiprocessing: have the computer do more than one thing at a time. If a computer can do two things each second, and if your visitors are spread out precisely over the course of a day, then you can handle 172,800 visitors. And if you can do three things at a time, you suddenly can handle 259,200 visitors—and so forth.

How can a computer do more than one thing at a time? With a single CPU, each process gets a "time slice", meaning one fraction of the time that the CPU is working. If you have ten processes, and each gets an equal time slice, then each will run once per second for 0.1 seconds. As you increase the number of processes, the time allocated to each process goes down.

Modern computers come with multiple CPUs (aka "cores"), which means that they actually can do things in parallel, rather than simply give each process a time slice on the system's single processor. In theory, a dual-core system with ten processes will run each process once per second for 0.2 seconds, divided across the processors.

Scaling is never perfectly linear, so you can't really predict things in that way, but it's not a bad way to think about this.

Processes, as described here, are a great way for the computer to do more than one thing at a time. And yet, many applications have other ways of dealing with concurrency. Two of the most popular alternatives to processes are threads and the reactor pattern, especially popular and well known in node.js and the nginx HTTP server.

So in this article, I explore the different types of multiprocessing that exist, looking at the advantages and disadvantages of each one. Even if you're not interested in switching, it's useful to know what is out there.

Processes

The idea behind a process is fairly simple. A running program consists of not only executing code, but also data and some context. Because the code, data and context all exist in memory, the operating system can switch from one process to another very quickly. This combination of code + data + context is known as a "process", and it's the basis for how Linux systems work.

When you start your Linux box, it has a single process. That process then "forks" itself, such that two identical processes are running. The second ("child") process reads new code, data and context ("exec"), and thus starts running a new process. This continues throughout the time that a system is running. When you execute a new program on the command line with & at the end of the line, you're forking the shell process and then exec'ing your desired program in its place.

The Apache httpd server, which is extremely popular and standard on many Linux systems, works by default on a process model. You might think that when a new request comes in, Apache will start up a new process to handle it. But starting it up takes some time, and no one wants to wait for it to happen. The solution is, thus, to "prefork" a bunch of servers. This way, when a new request arrives, Apache can hand off that request to a process. When Apache sees that you're running low on processes, it will add a bunch to the pool, ensuring that there are always enough spare servers. If you reach the limit, things start to cause problems for users,

In many cases, the process model is great. Linux is great at launching processes; it's a fairly low-cost operation, and one that a typical system does hundreds or even thousands of times every hour. Moreover, the kernel developers have learned through the years how to do things intelligently, such that a forked process uses (and writes to) its own memory only when it needs to; until that time, it continues to use memory from its parent process.

Moreover, processes are extremely stable and secure. Memory owned by one process is typically not visible to other processes, let alone writable by other processes. And if a process goes down, it shouldn't take the entire system down with it.

So, what's not to like? Processes are great, no?

Yes, but they also require a fair amount of overhead. If all you're doing is serving up some files, or doing a tiny amount of processing, using a full process for that might seem excessive.

Moreover, if you're doing a number of related tasks that are using the same memory, the fact that every process keeps data separate might make things safe, but also more of a memory hog.

Threads

People coming from a Windows or Java background often scoff at the UNIX tradition of using processes. They say that processes are too heavy for most things, and that you would be better off using threads instead. A thread is similar to a process, except that it exists inside a process.

Just as a computer splits its time across different processes, giving each one a time slice, a process splits its time across different threads, giving each one a time slice.

Because threads exist within an existing process, their startup time is much faster. And because threads share memory with other threads in the process, they consume less memory and are more efficient.

The fact that threads share memory can lead to all sorts of problematic situations. How do you ensure that two different threads aren't modifying the same data at the same time? How do you ensure that your threads execute in the appropriate order—or, how do you make sure that the order isn't actually that important? There are all sorts of issues associated with threading, and people who work with threads know them all too well. The benefits of threads are obvious, but ensuring that they work, and work correctly, can be quite frustrating. Indeed, people who grew up using processes find threads to be fraught with danger and complexity, and they do whatever they can to avoid them.

As a general rule, people with a Microsoft technology background use threads all of the time, starting up a new process only when necessary. By contrast, starting a new process in the Microsoft world takes a long time. People with a UNIX background, meanwhile, think that starting a new process is the easiest and safest thing to do, and they tend to shy away from threads.

Which is the right answer? It all depends. You probably can squeeze more performance out of your computer if you use threads, but at the same time, you can be sure that the code is written in a way that takes advantage of the threads, and it doesn't fall into any of the common traps.

What if you want to use threads, rather than processes, with your Apache server? Years ago, the Apache httpd developers realized that it was foolish for them to push a single model upon their users. For some, and especially for people using Windows, threads were preferable. For many, the traditional processes were preferable. And in some cases, it wasn't obvious which would be better without first doing some benchmarking.

The solution was something known as an MPM (multi-processing module). You can choose the traditional "pre-fork" MPM, typically used on UNIX. You can use the "worker" MPM, which is a combination of processes and threads. If you're on Windows, there's a special "mpm_winnt" MPM, which uses a single process and many threads.

The "worker" MPM is perhaps the most interesting of the bunch, in that it allows you to control the maximum number of processes (with the

`MaxClients` directive), but also a number of threads per process (with the `ThreadsPerChild` directive). You then can experiment with the optimal configuration for your server, deciding which mixture of processes and threads is going to give you the best performance.

An Old-New Idea

During the last few years, a number of network applications have re-discovered a way of writing code that seems counter to all of these ideas. Instead of having multiple processes, or multiple threads, just have a single process, without any threads. That process then can handle all of the incoming network traffic.

At first blush, that sounds a bit crazy. Why keep everything together in a single process?

But, then consider the fact that even on a highly optimized Linux system, there is still some overhead to the "context switch", moving from one process to another. This overhead is repeated at a smaller level within a process, when you switch from thread to thread.

If you handle all of the incoming network requests within a single process, you avoid all of those context switches. You can do that by having an event loop, and then by hanging functions on that event loop. If your event loop contains functions A, B and C, the system gives A a chance to run, then B, then C and then A again. With each opportunity provided by the event loop, you find that A, B and C each progress forward a bit each time.

This actually works quite well, and it has been demonstrated to scale better than processes and threads. What's the problem then?

First of all, the code needs to be written in such a way that it can be divided into functions and put into an event loop. Thinking this way, and writing this style of code, is different from what people typically are used to in the procedural and object-oriented worlds. In many ways, it's like creating a callback function, in that you don't know quite when it'll run.

Among other things, you need to be super careful when working with I/O in this kind of function. That's because disks and networks are extremely slow, and if function B is reading from the disk, then it's waiting idly while neither A nor C has a chance to run. Working with I/O thus requires special handling.

This is part of a bigger issue, namely that modern operating systems use "pre-emptive multitasking", telling each process when its time slice has expired. The reactor pattern uses "cooperative multitasking", in that a rogue function can hog the CPU simply by failing to abide by the rules.

This paradigm is known as the "reactor pattern" and underlies the nginx HTTP server, node.js, Twisted Python and the new asyncio libraries in Python. It has proven itself, but it does require that developers think in new and different ways.

Not to be outdone by nginx, Apache now has an "event" MPM, which handles things using this method. So if you're a fan of Apache and want to try out the reactor pattern without switching to nginx, you can do so. If you're simply using the server and connecting to an external application, rather than writing code that will be embedded within Apache, the MPM will affect performance, but not how you write your application.

Conclusion

So, where does this leave you? First of all, it means you have a number of options. It also means that when you start to worry about performance—and only then—you can start to run experiments to compare the different paradigms and how they work.

But it also means that in today's highly networked world, you might want to consider one or more of these options right away. At the very least, you should be familiar with them and how they work, and the trade-offs associated with them. In particular, while the reactor pattern can be hard to understand, such understanding will make it easier to design architectures that will scale—especially when you truly need it.■

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

RETURN TO CONTENTS

SPTechCon Goes to Washington!

SPTechCon
The SharePoint Technology Conference

November 12-15, 2017

Bethesda North Marriott Hotel & Conference Center

The Best SharePoint and Office 365 Training!

80+ Classes

40+ Expert Speakers and Microsoft MVPs

Get answers to your burning SharePoint questions

Get up to speed with the latest changes from Microsoft

Networking Events, Keynotes, Exhibit Hall and more!

A BZ Media Event

Register Early and Save!

Check out the new classes at www.sptechcon.com

Dungeons, Dragons and Dice

Dungeons, Dragons and Dice—a script that lets you roll those 3d6 and 2d20 that a surprising number of games require.

PREVIOUS
Reuven M. Lerner's
At the Forge

NEXT
Kyle Rankin's
Hack and /

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.

IN MY LAST ARTICLE, I talked about a really simple shell script for a game called Bunco, which is a dice game played in rounds where you roll three dice and compare your values to the round number. Match all three and match the round number, and you just got a bunco for 25 points. Otherwise, any die that match the round are worth one point each. It's simple—a game designed for people who are getting tipsy at the local pub, and it also is easy to program.

The core function in the Bunco program was one that produced a random number between 1–6 to simulate rolling a six-sided die. It looked like this:

```
rolldie()
{
   local result=$1
   rolled=$(( ( $RANDOM % 6 ) + 1 ))
   eval $result=$rolled
}
```

It's invoked with a variable name as the single argument, and it will load a random number between 1–6 into that value—for example:

```
rolldie die1
```

will assign a value 1..6 to $die1. Make sense?

If you can do that, however, what's to stop you from having a second argument that specifies the number of sides of the die you want to "roll" with the function? Something like this:

```
rolldie()
{
   local result=$1 sides=$2
   rolled=$(( ( $RANDOM % $sides ) + 1 ))
   eval $result=$rolled
}
```

To test it, let's just write a tiny wrapper that simply asks for a 20-sided die (d20) result:

```
rolldie die 20
echo resultant roll is $die
```

Easy enough. To make it a bit more useful, let's allow users to specify a sequence of dice rolls, using the standard D&D notation of nDm—that is, *n* *m*-sided dice. Bunco would have been done with 3d6, for example (three six-sided die). Got it?

Since you might well have starting flags too, let's build that into the parsing loop using the ever handy `getopt`:

```
while getopts "h" arg
do
  case "$arg" in
    * ) echo "dnd-dice NdM {NdM}"
        echo "NdM = N M-sided dice"; exit 0 ;;
  esac
done
shift $(( $OPTIND - 1 ))
for request in $* ; do
  echo "Rolling: $request"
done
```

With a well formed notation like 3d6, it's easy to break up the argument into its component parts, like so:

```
dice=$(echo $request | cut -dd -f1)
sides=$(echo $request | cut -dd -f2)
echo "Rolling $dice $sides-sided dice"
```

To test it, let's give it some arguments and see what the program outputs:

```
$ dnd-dice 3d6 1d20 2d100 4d3 d5
Rolling 3 6-sided dice
Rolling 1 20-sided dice
Rolling 2 100-sided dice
Rolling 4 3-sided dice
Rolling  5-sided dice
```

Ah, the last one points out a mistake in the script. If there's no number of dice specified, the default should be 1. You theoretically could default to a six-sided die too, but that's not anywhere near so safe an assumption.

With that, you're close to a functional program because all you need is a loop to process more than one die in a request. It's easily done with a while loop, but let's add some additional smarts to the script:

```
for request in $* ; do
  dice=$(echo $request | cut -dd -f1)
  sides=$(echo $request | cut -dd -f2)
  echo "Rolling $dice $sides-sided dice"
  sum=0  # reset
  while [ ${dice:=1} -gt 0 ] ; do
    rolldie die $sides
    echo "     dice roll = $die"
    sum=$(( $sum + $die ))
    dice=$(( $dice - 1 ))
  done
  echo "  sum total = $sum"
done
```

This is pretty solid actually, and although the output statements need to be cleaned up a bit, the code's basically fully functional:

```
$ dnd-dice 3d6 1d20 2d100 4d3 d5
Rolling 3 6-sided dice
     dice roll = 5
     dice roll = 6
     dice roll = 5
  sum total = 16
Rolling 1 20-sided dice
     dice roll = 16
  sum total = 16
Rolling 2 100-sided dice
     dice roll = 76
     dice roll = 84
  sum total = 160
Rolling 4 3-sided dice
     dice roll = 2
     dice roll = 2
     dice roll = 1
     dice roll = 3
  sum total = 8
Rolling  5-sided dice
     dice roll = 2
  sum total = 2
```

Did you catch that I fixed the case when `$dice` has no value? It's tucked into the reference in the while statement. Instead of referring to it as `$dice`, I'm using the notation `${dice:=1}`, which uses the value specified unless it's null or no value, in which case the value 1 is assigned and used. It's a handy and a perfect fix in this case.

In a game, you generally don't care much about individual die values; you just want to sum everything up and see what the total value is. So if you're rolling 4d20, for example, it's just a single value you calculate and share with the game master or dungeon master.

A bit of output statement cleanup and you can do that:

```
$ dnd-dice.sh 3d6 1d20 2d100 4d3 d5
3d6 = 16
1d20 = 13
2d100 = 74
4d3 = 8
d5 = 2
```

Let's run it a second time just to ensure you're getting different values too:

```
3d6 = 11
1d20 = 10
2d100 = 162
4d3 = 6
d5 = 3
```

There are definitely different values, and it's a pretty useful script, all in all.

You could create a number of variations with this as a basis, including what some gamers enjoy called "exploding dice". The idea is simple: if you roll the best possible value, you get to roll again and add the second value too. Roll a d20 and get a 20? You can roll again, and your result is then 20 + whatever the second value is. Where this gets crazy is that you can do this for multiple cycles, so a d20 could become 30, 40 or even 50.

And, that's it for this article. There isn't much else you can do with dice at this point. In my next article, I'll look at...well, you'll have to wait and see! Don't forget, if there's a topic you'd like me to tackle, please send me a note!■

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

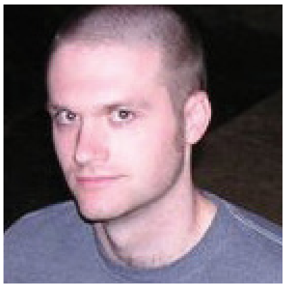
RETURN TO CONTENTS

Banana Backups

Learn how to turn the Banana Pi into a great low-cost, low-power local network backup server.

PREVIOUS
Dave Taylor's
Work the Shell

NEXT
Shawn Powers'
The Open-Source
Classroom



KYLE RANKIN

Kyle Rankin is VP of engineering operations at Final, Inc., the author of many books including *Linux Hardening in Hostile Networks*, *DevOps Troubleshooting* and *The Official Ubuntu Server Book*, and a columnist for *Linux Journal*. Follow him @kylerankin.

IN THE SEPTEMBER 2016 ISSUE, I WROTE AN ARTICLE CALLED "PAPA'S GOT A BRAND NEW NAS" where I described how I replaced my rackmounted gear with a small, low-powered ARM device—the Odroid XU4. Before I settled on that solution, I tried out a few others including a pair of Banana Pi computers—small single-board computers like Raspberry Pis only with gigabit networking and SATA2 controllers on board. In the end, I decided to go with a single higher-powered board and use a USB3 disk enclosure with RAID instead of building a cluster of Banana Pis that each had a single disk attached. Since I had two Banana Pis left over after this experiment, I decided to put them to use, so in this article, I describe how I turned one into a nice little backup server.

The Hardware

Although Raspberry Pis are incredibly popular and useful if you want a small, low-powered, cheap computer, they have their downsides as network backup servers. One of the main downsides is low-performance disk and network speeds. A Raspberry Pi maxes out at 100Mbit on the network and offers only USB2 ports if you want to add a hard drive. Those limitations are what drove me to look for other solutions for my home NAS in the first place, and it's one area where a Banana Pi has an edge. Even though the modern Raspberry Pi 3 has a faster CPU, the old Banana Pi still beats it on network and disk I/O. This makes it pretty ideal as a standalone system for home network backups, depending on your needs.

In my case, I'm not backing up terabytes of media; I just wanted bare-metal backups of my servers and workstations along with backups of important documents. The size of your backups is important, because the Banana Pi is limited to a single SATA2 port, and the board itself can

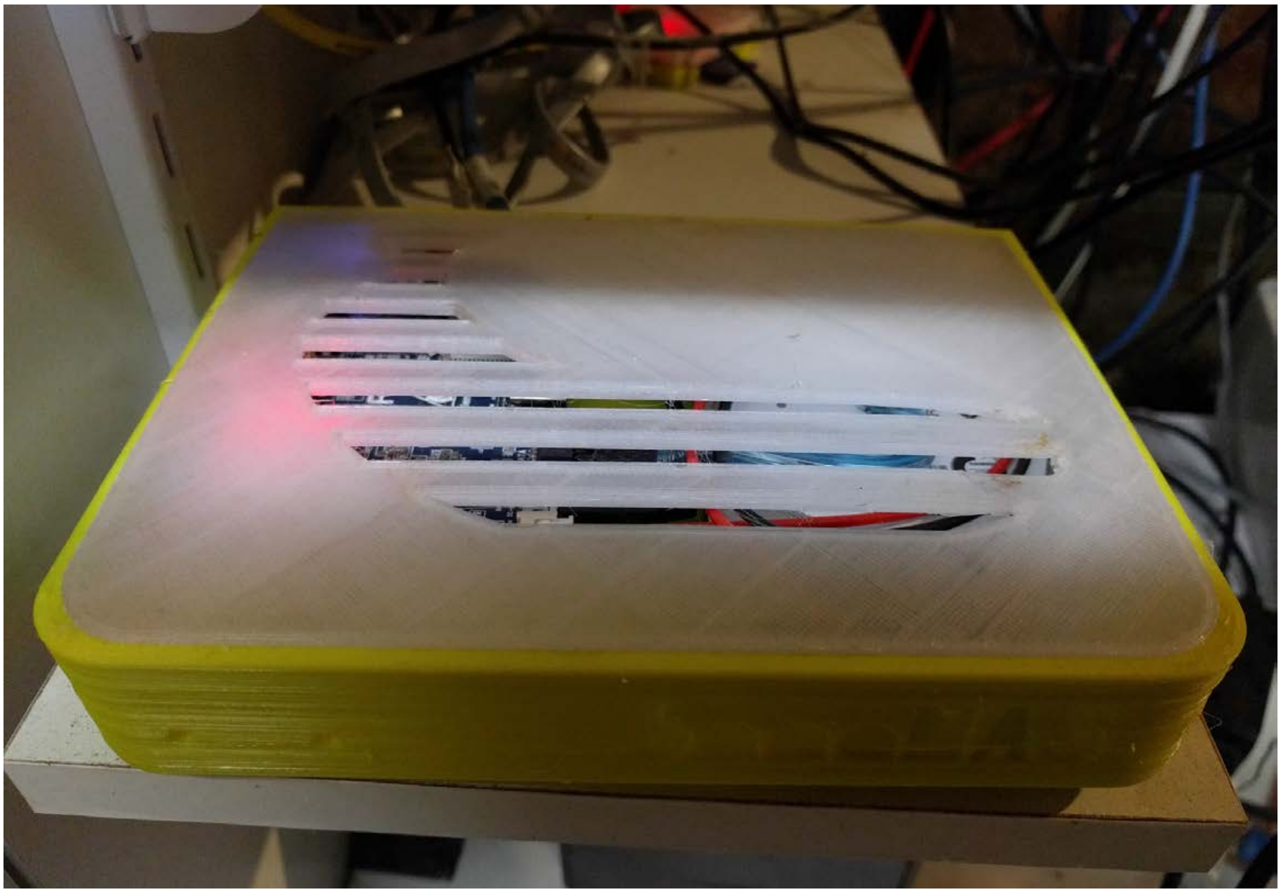


Figure 1. The Banana Pi NAS Case with Lid

power only a 2.5" laptop drive. So if you want to stick with local power, you are limited to 2.5" hard drive sizes. That said, if you were willing to splurge on an externally powered SATA2 enclosure, you could use a 3.5" drive instead. In my case, I happened to have an old 2.5" 500Gb laptop drive lying around that I had since replaced with an SSD. Note that you probably will need to order the appropriate SATA2 cable to connect your hard drive with your Banana Pi—it doesn't typically come with the board.

Although I imagine you could just have the board and a laptop drive sitting on a shelf, I wanted to protect it a bit more than that. Since I have a 3D printer, naturally I went to Thingiverse to see if it had any cases for a Banana Pi. It turns out someone made just the thing I needed—a Banana Pi case that also had mounting points for a 2.5" hard drive. I printed out the case (in yellow, naturally) and was able to mount the board and the laptop drive without any issues.

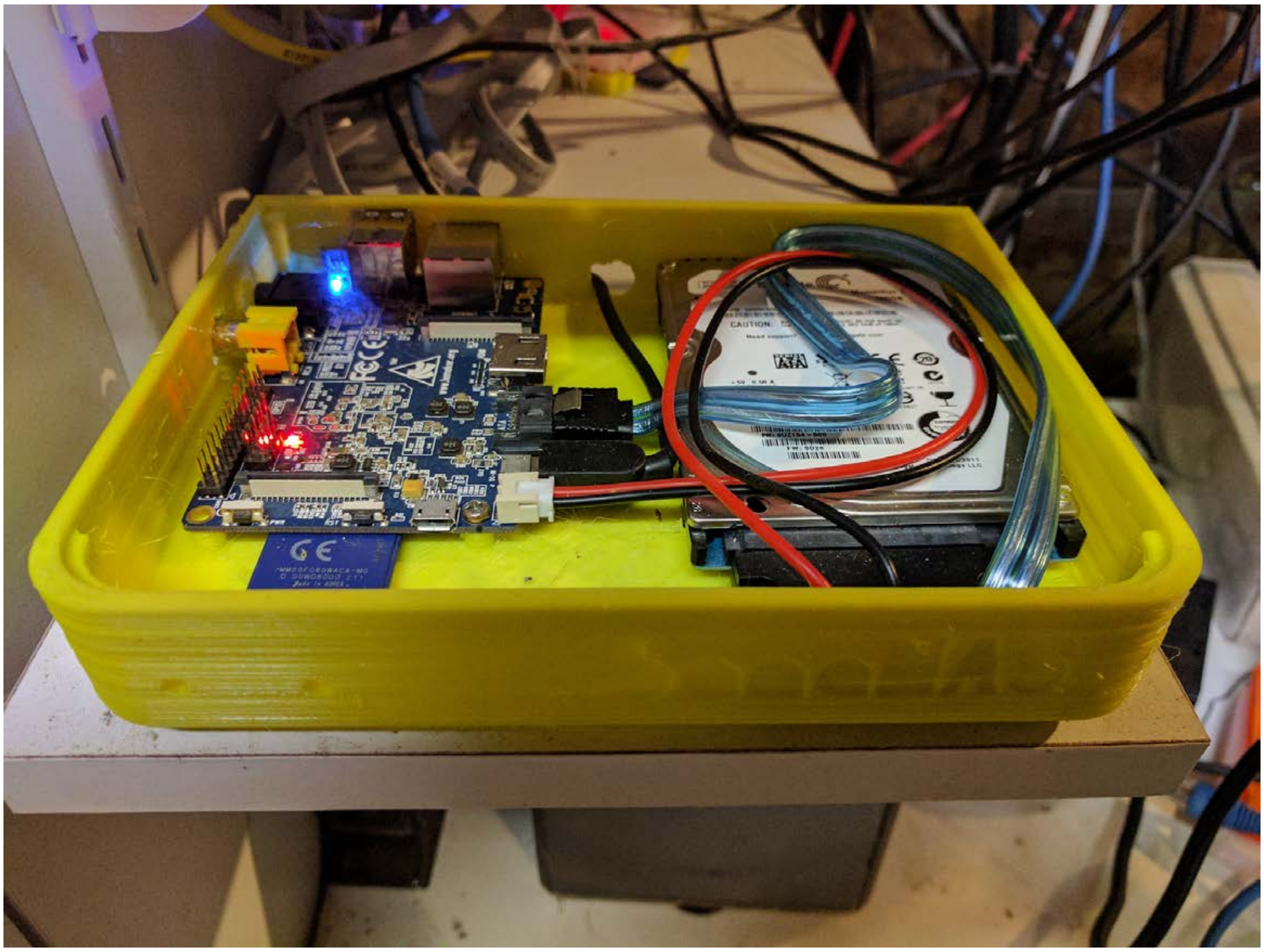


Figure 2. The Banana Pi NAS Case without Lid

I've been a fan of BackupPC as a backup solution for many years. It's available for just about any Linux distribution, supports SMB and rsync-based backups, and it provides a nice web interface to make it easy for users to navigate to their own backups and perform their own restores without having to involve an administrator.

The Software

I've been a fan of BackupPC as a backup solution for many years. It's available for just about any Linux distribution, supports SMB and rsync-based backups, and it provides a nice web interface to make it easy for users to navigate to their own backups and perform their own restores without having to involve an administrator. What's more, it's smart about combining full and incremental backups, and it takes advantage of hard links and compression so that it's very efficient with disk space. You can end up storing weeks of backups without taking much more space than one full backup might consume.

BackupPC already was packaged for the Debian-based distribution made for the Banana Pi, so it was easy to install. In my case, I already had been using BackupPC on my previous server, so I was able to copy over my configuration and backup archive to this new server, and I was ready to go. Obviously, in your case, you will want to follow BackupPC's documentation to add your own user accounts and backup jobs. It's pretty straightforward, and you can configure it both purely on the command line with configuration files or through the web UI.

I did make one tweak to my previous BackupPC configuration file for the Banana Pi. It turns out that the Banana Pi has a green LED on the board that you can control from Linux. The LED is bright enough that I was able to see it through the translucent plastic lid I printed for my case, so I thought it would be handy to turn it on whenever a backup was in progress. I ended up creating two basic scripts to control the LED. First, I created /usr/local/bin/led_on:

```
#!/bin/bash

echo 1 > /sys/class/leds/bananapi\:green\:usr/brightness
```

Then I created a corresponding /usr/local/bin/led_off script:

```
#!/bin/bash

echo 0 > /sys/class/leds/bananapi\:green\:usr/brightness
```

I then used `chmod +x` to add execute permissions to both of these scripts and tested them out with sudo (you need root privileges to write to that device), and sure enough, the LED turned off and on.

To make BackupPC use these scripts, first I had to make sure the local backuppc user could execute them as root, so I added the following line to my /etc/sudoers file:

```
backuppc ALL=(root) NOPASSWD: /usr/local/bin/led_on,
 ➥/usr/local/bin/led_off
```

Finally, I modified my /etc/backuppc/config.pl file to set the led_on script as a command BackupPC would run before it started a backup and led_off to be run after a backup completed. To turn the LED on, I changed the following BackupPC configuration settings:

```
$Conf{DumpPreUserCmd}     = 'sudo /usr/local/bin/led_on';
$Conf{DumpPreShareCmd}    = 'sudo /usr/local/bin/led_on';
$Conf{RestorePreUserCmd}  = 'sudo /usr/local/bin/led_on';
$Conf{ArchivePreUserCmd}  = 'sudo /usr/local/bin/led_on';
```

And to turn the LED back off, I updated the corresponding post-backup settings:

```
$Conf{DumpPostUserCmd}     = 'sudo /usr/local/bin/led_off';
$Conf{DumpPostShareCmd}    = 'sudo /usr/local/bin/led_off';
$Conf{RestorePostUserCmd} = 'sudo /usr/local/bin/led_off';
$Conf{ArchivePostUserCmd} = 'sudo /usr/local/bin/led_off';
```

Once I reloaded the BackupPC service, the LEDs worked great, and I could tell at a glance whether a backup was in progress. The nice thing about this setup for local backups is that it draws very little power and is silent apart from the noise from the hard drive.■

RESOURCES

Banana Pi NAS Case on Thingiverse: https://www.thingiverse.com/thing:1323881

"Papa's Got a Brand New NAS" by Kyle Rankin, September 2016, *LJ*: http://www.linuxjournal.com/content/papas-got-brand-new-nas

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

RETURN TO CONTENTS

Ansible: Making Things Happen

Finally, an automation framework that thinks like a sysadmin. Ansible, you're hired.

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 IRC channel on Freenode.net.

PREVIOUS
Kyle Rankin's Hack and /

NEXT
New Products

IN MY LAST ARTICLE, I DESCRIBED HOW TO CONFIGURE YOUR SERVER AND CLIENTS SO YOU COULD CONNECT TO EACH CLIENT FROM THE SERVER. Ansible is a push-based automation tool, so the connection is initiated from your "server", which is usually just a workstation or a server you `ssh` in to from your workstation. In this article, I explain how modules work and how you can use Ansible in ad-hoc mode from the command line.

Ansible is supposed to make your job easier, so the first thing you need to learn is how to do familiar tasks. For most sysadmins, that means some simple command-line work. Ansible has a few quirks when it comes to command-line utilities, but it's worth learning the nuances, because it makes for a powerful system.

Command Module

This is the safest module to execute remote commands on the client machine. As with most Ansible modules, it requires Python to be installed on the client, but that's it. When Ansible executes commands using the Command Module, it does not process those commands through the user's shell. This means some variables like `$HOME` are not available. It also means stream functions (redirects, pipes) don't work. If you don't need to redirect output or to reference the user's home directory as a shell variable, the Command Module is what you want to use. To invoke the Command Module in ad-hoc mode, do something like this:

```
ansible host_or_groupname -m command -a "whoami"
```

Your output should show `SUCCESS` for each host referenced and then return the user name that the user used to log in. You'll notice that the user is not root, unless that's the user you used to connect to the client computer.

If you want to see the elevated user, you'll add another argument to the `ansible` command. You can add `-b` in order to "become" the elevated user (or the sudo user). So, if you were to run the same command as above with a "-b" flag:

```
ansible host_or_groupname -b -m command -a "whoami"
```

you should see a similar result, but the `whoami` results should say `root` instead of the user you used to connect. That flag is important to use, especially if you try to run remote commands that require root access!

Shell Module

There's nothing wrong with using the Shell Module to execute remote commands. It's just important to know that since it uses the remote user's environment, if there's something goofy with the user's account, it might cause problems that the Command Module avoids. If you use the Shell Module, however, you're able to use redirects and pipes. You can use the whoami example to see the difference. This command:

```
ansible host_or_groupname -m command -a "whoami > myname.txt"
```

should result in an error about > not being a valid argument. Since the Command Module doesn't run inside any shell, it interprets the greater-than character as something you're trying to pass to the whoami command. If you use the Shell Module, however, you have no problems:

```
ansible host_or_groupname -m shell -a "whom > myname.txt"
```

This should execute and give you a SUCCESS message for each host, but there should be nothing returned as output. On the remote machine, however, there should be a file called myname.txt in the user's home directory that contains the name of the user. My personal policy is to use the Command Module whenever possible and to use the Shell Module if needed.

The Raw Module

Functionally, the Raw Module works like the Shell Module. The key difference is that Ansible doesn't do any error checking, and STDERR, STDOUT and Return Code is returned. Other than that, Ansible has no idea what happens, because it just executes the command over SSH directly. So while the Shell Module will use /bin/sh by default, the Raw Module just uses whatever the user's personal default shell might be.

Why would a person decide to use the Raw Module? It doesn't require Python on the remote computer—at all. Although it's true that most servers have Python installed by default, or easily could have it installed, many embedded devices don't and can't have Python installed. For most configuration management tools, not having an agent program installed means the remote device can't be managed. With Ansible, if all you have is SSH, you still can execute remote commands using the Raw Module. I've used the Raw Module to manage Bitcoin miners that have a very minimal embedded environment. It's a powerful tool, and when you need it, it's invaluable!

Copy Module

Although it's certainly possible to do file and folder manipulation with the Command and Shell Modules, Ansible includes a module specifically for copying files to the server. Even though it requires learning a new syntax for copying files, I like to use it because Ansible will check to see whether a file exists, and whether it's the same file. That means it copies the file only if it needs to, saving time and bandwidth. It even will make backups of existing files! I can't tell you how many times I've used `scp` and `sshpass` in a Bash `FOR` loop and dumped files on servers, even if they didn't need them. Ansible makes it easy and doesn't require `FOR` loops and IP iterations.

The syntax is a little more complicated than with Command, Shell or Raw. Thankfully, as with most things in the Ansible world, it's easy to understand—for example:

```
ansible host_or_groupname -b -m copy \
    -a "src=./updated.conf dest=/etc/ntp.conf \
        owner=root group=root mode=0644 backup=yes"
```

This will look in the current directory (on the Ansible server/workstation) for a file called updated.conf and then copy it to each host. On the remote system, the file will be put in /etc/ntp.conf, and if a file already exists, and it's different, the original will be backed up with a date extension. If the files are the same, Ansible won't make any changes.

I tend to use the Copy Module when updating configuration files. It would be perfect for updating configuration files on Bitcoin miners, but unfortunately, the Copy Module does require that the remote machine has Python installed. Nevertheless, it's a great way to update common files on many remote machines with one simple command. It's also important to note that the Copy Module supports copying remote files to other locations on the remote filesystem using the `remote_src=true` directive.

File Module

The File Module has a lot in common with the Copy Module, but if you try to use the File Module to copy a file, it doesn't work as expected. The File Module does all its actions on the remote machine, so `src` and `dest` are all references to the remote filesystem. The File Module often is used for creating directories, creating links or deleting remote files and folders. The following will simply create a folder named /etc/newfolder on the remote servers and set the mode:

```
ansible host_or_groupname -b -m file \
       -a "path=/etc/newfolder state=directory mode=0755"
```

You can, of course, set the owner and group, along with a bunch of other options, which you can learn about on the Ansible doc site. I find I most often will either create a folder or symbolically link a file using the File Module. To create a symlink:

```
sensible host_or_groupname -b -m file \
         -a "src=/etc/ntp.conf dest=/home/user/ntp.conf \
             owner=user group=user state=link"
```

Notice that the `state` directive is how you inform Ansible what you actually want to do. There are several state options:

- `link` — create symlink.
- `directory` — create directory.
- `hard` — create hardlink.
- `touch` — create empty file.
- `absent` — delete file or directory recursively.

This might seem a bit complicated, especially when you easily could do the same with a Command or Shell Module command, but the clarity of using the appropriate module makes it more difficult to make mistakes. Plus, learning these commands in ad-hoc mode will make playbooks, which consist of many commands, easier to understand (I plan to cover this in my next article).

File Management

Anyone who manages multiple distributions knows it can be tricky to handle the various package managers. Ansible handles this in a couple ways. There are specific modules for apt and yum, but there's also a generic module called "package" that will install on the remote computer regardless of whether it's Red Hat- or Debian/Ubuntu-based.

Unfortunately, while Ansible usually can detect the type of package manager it needs to use, it doesn't have a way to fix packages with different names. One prime example is Apache. On Red Hat-based systems, the package is "httpd", but on Debian/Ubuntu systems, it's "apache2". That means some more complex things need to happen in order to install the correct package automatically. The individual modules, however, are very easy to use. I find myself just using apt or

Anyone who manages multiple distributions knows it can be tricky to handle the various package managers. Ansible handles this in a couple ways.

yum as appropriate, just like when I manually manage servers. Here's an apt example:

```
ansible host_or_groupname -b -m apt \
          -a "update_cache=yes name=apache2 state=latest"
```

With this one simple line, all the host machines will run `apt-get update` (that's the `update_cache` directive at work), then install apache2's latest version including any dependencies required. Much like the File Module, the `state` directive has a few options:

- `latest` — get the latest version, upgrading existing if needed.

- `absent` — remove package if installed.
- `present` — make sure package is installed, but don't upgrade existing.

The Yum Module works similarly to the Apt Module, but I generally don't bother with the `update_cache` directive, because yum updates automatically. Although very similar, installing Apache on a Red Hat-based system looks like this:

```
ansible host_or_groupname -b -m yum \
      -a "name=httpd state=present"
```

The difference with this example is that if Apache is already installed, it won't update, even if an update is available. Sometimes updating to the latest version isn't want you want, so this stops that from accidentally happening.

Just the Facts, Ma'am

One frustrating thing about using Ansible in ad-hoc mode is that you don't have access to the "facts" about the remote systems. In my next article, where I plan to explore creating playbooks full of various tasks, you'll see how you can reference the facts Ansible learns about the systems. It makes Ansible far more powerful, but again, it can be utilized only in playbook mode. Nevertheless, it's possible to use ad-hoc mode to peek at the sorts information Ansible gathers. If you run the `setup` module, it will show you all the details from a remote system:

```
ansible host_or_groupname -b -m setup
```

That command will spew a ton of variables on your screen. You can scroll through them all to see the vast amount of information Ansible pulls from the host machines. In fact, it shows so much information, it can be overwhelming. You can filter the results:

```
ansible host_or_groupname -b -m setup -a "filter=*family*"
```

That should just return a single variable, `ansible_os_family`, which likely will be Debian or Red Hat. When you start building more complex Ansible setups with playbooks, it's possible to insert some logic and conditionals in order to use yum where appropriate and apt where the system is Debian-based. Really, the facts variables are incredibly useful and make building playbooks that much more exciting.

But, that's for another article, because you've come to the end of the second installment. Your assignment for now is to get comfortable using Ansible in ad-hoc mode, doing one thing at a time. Most people think ad-hoc mode is just a stepping stone to more complex Ansible setups, but I disagree. The ability to configure hundreds of servers consistently and reliably with a single command is nothing to scoff at. I love making elaborate playbooks, but just as often, I'll use an ad-hoc command in a situation that used to require me to `ssh` in to a bunch of servers to do simple tasks. Have fun with Ansible; it just gets more interesting from here!■

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

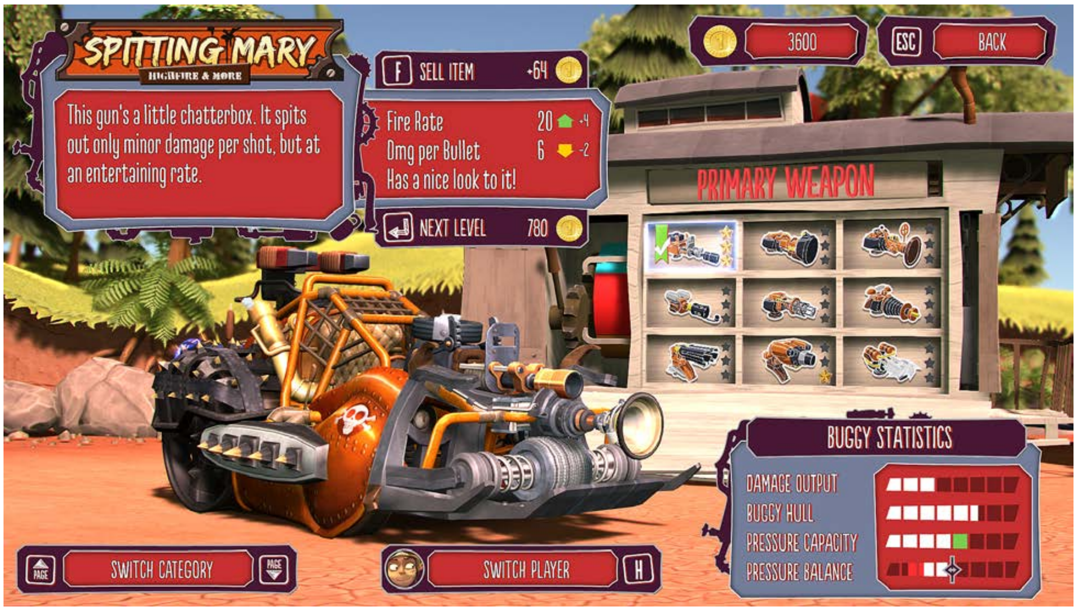
RETURN TO CONTENTS

PREVIOUS
Shawn Powers'
The Open-Source
Classroom

NEXT
Feature: The
Full Stack Project

SUSE Linux Enterprise Server for SAP Applications

Saving customers time, effort and budget as they implement SAP landscapes, including on-premises and now on-demand, are the core selling points for SUSE Linux Enterprise Server for SAP Applications. The latest release of the SAP-focused SUSE Linux server is also now available as the operating system for SAP solutions on Google Cloud Platform (GCP). The first supported Linux for SAP HANA on GCP, SUSE Linux Enterprise Server for SAP Applications bolsters enterprise agility and reduces operating costs as customers pay only for what they use. With the addition of Google Cloud Platform, SUSE Linux Enterprise Server for SAP Applications now is available on three major public cloud providers, including Amazon Web Services and Microsoft Azure.
http:/suse.com/google



Chasing Carrots' *Pressure Overdrive*

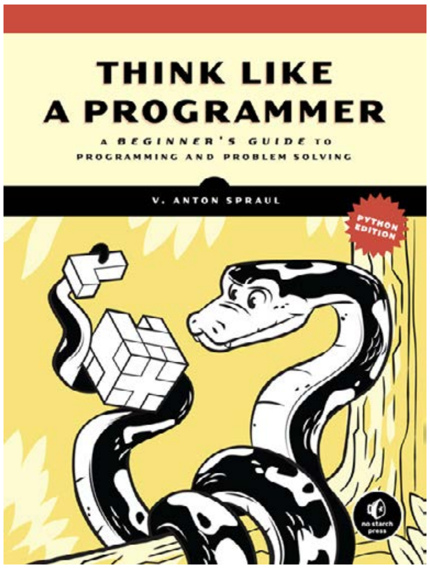
A "funky, four-wheeled shoot 'em up" is how independent game-developer Chasing Carrots describes its newest game release *Pressure Overdrive* for Linux, Mac OS, Windows and Xbox. Shifting to a higher gear from its high-octane 2013 predecessor, *Pressure*, *Pressure Overdrive* seeks a wider audience with a fast-paced mix of humor and action, classic couch co-op fun and constant car crashes in colorful environments. *Pressure Overdrive* takes players on a wild ride as they fight against the evil Count Soap, who has stolen the water out of nearby surrounding rivers to fill his gigantic "Uber-Spa". As pilots of steam-powered and heavily armed buggies, players must ram and blast their way through hordes of wacko enemies, including a string of powerful bosses. Between missions, players bling up their ride with a variety of upgrades and customization options. Other novelties in *Pressure Overdrive* include a new twin stick control scheme, additional weapons, new upgrade paths, an endless mode and couch co-op for playing with friends. Rounding out the new feature set is a new tutorial level that helps players ease themselves into the game.
http://chasing-carrots.com

Caldwell Partners' Cyber Advisory Board Service

For many enterprises, cyber risk is the top business risk. Meanwhile, there is simply not a sufficiently large talent pool of cyber-risk professionals to satisfy the ever-growing demand. To assuage this business challenge, executive search firm Caldwell Partners announced the launch of its Cyber Advisory Board service, a means for companies to obtain the level of expertise they require to implement industry best practices quickly and effectively. Caldwell Partners' new service allows enterprises to eschew failed searches for CISOs, which frequently result in over-priced and under-qualified candidates. Companies obtain access to a Cyber Advisory Board of recognized cyber industry leaders, tailored to fit needs the needs of a particular company and its industry. This solution is especially helpful for clients who opt for step-up candidates or deploy trusted executives open to a career change into cyber security. The advisory board provides a range of services—from mentoring a company's CISO to helping develop cyber strategy to educating the company's C-level executives and boards, all the way to giving guidance on best practices with regard to cyber breach responses. A Cyber Advisory Board provides deep expertise and consistency and satisfies increased governance on public company boards faced with regulators who exert ever greater pressure on companies to demonstrate a commitment to best practices.
http://caldwellpartners.com/cyber

Paragon Software Group's Paragon ExtFS for Mac

Ever more Mac aficionados are discovering the virtues of Linux, especially when their older hardware can experience a renaissance. One annoying barrier to dual-boot nirvana is filesystem incompatibility, whereby the Linux side can access the Mac side, but Apple's macOS doesn't support Linux drives at all—not even in read-only mode. A handy solution is Paragon Software Group's Paragon ExtFS for Mac, a low-level macOS filesystem driver designed to eliminate filesystem incompatibility between Linux and Mac operating systems. The solution grants transparent read/write access to ExtFS Linux partitions on macOS, allowing Mac users to access files fully that are stored on Linux volumes hassle-free, streamline data sharing and transfer up to 4GB+ files at a high rate. The new 11th edition of ExtFS for Mac features a completely new UI and advanced features, including extended mounting options and a menu bar app allowing users one-click access to all ExtFS drives and instant execution of the most common volume operations—for example, volume mount, unmount and verify. Large volume support enables mounting volumes more than 2TB in size. ExtFS for Windows also is available from the company.
http://paragon-drivers.com/extfs-mac

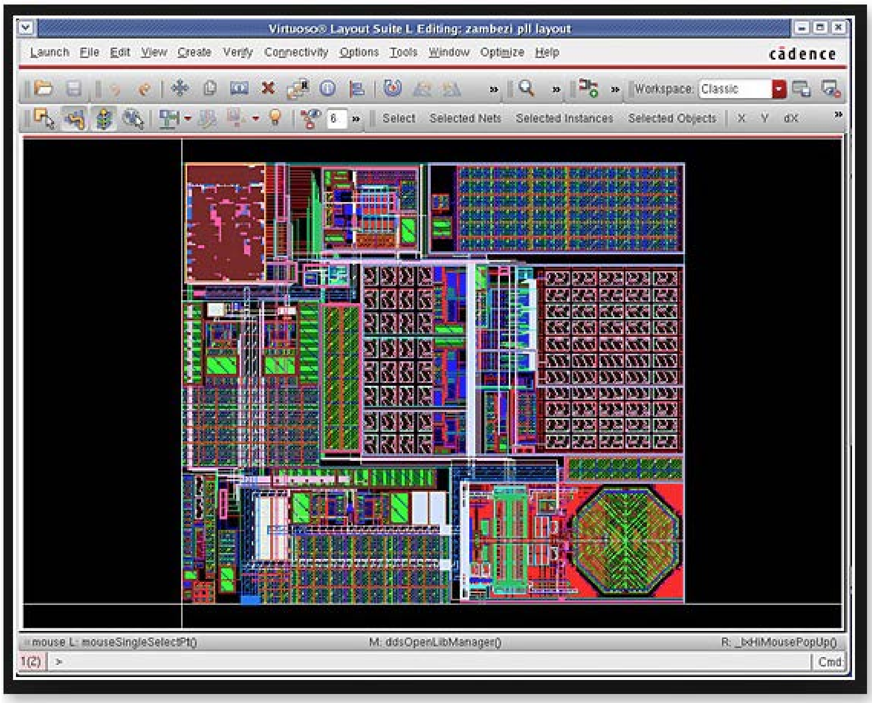


V. Anton Spraul's *Think Like a Programmer*, Python Edition

What is programming? Sure, it consists of syntax and the assembly of code, but it is essentially a means to solve problems. To study programming, then, is to study the art of problem solving, and a new book from V. Anton Spraul, *Think Like a Programmer*, Python Edition, is a guide to sharpening skills in both spheres. Subtitled *A Beginner's Guide to Programming and Problem Solving*, Spraul's book helps transition programmers in training from reading programs to writing them in Python. No prior programming experience required! Rather than simply point out solutions to problems, Spraul gets readers thinking by illustrating techniques that instruct how to self-solve programming problems. Each chapter covers a single programming concept, such as data types, control flow, code reuse, recursion and classes, topped off by a series of Python-based exercises that put readers' skills to the test. In *Think Like a Programmer, Python Edition*, readers break big problems down into simple, manageable steps to build into solutions, write custom functions to solve new problems, use a debugger to examine each line of a running program in order to understand fully how it works and tackle problems strategically by turning each new concept into a problem-solving tool. Additional chapters are included on early programming topics, such as variables, decisions and looping.
http://nostarch.com/thinkpython

eCosCentric Limited's eCosPro

eCos—which means the "Embedded Configurable Operating System"—is an open-source RTOS for deeply embedded applications. Deployed in a diversity of markets and devices, eCos' popularity is a result of a variety of commercial and technical advantages over competing RTOS offerings. The developer of eCos, eCosCentric Limited, recently announced the latest 4.1 release of eCosPro, the stable, fully tested and supported version of the operating system and RedBoot bootstrap firmware. The new 4.1 release of the eCosPro Developer's Kit includes the latest Eclipse Neon IDE, provides improvements to the eCosPro Eclipse plugin and development tools and integrates a variety of runtime enhancements. Complementary to the major upgrade to Eclipse Neon.3, enhancements were added to the eCosPro Eclipse plugin that improve the eCos developer experience, such as multiple eCos Application and Configuration projects per workspace and a new hardware debug launcher. Meanwhile, the eCosPro 4.1 eCos Configuration tool boasts further refinements that aid developer productivity, such as locally installed documentation that is fully searchable. Runtime enhancements include support for Cortex-A53 and Cortex-A7 SMP, Raspberry Pi target platform and ST's low-power STM32L series and its enhanced on-chip peripheral feature set, among other upgrades.
http://ecoscentric.com


Virtuoso® Layout Suite L Editing: zambezi pll layout
Launch File Edit View Create Verify Connectivity Options Tools Window Optimize Help
cādence
Workspace: Classic
Select Selected Nets Selected Instances Selected Objects X Y dX
mouse L: mouseSingleSelectPt()
M: ddsOpenLibManager()
R: _lxHiMousePopUp()
1(2) >
Cmd

StarNet Communications Corp's FastX

WebAssembly browser technology is important for making the browser go beyond what JavaScript can do. StarNet Communications Corp says it is the first to plant a WebAssembly flag in the EDA space by integrating WebAssembly technology into its FastX remote Linux display solution. The addition is part of StarNet's new FastX 2.4 release, which the company promises will provide EDA engineers with a significant browser client performance upgrade, particularly with video and graphics applications, such as Computer Aided Engineering (CAE) tools used in semiconductor design. The integration of WebAssembly improves in-browser client-side scripting by executing instructions natively rather than through an interpreter, such that Linux applications through a browser will run at near native speed. Browsers with support for WebAssembly include Firefox, Chrome, Safari and Edge. Besides the performance upgrade, users also will enjoy an updated UI, optimizations for operating in cloud environments and additions to the admin toolset, namely several new RDP Protocol extensions to help reduce bandwidth consumption, simplified upgrades and installation and an Advanced Windows Management system.
http://starnet.com

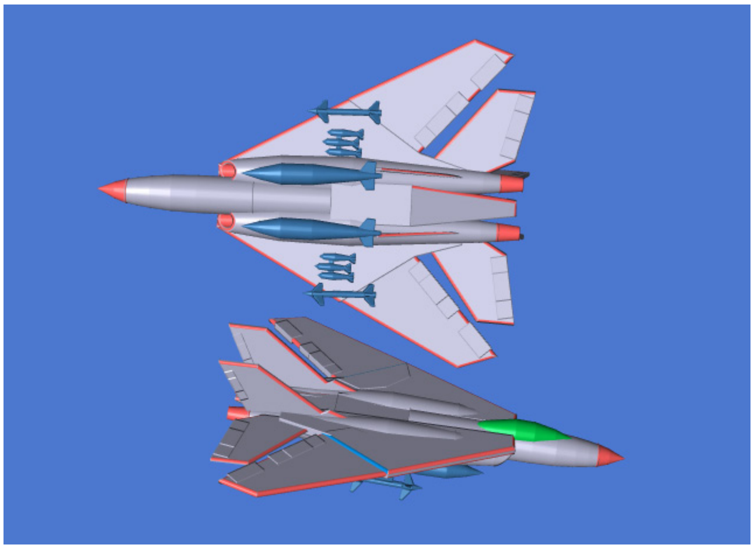
Whether you're trying to grow your company or looking for a change in your career, when you use Drupal Jobs, you don't just help yourself -- you help the community thrive.

Proceeds from every job listing on Drupal Jobs go towards funding improvements to Drupal.org, the Drupal community's online home.

Get a job. Give a job. And invest in the future of Drupal when you do it.

Drupal™ Jobs

Drupal™ ASSOCIATION



VariCAD s.r.o.'s VariCAD

"Designing Has Never Been Easier!", declares VariCAD s.r.o. in conjunction with the company's new release of VariCAD 2017-2.0 3D mechanical CAD system and VariCAD Viewer/Converter. The new VariCAD CAD system provides a plethora of useful improvements to users, such as support for 4k resolution (UHD); exploded views of assemblies; 3D texts extruded into space; list of materials used during the design process; new possibilities for creating solids; changes in the 3D kernel that accelerate operations on complicated solid trees; changes in STEP input/output of files; a new, complex system of item numbers management; new transient/temporary construction lines in 2D drawing and 3D sketching; rebuilt methods of 2D filleting; chamfering and corner creation and much more. The companion product, VariCAD Viewer 2017-2.0, likewise has been updated. This free viewer, converter and printing software facilitates working with 2D DWG, DXF, 3D STEP and 2D/3D VariCAD file formats; converting DWG to DXF and vice-versa and STEP to 3D IGES or STL formats; and printing 2D DWG, DXF or VariCAD formats and batch printing or conversions.

http://varicad.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

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!

```
 *
 * Usage: $(obj).insertAtCursor(text);
 *   obj = a textarea or textfield
 *   text = a string to insert
 */
$.fn.extend({
  insertAtCaret: function(myValue){
    return $(this).each(function(){
      // If target element is hidden, don't do it.
      if ($(this).is(':hidden') || $(this).parents(':hidde
        return;
      }
      if (document.selection) {
        this.fo
        sel = d
        sel.tex
        this.fo
      }
      else if (
        var sta
        var end
        var scr
        this.va
        this.fo
        // High
```

The Full Stack Project

Deploy an app from the ground up with Linux and open-source software.

JOHN S. TONELLO

PREVIOUS
New Products

NEXT
Feature: Technical Writing with Pandoc and Panflute

Do a quick search of Indeed, Dice or your favorite job site, and you'll quickly discover the massive demand for full stack developers, with salaries on the rise for talented men and women who are equally comfortable with back-end and front-end development. Linux and free, open-source software provide all the tools you need to do your own full stack experimenting, and the project described here will give you a taste of doing it all.

In this project, you'll build a little app for keeping track of all the books you've read and want to read. Think of it as your own digital library tool not tied to Amazon or any other online book shop. You'll start by setting up a LAMP server (on a virtual machine, bare metal or a cloud-based virtual private server); tap into the public Google Books API; take advantage of GitHub; and write a little PHP, jQuery, CSS and SQL. At the end, you'll have a working app and a better understanding of the big-picture thinking of a full stack developer.

Getting Started

For this project, you'll use some common, readily available software that will be familiar to anyone who's tinkered with Linux. For the OS, you'll deploy Ubuntu 16.04 LTS, but feel free to use any flavor you prefer. The rest of the software you'll use is well tested on literally any flavor of Linux you can think of.

You'll install apache2, MySQL, PHP, git (so you can pull and push your project from and to GitHub) and Adminer, a handy browser-based tool for graphically managing databases. If you happen to have a domain name you want to point to your server, that will be helpful, but I won't cover domain configuration in detail. I recommend using a domain because it'll help you dig into the apache2 configuration, an important full stack developer skill.

Once the host is set up (and secured), you'll create accounts on GitHub and Google Cloud Platform. The former isn't strictly necessary if you're just cloning my jtonello/booklist repository, but again, I think a tool like GitHub will serve you well as you begin to grow your code base and start branching out. Google Cloud Platform is required so you can take advantage of Google's massive books database at the heart of this project. Real-time API calls to Google Books will return a flood of data that you can use for this and other projects.

Finally, you'll start writing the application itself, something I've called "My Library". You can name it what you want and even open it up to multiple users. Just like any software developer, you'll focus on how to make your app work for any number of users, not just as a one-off bit of stuff for yourself. It'll have PHP at its core, and you'll use the JavaScript framework jQuery, create a user-friendly layout with CSS, decode some JSON and do some simple debugging and refactoring.

Before wrapping it all up, I'll offer a few ways you can build out your app and make it your own. You can change the layout; make a mechanism for adding users; link your tool to booksellers like Amazon, Barnes & Noble or AbeBooks.com; or follow your own path.

The Host

Of course, it all begins with the host—the server that will become the home for your app. Ironically, many developers who are quite good at software development never really got their hands on setting up a server. Maybe their company has an operations group responsible for that, or maybe they were just given credentials to some server and never had to think about it too much.

To get started setting up the server, download the Ubuntu 16.04 .iso for the architecture you have (32- or 64-bit). If you plan to build this project on a virtual machine—anything from VMware to VirtualBox to Proxmox—you'll use this .iso file to install the OS. If you're using Proxmox or other free KVM-based VM host, you can use a prebuilt Linux container, or LXC. (For more on this, check out my Tiny Internet Project; see Resources for the URLs.)

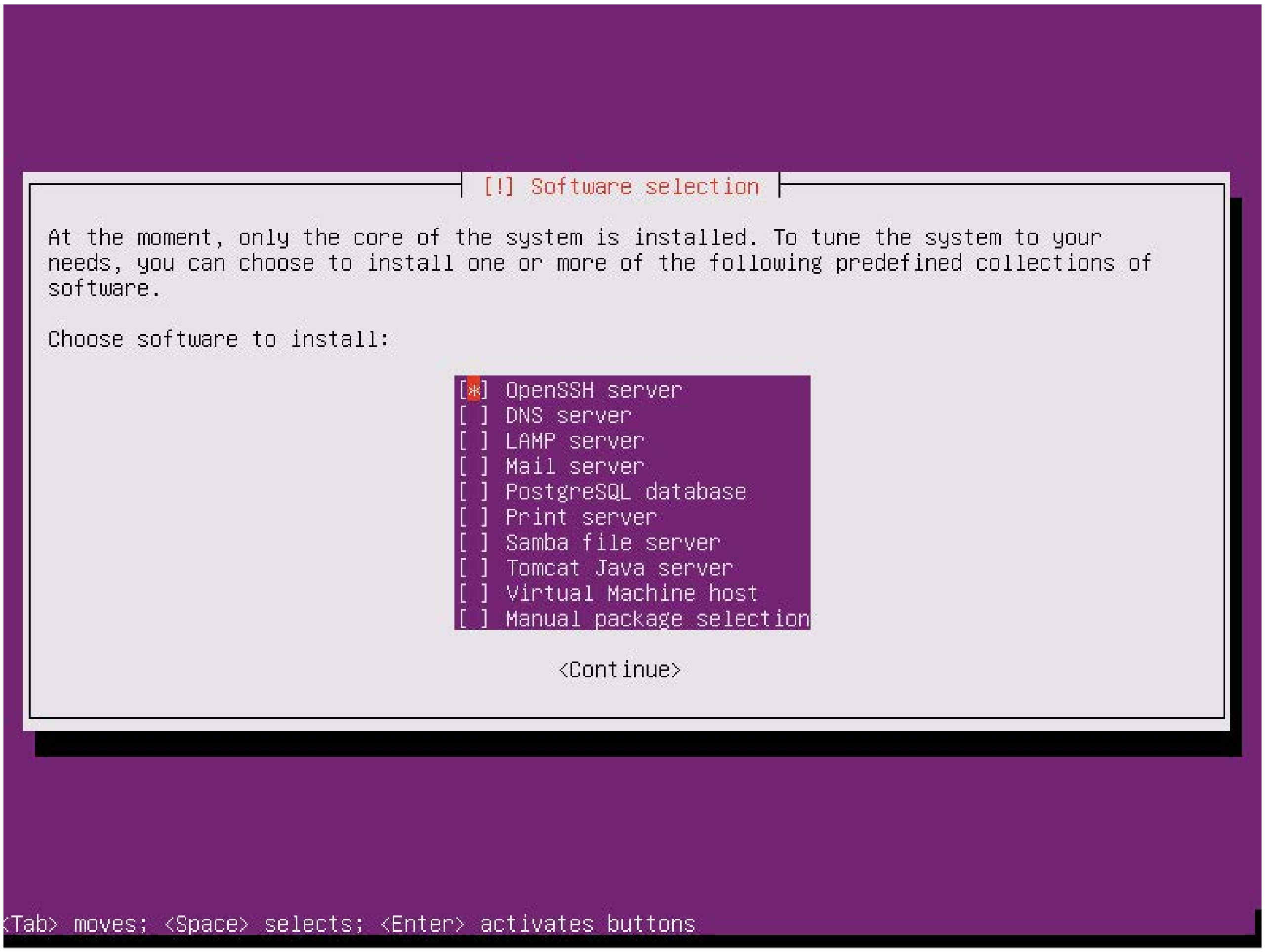
If you're turning an old PC or laptop into your server, you'll need to burn the .iso to a USB thumbdrive or DVD and boot your system. If all you have is an old 32-bit PC, no matter. It'll work great for this project too. Just make sure to download the right version of the Ubuntu .iso for your platform.

More recently, I've started experimenting with VPS hosts—virtual private servers. Unlike web hosting, VPS offer you very inexpensive full-fledged Linux hosts. You can choose the Linux flavor you like, get root access to the whole server and pay as little as $3 a month. Some good VPS hosts include DigitalOcean, Linode and VirMach.

I've included a bunch of handy links about VPS hosts and more in the Resources section at the end of this article.

Install the OS

If you've never installed Ubuntu (or whatever Linux distro you're using) from scratch, you'll find the process pretty painless these days. When you boot from the .iso, just follow the on-screen instructions. When you get to the screen asking which packages to install, select only ssh-server. If you're short on time, you can select the LAMP stack option too and skip ahead to the Apache configuration.



```
[!] Software selection

At the moment, only the core of the system is installed. To tune the system to your
needs, you can choose to install one or more of the following predefined collections of
software.

Choose software to install:

                         [*] OpenSSH server
                         [ ] DNS server
                         [ ] LAMP server
                         [ ] Mail server
                         [ ] PostgreSQL database
                         [ ] Print server
                         [ ] Samba file server
                         [ ] Tomcat Java server
                         [ ] Virtual Machine host
                         [ ] Manual package selection

                              <Continue>

<Tab> moves; <Space> selects; <Enter> activates buttons
```

Figure 1. The Ubuntu Package Installation Screen

Networking

If you're creating your host on a VM on a computer attached to your home network, the setup probably will grab an IP address from your router's DHCP server. If not, you'll have to assign an address manually. Of course, any hosted solution will provide you with the IP address.

Once the installation is complete, log in as the user you created during setup and check your IP address:

```
# ifconfig
```

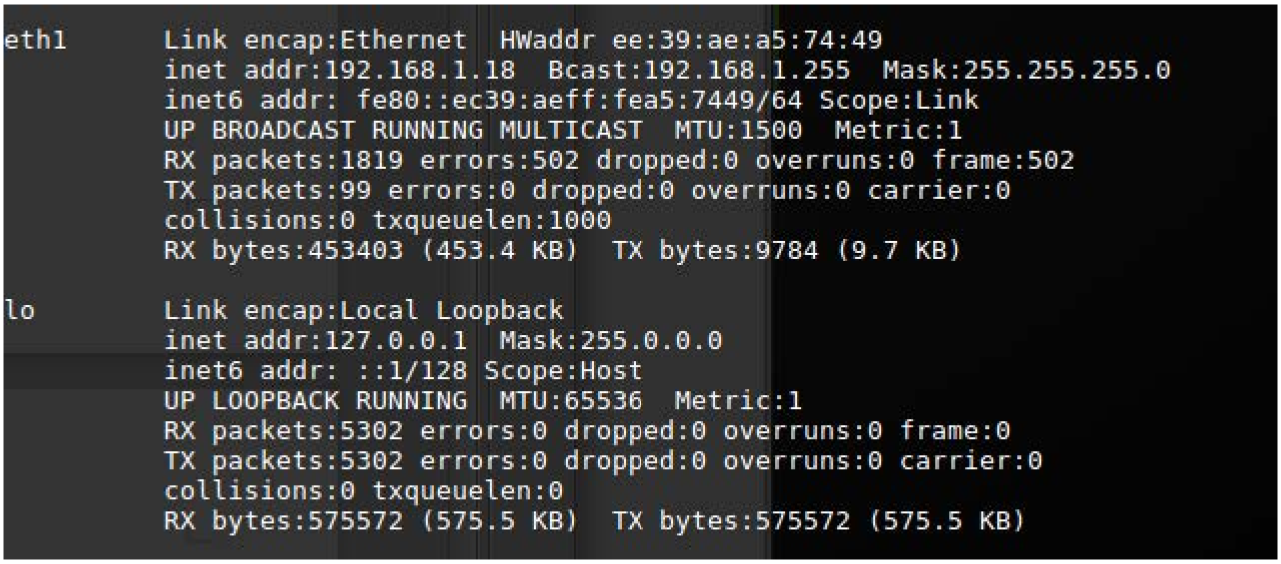
Note the IP address of your host and jot it down. Next, test your network connectivity with a quick ping:

```
# ping google.com
```

If you don't get results, something's wrong with your networking. Check /etc/network/interfaces and look at the Resources link on how to troubleshoot.

If you haven't already added a user account with root privileges (Ubuntu creates this user during install), go ahead and do that now:

```
# adduser jjohnson
```



```
eth1      Link encap:Ethernet  HWaddr ee:39:ae:a5:74:49
          inet addr:192.168.1.18  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::ec39:aeff:fea5:7449/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1819 errors:502 dropped:0 overruns:0 frame:502
          TX packets:99 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:453403 (453.4 KB)  TX bytes:9784 (9.7 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:5302 errors:0 dropped:0 overruns:0 frame:0
          TX packets:5302 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:575572 (575.5 KB)  TX bytes:575572 (575.5 KB)
```

Figure 2. Running `ifconfig` returns your IP address and active network interface.

Answer the on-screen questions, starting with a password. When you're back at the prompt, add your new user to the sudo and www-data groups:

```
# adduser jjohnson sudo www-data
```

Log out of the root shell and log in with your new user account. From now on, you'll prefix your commands with `sudo` instead of operating in the more hazardous superuser account.

Test your sudo privileges by making sure openssh-server is installed. If you selected it during install, it'll be there. If not, simply run:

```
$ sudo apt install openssh-server
```

Next, run updates:

```
$ sudo apt update && sudo apt upgrade -y
```

Now that your system is up to date, install `iptables-persistent` so you can set up firewall rules. iptables is a very common and robust way to manage network access to your host. If you're toying around in a lab on your home network, setting up iptables isn't strictly necessary, but anywhere else, it's important to protect your system from intrusion:

```
$ sudo apt install iptables-persistent
```

The default iptables rules are set to make your host wide open, accepting traffic from anywhere to any port on your machine. That's dangerous. You'll want to prevent access to your host from outside, so you'll create iptables INPUT rules. So, let's say you want to allow traffic on port 80 (http) from everywhere, and all traffic, regardless of port, to the host from your private network, which might be any machine with an IP address in the range of 192.168.1.1 to 192.168.1.255. You'd add rules like this:

```
$ sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT
```

And then add:

```
$ sudo iptables -A INPUT -p tcp -s 192.168.1.0/24 -j ACCEPT
```

The rules are saved in /etc/iptables/rules.v4. To apply them, you can use iptables' `save` command, but for now, just reboot to make these rules permanent.

Of course, these are very basic iptables examples, but enough to get you started. Adding similar rules in /etc/iptables/rules.v6 will protect your host from unwanted IPv6 traffic too. Check the Resources section at the end of this article for links to other good references.

Install Apache, MySQL and PHP

If you selected the LAMP software package during installation, you can skip this next part. If not, manually install the apache2 web server, the MySQL database and PHP with the following commands:

```
$ sudo apt install apache2ctl
$ sudo apt install mysql-server
$ sudo apt-get install php libapache2-mod-php php-mcrypt
 ➥php-mysql
```

The last command installs PHP and gives it hooks into both Apache and MySQL so it serves up pages properly.

Apache configuration files live in /etc/apache2/. The key files are apache2.conf and /etc/apache2/sites-available/000-default.conf. Also, look in /etc/apache2/sites-enabled and see how the 000-default.conf file there is just a link to the file in sites-available:

```
$ ls -la /etc/apache2/sites-enabled
```

Open and look at each of these files, which contain default settings for a single default website (see Resources for more details).

Test Your Connection

Before going on to the next step, you should test to make sure that you have network access to your host machine, that your web server is running, that you can shell in from another machine and that PHP is working properly.

First, test the web server. When you installed apache2, it set up a default website and enabled it for you. If your host is 192.168.1.18, you can point your browser to http://192.168.1.18, and you should see the default "It works!" page.

Next, shell in to your host. If you're coming from another Linux machine or a Mac, use the built-in terminal application. If you're on Windows, download PuTTY and use that to shell in (see Resources for more on PuTTY):

```
$ ssh 192.169.1.18
```

To test PHP, shell in to your new web server and create a small info.php file in /var/www/html that contains the simple `phpinfo` function.

Use vi or nano, built-in Linux text editors, to create and edit the file:

```
$ sudo vi /var/www/html/info.php

<?php

    phpinfo();

?>
```

Save and exit, and point your browser to http://your-host/info.php. You should see the welcoming purple PHP configuration page. The fact that it shows up is all you need to know that PHP is installed and ready to go. Best to delete the info.php file before proceeding.

Database and Repository Tools

Now you're ready to install Adminer, a handy browser-based tool for managing databases and tables, and git, the document repository you'll use to pull down the My Library app itself:

```
$ sudo apt install adminer
$ sudo apt install git
```

The package that installs Adminer places it in /usr/share/adminer, so you have to do a couple things to make it available from your web server. First set up the configuration file:

```
$ sudo echo "Alias /adminer.php /usr/share/adminer/adminer.php"
 ➥| sudo tee /etc/apache2/conf-available/adminer.conf
```

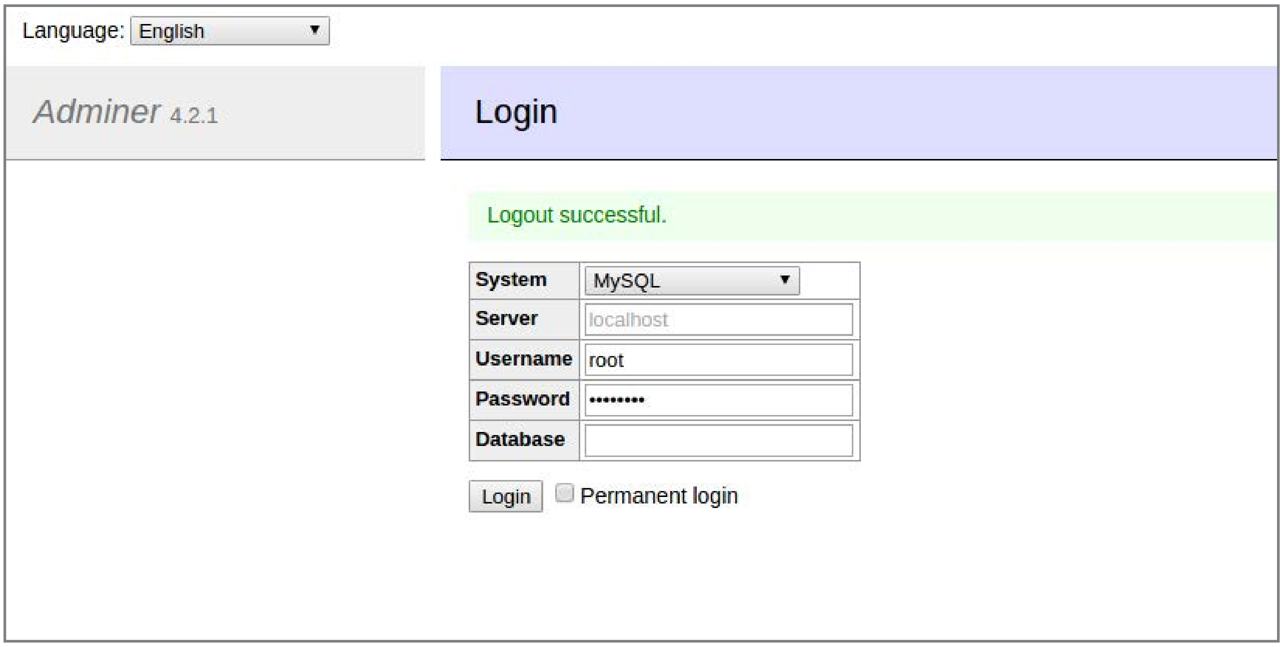
This copies the Adminer configuration to your Apache configuration directory. Enable it with:

```
$ sudo a2enconf adminer.conf
```

Create a symbolic link from the adminer directory to your website:

```
$ sudo ln -s /usr/share/adminer/adminer/ /var/www/html/adminer
```

If all goes well, you can point your browser to http://your-host/adminer and log in with the user name root and the password you created when you installed mysql-server above.



Logout successful.

System	MySQL
Server	localhost
Username	root
Password	••••••••
Database	

Figure 3. Use the credentials you created during the mysql-server installation to log in to Adminer.

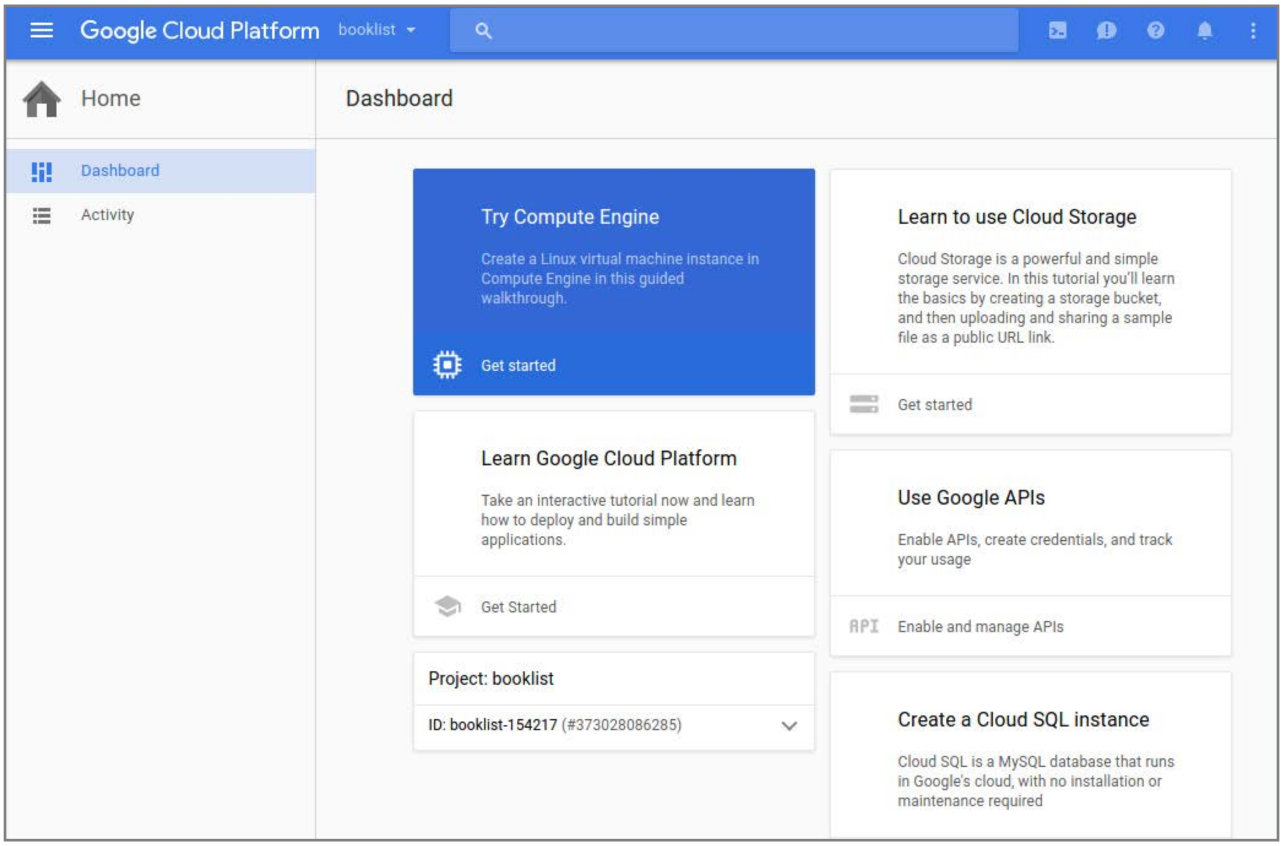
Set Up External Accounts

At the heart of the My Library app you're about to deploy is Google's booklist database, and in order to use it, you need a Google Cloud Platform account. If you have a Gmail account, adding a cloud account is free and simple. Just go to https://console.cloud.google.com and set yourself up.

You'll notice that this account gives you a lot more than just access to Google APIs, including Google's cloud solution, Compute Engine. For now though, you just want to enable and manage the book API. On the main dashboard page, look for the link under "Use Google APIs".

Click "Library" in the left-hand column or the "Enable API" link at the top of the page to see the full list of available APIs. Search for "book", and click the "Google Books API" link. At the top of the page, click the "Enable" button.

The API is almost ready to use, but first you need to create credentials. Click the "Create credentials" button and answer the questions. You'll be



Try Compute Engine

Create a Linux virtual machine instance in Compute Engine in this guided walkthrough.

Get started

Learn to use Cloud Storage

Cloud Storage is a powerful and simple storage service. In this tutorial you'll learn the basics by creating a storage bucket, and then uploading and sharing a sample file as a public URL link.

Get started

Learn Google Cloud Platform

Take an interactive tutorial now and learn how to deploy and build simple applications.

Get Started

Use Google APIs

Enable APIs, create credentials, and track your usage

Enable and manage APIs

Project: booklist

ID: booklist-154217 (#373028086285)

Create a Cloud SQL instance

Cloud SQL is a MySQL database that runs in Google's cloud, with no installation or maintenance required

Figure 4. The Google Cloud Platform API Dashboard

calling the API from a "Web server" and using "Public data".

When you click "What credentials do I need", you'll be taken to a screen and shown your new API key. Copy that down; you'll need it later in your app. Check the Quotas tab to look at your daily limits for this API. The free version gives you 1,000 queries a day. That should be plenty, but if you accidentally build a loop function that makes dozens of calls to the API each second, you'll use up your day's worth in a hurry.

With these pieces in place, I recommend you create a GitHub account, although an account isn't required for this project, only git is. As public repositories go, it's a good one, and it will get you in the habit of thinking about the code you create in versions that can be branched and shared. Keep in mind though that the default for GitHub is public. Your projects will be freely available—and seen—by anyone. If you want your code to be private, you have to pay for that.

Building the Application

So far, you've set up a host, installed the core applications you need to serve up your web app and prepared a public API source for use. It's now time to deploy the code.

Start by setting up the MySQL database and the three tables you'll need to run the application. You can do this from the command line by `ssh`-ing in to your host and logging in to MySQL using the credentials you created:

```
$ mysql -u root -p
```

If you've never done this before though, it can be a little taxing to manage the database and tables from the command line. Instead, use your MySQL credentials to log in to the Adminer tool (http://your-host/adminer).

On Adminer's main dashboard, select "Create new database", give it the name "booklist", and set collation to "utf8_general_ci". Of course, you can call the database whatever you want, but I'll be referencing "booklist" in the PHP code.

Create a new user for the booklist database by clicking the "Privileges" link in the "Server" view. Set "Server" to "localhost" and "Username" to "webuser". Add a password. I like to give a user like this just the privileges it needs, so under the `'booklist '.*` column, check the "Select, Update and Insert" boxes under Table and Column, and "Delete" under Table. Click save.

With the database created, you need to create three tables: one to hold information about each book; one to hold information about your library, and one for users.

The books table will hold the core information about each book, and the library table will hold each user's information about that book—namely, whether it's been read or wishlisted.

The books table has the following fields, types and sizes, with the ISBN used as the unique ID field and primary key:

```
isbn varchar(40)
title     varchar(120)
author    varchar(120)
selfLink  varchar(255)
publishddate   year(4)
saveddate timestamp[CURRENT_TIMESTAMP]
thumbnail varchar(255)
description    mediumtext
```

Most of these fields are self-explanatory, but it's worth describing `selfLink` and `thumbnail`. The Google API offers up these fields and it's nice to save them locally. `selfLink` is a URL that points to the book in the Google database, which displays details, bibliographic information and more. The `thumbnail` field is a URL that points to the image the Google API has associated with the book. The thumbnail URLs will give you a good way to dress up your library app and make it more user-friendly.

Next, create the "library" table and its five fields:

```
isbn    varchar(40)
user_id varchar(50)
readit  tinyint(1)
wishlist    tinyint(1)
saveddate   timestamp[CURRENT_TIMESTAMP]
```

The user_id field is large enough to store an email address, which will be each user's unique ID. The primary key though, will be a composite of isbn and user_id. That's so different users can tag the same book. To create the composite key, click "SQL command" in the "Table: library" view and execute the following SQL:

```
ALTER TABLE library DROP PRIMARY KEY, ADD PRIMARY
 ➥KEY(isbn,user_id);
```

The readit and wishlist fields will contain either a 1 or 0, integers set when users add books to their libraries.

Finally, create the users table, which sets the user_id field as the primary key:

```
user_id   varchar(50)
first_name    varchar(30)
last_name     varchar(30)
```

After you've created the users table, click the "New item" link to create your first entry. Use your own email and name. Set the password type to "password" so you can use the MySQL PASSWORD() function to parse it later in your PHP. I don't fully implement a password look-up function in this app, but you can add that on your own to allow users to log in securely.

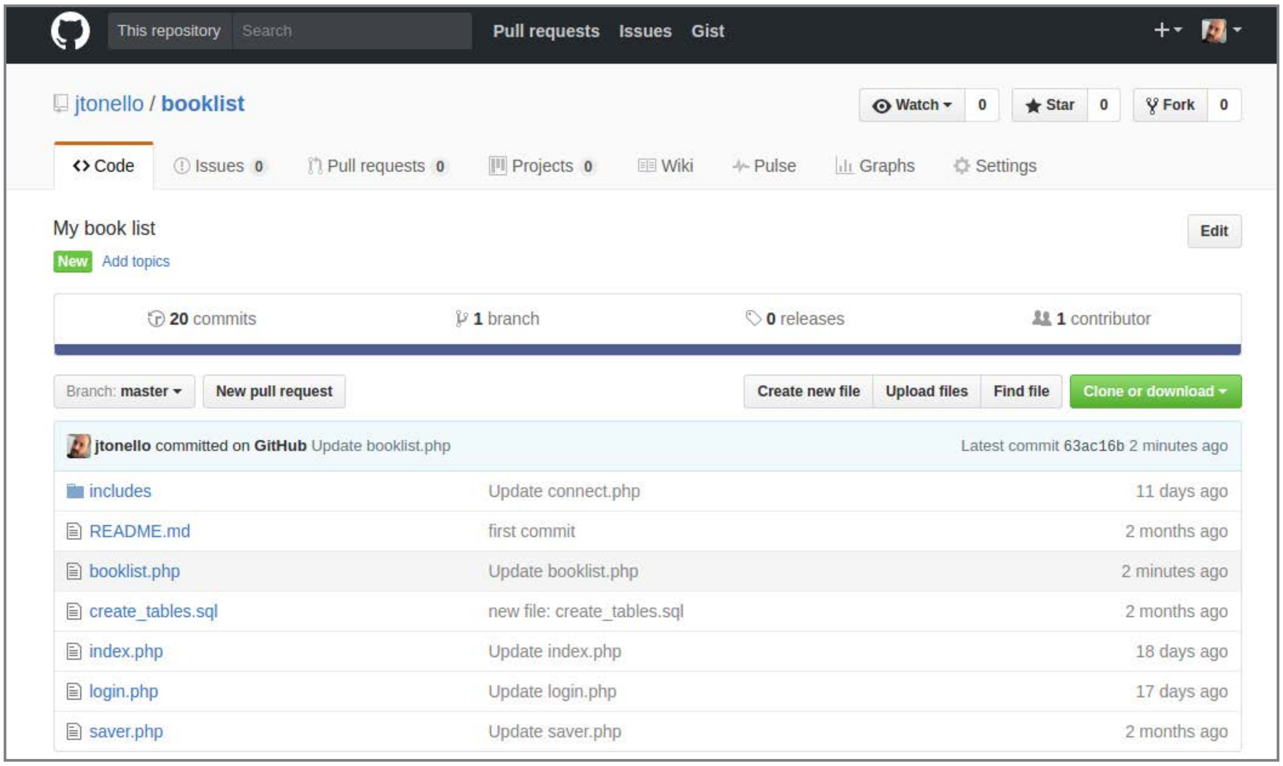
Download the Application Files from GitHub

With the database and tables set, you can start using them to store data. In order to do that, you'll need to make some calls to the Google API and store the results in the database.

Move to the root of your Web server (cd /var/www/html), and use git to clone my booklist repository:

```
$ sudo git clone https://github.com/jtonello/booklist
```

This should create a folder with all the files in /var/www/html/booklist. If not, move the contents to that path and then change the ownership to



jtonello / booklist

My book list

20 commits | 1 branch | 0 releases | 1 contributor

jtonello committed on GitHub Update booklist.php — Latest commit 63ac16b 2 minutes ago

includes	Update connect.php	11 days ago
README.md	first commit	2 months ago
booklist.php	Update booklist.php	2 minutes ago
create_tables.sql	new file: create_tables.sql	2 months ago
index.php	Update index.php	18 days ago
login.php	Update login.php	17 days ago
saver.php	Update saver.php	2 months ago

Figure 5. The Complete File List, Available in GitHub Repository jtonello/booklist.

www-data and give it permissions so you can edit it:

```
$ sudo chown www-data:www-data /var/www/html/booklist -R
$ sudo chmod 775 /var/www/html/booklist -R
```

By changing the permissions to 775, you'll be able to edit the files as you, not as sudo, since you added yourself to the www-data group earlier.

Take some time and look over the files, either on the GitHub page or in your newly created directory. Each page includes documentation that I won't repeat here, but here's some of the logic behind it all:

- /includes — contains a common connection script (connect.php) and a functions.php file that holds function code used on other pages. Edit connect.php to add your MySQL credentials.

- booklist.php — this is the API page, which features the book search that connects to the Google book database. Edit this to add the

Google API key you set up earlier.

- delete.php — this page executes queries to delete database entries.
- index.php — the main page, which features tabs for Search, Read and Wishlist.
- login.php — the login form page.
- saver.php — this page executes queries to save data to the MySQL database.
- create_tables.sql — SQL that can be used to create the MySQL database tables.

How It Works

When users go to http://yourhost.com/booklist, they're greeted by a login screen. When the form is submitted, the app checks the value of the user name field against a static value saved in /var/www/html/booklist/login.php. Later, you might attempt to make this database-driven so more people can use it. This will get you started though.

Successful login takes the user to the main My Library view, which includes three tabs generated by jQuery UI: Search, Read and Wishlist.

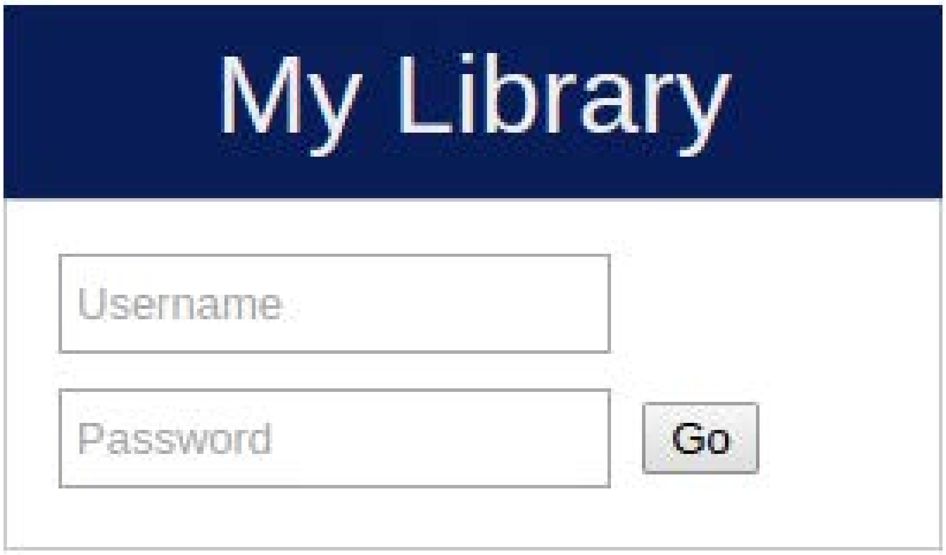
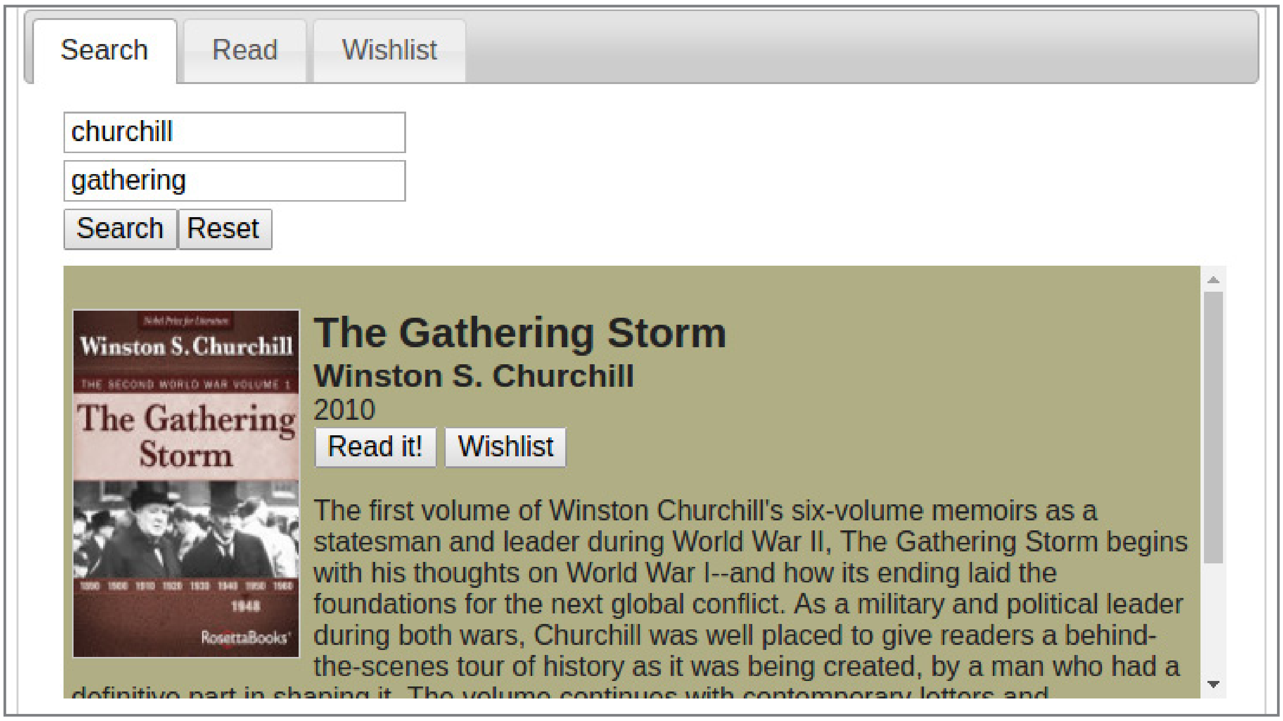


Figure 6. The initial login screen determines the user.



The Gathering Storm
Winston S. Churchill
2010

The first volume of Winston Churchill's six-volume memoirs as a statesman and leader during World War II, The Gathering Storm begins with his thoughts on World War I--and how its ending laid the foundations for the next global conflict. As a military and political leader during both wars, Churchill was well placed to give readers a behind-the-scenes tour of history as it was being created, by a man who had a definitive part in shaping it. The volume continues with contemporary letters and

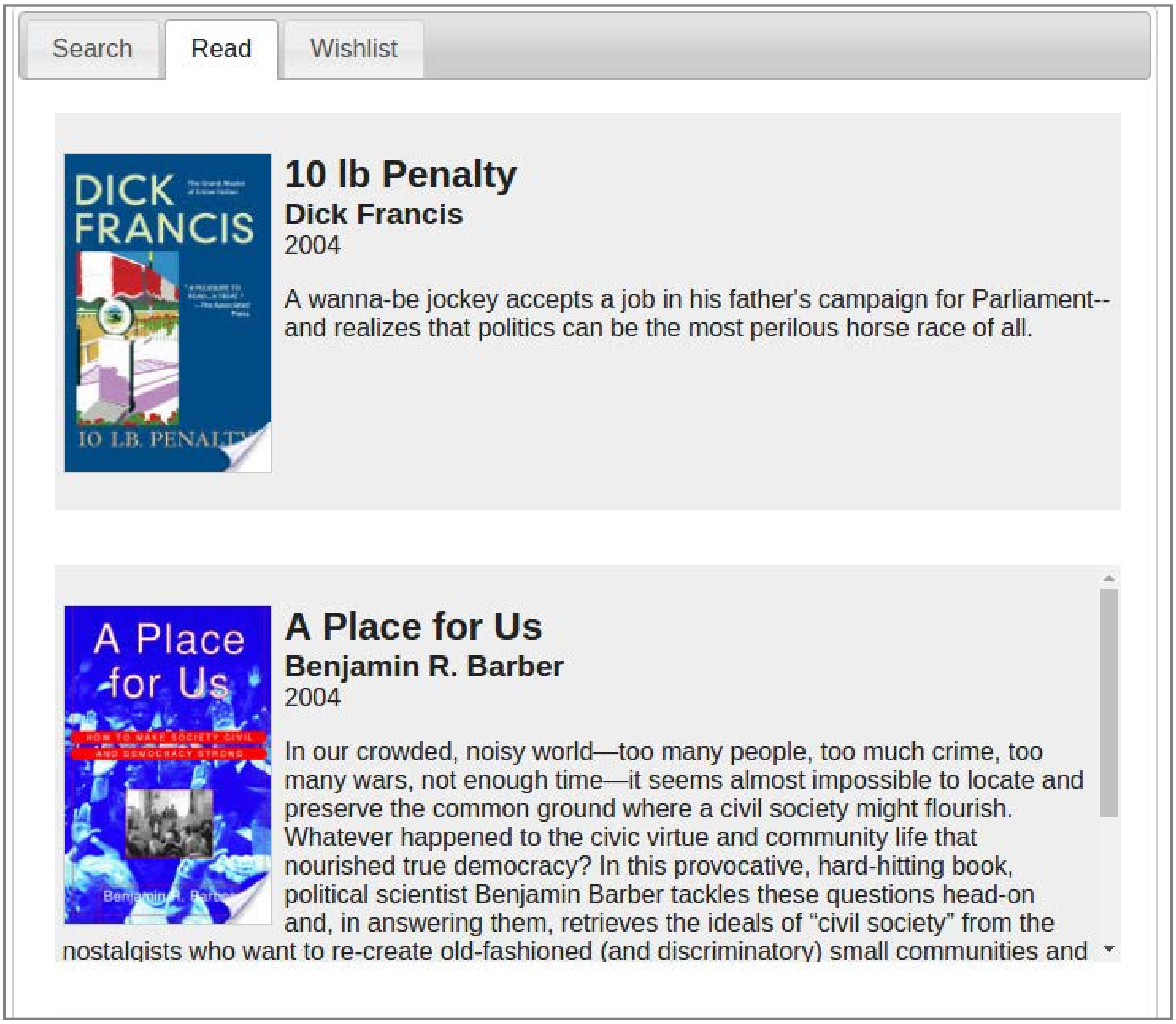
Figure 7. A simple search returns books from the Google database via the Google API.

The Search page features the author and title search, and the search results. The Read and Wishlist pages show previously saved books. Search results appear on the Search tab through an AJAX function; clicking the buttons in each book block will save the result in the MySQL database and simultaneously change its color.

Once saved, these books now appear under the appropriate tabs: Read or Wishlist.

Each time a user clicks the Read and Wishlist buttons, behind the scenes, two database queries are executed. One saves the book's ISBN, title, author, description and other information to the MySQL table called books, and one saves the ISBN, the user's ID, the Read or Wishlist flag, and the date to the table called library.

The information stored in the books table could be limited to just the ISBN, which could then be used to pull the title and other information from the Google database, but doing it that way is slow and makes for an awkward user experience. By saving key elements from the Google database locally, you can see the key information for each book in fast-loading tabs.



10 lb Penalty
Dick Francis
2004

A wanna-be jockey accepts a job in his father's campaign for Parliament--and realizes that politics can be the most perilous horse race of all.

A Place for Us
Benjamin R. Barber
2004

In our crowded, noisy world—too many people, too much crime, too many wars, not enough time—it seems almost impossible to locate and preserve the common ground where a civil society might flourish. Whatever happened to the civic virtue and community life that nourished true democracy? In this provocative, hard-hitting book, political scientist Benjamin Barber tackles these questions head-on and, in answering them, retrieves the ideals of "civil society" from the nostalgists who want to re-create old-fashioned (and discriminatory) small communities and

Figure 8. Previously saved items appear in the Read tab.

The database tables are normalized, meaning the only overlapping data they contain is the shared ISBN key. Queries that show Read or Wishlist books perform a join to link the data in the separate tables together. That also means that no matter how many users you have, you'll only ever need to store a single copy of a book's details. You also can have unlimited saved information for each user.

Make It Your Own

You now have a working application, created from the ground up. You've installed a Linux distro from scratch, configured the firewall and packages, and finally deployed the application. By building everything, you've hopefully gotten a taste of how a full stack developer thinks—and works.

Of course, the My Library application is far from perfect and there's plenty of opportunity for you to make it better. For example, you might create a better login function that makes the tool usable to any number of users. You also could change the page layouts by tweaking the CSS, or you could improve how the contents of the Read and Wishlist tabs load behind the scenes. Maybe you'll try to branch the app and make a different version available in your own GitHub repository. The point is, don't be afraid to experiment and make it your own.■

John S. Tonello is the Director of IT and Communications Manager for NYSERNet, New York's regional optical networking company, serving the state's colleges, universities and research centers. He's been a Linux user and enthusiast since building his first Slackware system from diskette more than 20 years ago. You can follow him @johntonello.

Resources

"The Tiny Internet Project" Series by John S. Tonello:

- Part I: http://www.linuxjournal.com/content/tiny-internet-project-part-i
- Part II: http://www.linuxjournal.com/content/tiny-internet-project-part-ii
- Part III: http://www.linuxjournal.com/content/tiny-internet-project-part-iii

Ubuntu Downloads: https://www.ubuntu.com/download

VPS Hosts:

- https://www.digitalocean.com
- https://www.linode.com
- https://virmach.com

Networking: http://askubuntu.com/questions/431682/how-do-i-use-etc-network-interfaces-instead-of-network-manager

iptables Rules:

- https://help.ubuntu.com/community/IptablesHowTo
- https://www.cyberciti.biz/tips/linux-iptables-examples.html
- https://www.digitalocean.com/community/tutorials/iptables-essentials-common-firewall-rules-and-commands

Apache Configuration: https://www.digitalocean.com/community/tutorials/how-to-configure-the-apache-web-server-on-an-ubuntu-or-debian-vps

Install PuTTY: http://www.putty.org

Install Adminer: https://www.leaseweb.com/labs/2014/06/install-adminer-manually-ubuntu-14-04

Google Cloud Platform: https://console.cloud.google.com

GitHub: https://www.github.com

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

RETURN TO CONTENTS

SUPERMICRO®

MARKETPLACE

Powered by Silicon Mechanics

Broad Selection

Zero Defects

3 YEAR WARRANTY

3-Year Warranty

Your Source for Supermicro Platform Technology

Twin, TwinPro & BigTwin

High-density, high-value servers

FatTwin

Highest performance per watt

SuperStorage

Flexible and efficient storage

1U Servers

Entry & enterprise 1U form factor servers

2U Servers

Flexible 2U form factor servers

3U+ Servers

All 3U and larger form factor servers

Ultra Servers

Unrivaled performance, flexibility & scalability

MP Servers

Servers based on the Intel® Xeon® E7 Product Family

SuperWorkstations

Server-grade performance at your desk

Talk to a Supermicro Expert! 866.352.1173

Technical Writing with Pandoc and Panflute

Create an automated publishing pipeline by writing Pandoc filters in Python.

LEE PHILLIPS

PREVIOUS
Feature: The Full Stack Project

NEXT
Guest EOF

Those of us of a certain age, and with the need to write technical papers, mathematical or otherwise, are likely to be intimate with TeX and LaTeX. We studied Donald Knuth's *TexBook*, climbed the ferocious learning curve and were pleased with the result. Our papers may or may not have been any good, but they certainly looked good.

Then, the web took over the world, seemingly overnight. Some of us wanted our work to live in this new, online environment, so we learned a wholly new system of markup. Then we learned a third language, CSS (Cascading Style Sheets), and maybe even a little JavaScript.

We still wrote our real work in LaTeX, while turning to HTML for our hobbies. If we wanted to put one of our papers on the web, we resorted to one of several options for translation from LaTeX to HTML. None of these were excellent, but they could produce some approximation of how the paper was supposed to appear. Of course, we always could translate our papers by hand, but most people were, naturally, galled by the duplication of effort this entailed.

Since TeX is a Turing-complete language, it is generally impossible to translate it into a declarative markup system like HTML. Several projects address this problem by defining a subset of LaTeX allowed for input. A better approach is that followed by Docbook, Tbook, Pandoc (http://pandoc.org/index.html) and some others: define a more general markup language, or system of XML tags, and translate that into any of the desired targets. Pandoc, one of the subjects of this article, offers more than that, promising to translate between many possible pairs of formats. However, its parsing of some of these, including, critically, LaTeX, is faulty or incomplete. This needn't concern us, because the best way to unlock Pandoc's power is by using its "native" input format, which is an extended dialect of Markdown. Pandoc can translate this into a really impressive set of output formats, including not just LaTeX and HTML, but even ODT and docx, so we can communicate with colleagues unfortunate enough to be stuck using Word. Pandoc is a mature, free software project; a recent version probably is available through your system's package manager. And if you use LaTeX, you probably know that the best way to outfit yourself is by installing the Tex Live distribution, either through your package manager or directly from the Tex Users' Group source (https://www.tug.org/texlive).

If you are familiar with Markdown, you may be skeptical of the abilities of a system based upon it, as the original Markdown is too anemic for any type of complex or technical writing. However, Pandoc's extended version of Markdown, while retaining its simplicity for simple things, allows you to include LaTeX math, bibliographic references, internal and external hyperlinks, tables, language-specific syntax highlighting, bulleted and automatically numbered lists, images with captions and much more:

```
Here is a simple example of Pandoc's Markdown.

That was the first paragraph. This is the second.
It was originally based on email conventions.
You can have *italics*, **boldface**, and
~strikethrough~~ text.

Creating hyperlinks is as easy as
[example.com](http://example.com);
you can also use your
[BibTeX keys][@winterberg2004] this way.

Insert LaTeX equations directly, like $e^{iπ} = -1$
```

In addition to all this, Pandoc has the ability to be extended and customized through filters and templates—and that brings me to the real subject of this article.

Filters

Pandoc works by translating its input into an internal representation, performing certain transformations on the representation and then translating the result into the desired output format. Because of this architecture, where parsing, transformation and output are decoupled, developers can add, for instance, the ability to translate a new format by just writing a module for parsing that format into the internal structure. And, they can support a new output language just by writing a translation from the internal representation into that language.

Crucial to the advanced use of Pandoc is the idea of a filter. A filter is a program, invoked with a flag on the command line, that steps in after Pandoc has parsed its input and makes changes to the document's internal representation. The filter can be indifferent to the eventual output format, or it can do different things depending on the target. For example, when creating HTML, you may want to alter the behavior of footnotes so that the conventional superscript with note at the bottom is changed to revealing the footnote text when the reader hovers over the footnoted word.

Some of the useful things that Pandoc does out of the box are implemented as filters. Academic users depend on adding the flag `--filter pandoc-citeproc`, along with the `--bibliography` flag to indicate the location of a BibTex database file. (The system can handle EndNote, MEDLINE and several other database formats as well.) This filter takes keys from the BibTex file, turns them into citations in your chosen style and adds a bibliography to the end. It replicates the abilities of LaTeX/BibTex to handle reference data stored in a BibTex database, but it can handle HTML and other output formats besides TeX.

Pandoc is written in Haskell, which makes that the native and most natural language in which to write Pandoc filters. But, knowledge of this language is not very widespread. Since Haskell and I are at most distant acquaintances, I turned to panflute (http://scorreia.com/software/panflute), a package for writing Pandoc filters in Python. There are others, but panflute is somewhat easier to use, making the potentially arcane knowledge of Pandoc filters accessible to the average developer. You can install panflute with pip, the Python package manager.

The hardest part of getting up to speed with panflute is learning about its data structures, which shadow the data types internal to Pandoc. But once you become familiar with it, you can do powerful things with very few lines of code. Before I get to the case study that became the impetus for writing this article, let's have a look at a few simple examples to get the general flavor of Pandoc filters in Python.

Suppose you've sprinkled your article with italics and boldface, but the journal's editor has just informed you that their house style does not allow bold text. You could resort to a regular expression substitution, but this is notoriously error-prone, especially when applied to textual markup. Here's a Pandoc filter, using panflute, that changes all instances of bold text to italic, while leaving italic text as is:

```
#!/usr/bin/python3.5
import panflute as pf

def action(elem, doc):
    if isinstance(elem, pf.Strong):
        return pf.Emph(*elem.content)

if __name__ == '__main__':
    pf.toJSONFilter(action)
```

If you save this program as, say "myfilter.py", you can say `cat file | pandoc --filter myfilter.py` to get "file" translated into HTML, with all the boldface turned into italics. You can try it on the Markdown sample above and experiment with different output formats (using the `-t` flag) to verify that Pandoc will use the correct markup for italics in each case.

This example embodies the basic pattern used in all Pandoc filters. The final two lines cause the program to behave as a filter, walking through each element in the input document and applying the "action". The action function must have the two arguments shown; the first is the current element, and the second is the entire document. Both refer to Pandoc's internal version of the document, after the input markup is parsed and translated. For each element, you test whether it's boldface, which Pandoc/panflute represents as a "Strong" element, and, if it is, return its content (which can contain arbitrary arrays of other elements) wrapped in the "Emph" element, which is used for italics.

Although Pandoc's extended version of Markdown covers most of the basic elements that an author of technical material is likely to need, you inevitably will wish that it had syntax for something that is not included. There is a potentially endless number of different semantic elements that an author might invent for a document, and no markup language can anticipate them all. In writing for the web, we typically extend the semantics of HTML by defining classes in CSS, for good or ill. In LaTeX, this job usually leads to the creation of macros, which are simple for simple cases but quickly can turn to dark magic. In either case, we resort to inventing a presentational expression of our semantic intent, and working out a special case for each end target that we wish to write for.

Once you know how to write simple Pandoc filters, however, you can leverage some special features of Pandoc's extended Markdown to extend it further by creating your own customized elements. The first of these special features that I discuss here is the ability to add arbitrary attributes to inline code and code blocks.

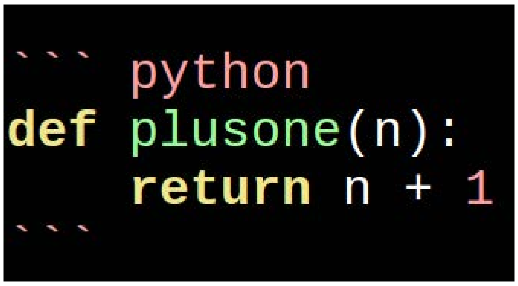
In running text, you can indicate code, commands or similar text by surrounding it with backticks. You usually will see this rendered in a monospace font, and the author almost always wants it rendered literally (no use of ligatures, for example). The Pandoc/Markdown default is to choose the correct semantic markup for the target language, if one exists. For example, the Markdown input ``type `ls` to see a listing`` is translated into the HTML `<p>type <code>ls</code> to see a listing</p>` (Pandoc puts fragments into paragraphs).

If you want to include a full-fledged code sample, you have the choice of several syntaxes. For our purposes here, the backtick syntax will be most convenient: just put your code in a separate paragraph beginning and ending with a line of three backticks.

Pandoc extends Markdown by allowing you to add arbitrary lists of attributes to these elements. For our purposes, one attribute will serve. For inline code, the syntax is ``` `code fragment`{.attribute} ```. For code blocks, it's even simpler (Figure 1).

You are allowed to place any number of spaces between the opening backticks and the attribute name.

Pandoc intends these attributes to indicate the language name, and it uses them to create syntax highlighting for HTML and LaTeX output. You might have noticed that the input example is also syntax-highlighted. This delightful feature is provided by the Vim editor; some details about how to use it are at https://github.com/tpope/vim-markdown.



~~~
``` python
def plusone(n):
    return n + 1
```
~~~

Figure 1. Writing Pandoc/Markdown in Vim

You don't have to use these attributes for language names, however. Since you can get at them through panflute filters, you can use them to extend the language by defining your own elements. I'll give a few examples of elements I defined for my own work.

Soon after I began using Pandoc, I decided I needed a way to include comments in the text that would be passed over and not copied to the output: a way to "comment out" passages. I was sure that something like this was already part of the language, but some Googling revealed that there was no convenient way to accomplish it. Here is how I decided to implement this using filters. First, I decided on the name for my new attribute: "n", for "note". I want to be able to type `` `ignore me`{.n} `` in running text, or, for longer commented-out passages:

~~~
This is text that will be translated, but

``` n
this paragraph,
no matter what it contains,
will just disappear.
```

And we are back to regular input.
~~~

and not see those passages in the output.

The filter that does this is pretty simple:

```
import panflute as pf

def action(elem, doc):
    if (isinstance(elem, pf.Code) or
        isinstance(elem, pf.CodeBlock)):
        if ('n' in elem.classes):
            return []

if __name__ == '__main__':
    pf.toJSONFilter(action)
```

As before, the first line of the action function looks for inline code or CodeBlock elements. When it finds one, it checks whether our special "n" class is in its list of attributes, called "classes" in panflute. The final line of the action function in a filter will be the transformation of the element. In this case, we return an empty list, which accomplishes what we want by simply deleting the element from the document.

One important detail: you should save your panflute filters in files that end with .py, so that Pandoc knows they are Python programs; otherwise, it will assume they are Haskell and print a mysterious error. You don't need a separate file for each filter; you can create multiple filters by including all their relevant conditions in one big action function.

Here's one more simple example that shows what that looks like. It also illustrates output-specific processing: applying different transformations depending on the target. Sometimes I want to put a "publication note" at the beginning of an article on my website to notify the reader that the article may have been updated, for example. But, I don't want these notes to appear in a PDF (via LaTeX) version of the article. So, I want different things to happen depending on the output format. I call this custom element "pubnote"; here is the previous filter with the pubnote rule added:

```
import panflute as pf


def action(elem, doc):
    if (isinstance(elem, pf.Code) or
        isinstance(elem, pf.CodeBlock)):
        if ('n' in elem.classes):
            return []
    if isinstance(elem, pf.CodeBlock):
        if 'pubnote' in elem.classes:
            if doc.format == 'html':
                return pf.convert_text(
                  '<div class = "pubnote">{}</div>'.format(elem.text))
            else:
                return []

if __name__ == '__main__':
    pf.toJSONFilter(action)
```

This shows the main reason to pass the `doc` argument into the action function: it carries with it attributes global to the document as a whole, including, in this case, the output format requested of Pandoc. If this format is HTML, I would like the publication note simply wrapped in a `div` with a certain class, so that I can style it appropriately with my stylesheet; if it's any other format, it can expunge it. I used the function `convert_text` to get this done; this function takes its argument, as if it were a Pandoc input string, and passes it through Pandoc, using the active output format. The example also shows one way to extract the contents of an element, with its text attribute. You can find more information about the Element class and the rest of the panflute API at http://scorreia.com/software/panflute/code.html.

Templates

Your document can define global variables that are carried through the translation process until the end. I've already described one: the output format, culled from the command-line flag, that you can access in your filters as doc.format.

The panflute API extends code blocks to contain YAML data (http://www.yaml.org/start.html). YAML is a data description language, like XML, but less verbose and easier on the eyes. You can use this to define named variables, right in your document, of various types, including lists of values.

Here's a simple example of how I use this facility in my personal set of filters. Sometimes I want to include a short quotation, or epigraph, at the beginning of an article or chapter. The epigraph has two distinct components, which are the quote itself and the person to which it is attributed:

```
~~ epigraph
quote: Simplicity is the ultimate sophistication.
who: Leonardo da Vinci
---
~~
```

I can include this data block anywhere in the document. My convention happens to be to use a row of three tildes, as you can see, but backticks work as well. Panflute will put the variables, defined between the start of the block and the "---" line, into a Python dictionary: `>{'quote': 'Simplicity is the ultimate sophistication.', 'who': 'Leonardo da Vinci'}` The space between the "---" line and the line ending the data block is for optional data.

Here is a filter that processes the "epigraph" blocks. It assumes HTML output, but from the examples above, you will understand how to extend it to handle other formats. I've used the convenience function `yaml_filter`, which is part of panflute and handles the parsing of YAML-extended code blocks. You pass it as an argument to the `toJSONFilter` function that I mentioned before, along with a dictionary of tags. This dictionary associates attribute names with which you tag your data blocks with function names. The first argument to these filter functions will be the dictionary of values parsed from the block, while the second will be the optional data. The element and doc arguments are as you've seen previously:

```
import panflute as pf

def epigraph(options, data, element, doc):
    return pf.convert_text(
     '<div class = "epigraph">{} <span class = "who">{}</span></div>'
       .format(options.get('quote'), options.get('who')))


if __name__ == '__main__':
    pf.toJSONFilter(pf.yaml_filter,
       tags = {'epigraph': epigraph})
```

You also can use the global variables from your YAML blocks in Pandoc templates. The Pandoc examples I've given so far have involved translating fragments of text on the command line. Once you have it installed, you can say, for example, `echo "*Hello*, **world**" | pandoc` to get the phrase translated into the default HTML, or into LaTeX by adding the flag `-t latex`. When creating entire documents, however, you invariably need other material surrounding the text. For web pages, you'll have a header linking to your site's stylesheet and probably much more, and `body` and `html` tags. When using LaTeX, you will need a preamble that defines your document class, loads all the required packages and, quite likely, defines all your personal macros and definitions.

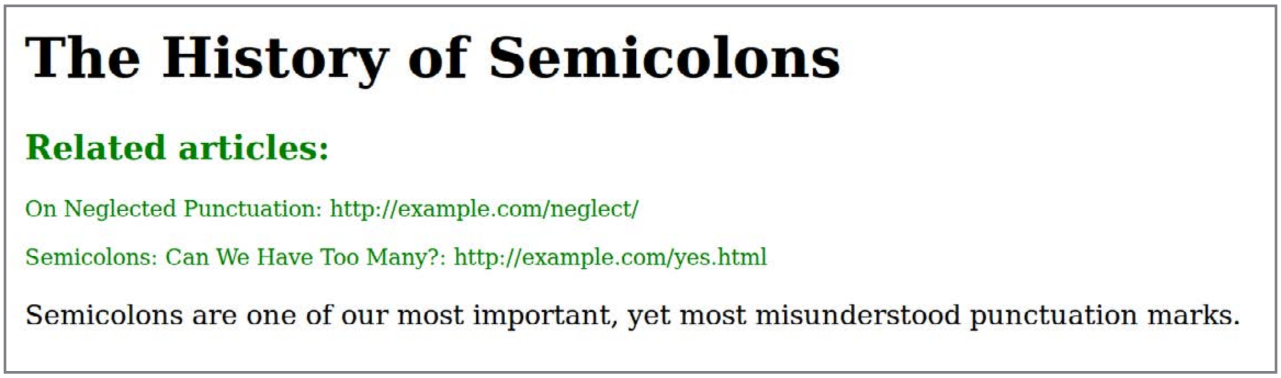
If you supply the `--standalone` flag to Pandoc, it will embed the translated content into a default template appropriate for your chosen output format. These templates are quite capable, including such things as style definitions for the language-specific syntax highlighting that I mentioned previously. However, they are rarely what you want. You inevitably will wind up creating your own templates for real work. Fortunately, this is quite simple. You merely need to take whatever template you already are using for document creation and add placeholders for global variables defined by Pandoc and by you in your document. These are variable names surrounded by dollar signs. The main one, `$body$`, contains the translated text of your document. Other sources for global variables are the YAML-extended code blocks that I introduced above, certain command-line flags, the general flag `--variable`, unnamed YAML blocks and Pandoc title blocks. For those last two, see the Pandoc documentation at http://pandoc.org/MANUAL.html for all the details. Remember to save your templates with filename extensions that match their format: .html for HTML and .latex for LaTeX. Here's an example of an unrealistically simple HTML template, showing the use of two variables. It also serves as an example of three very useful features of Pandoc's template language: conditions, looping over variable lists and extracting fields from variables:

```
<!DOCTYPE HTML>
<html dir="ltr" lang="en-US">
<head><meta content="text/html;charset=utf-8"
      http-equiv="Content-Type" />

    <title>$title$</title>

</head>
<body>
```

```
 <h1>$title$</h1>

 $if(related)$
  <div style = 'font-size: 0.8rem; color: green;'>
   <h2>Related articles:</h2>
   $for(related)$
    <p>$related.title$: $related.url$</p>
   $endfor$
  </div>
 $endif$

 $body$

</body>
</html>
```

Here is the small Pandoc/Markdown document that I'll use as input to this template. It begins with a standard YAML block that Pandoc uses to populate the metadata variables used in the template automatically. This is the syntax that panflute extends to define the YAML-extended data blocks that I used earlier. Notice the terse yet readable syntax for lists and variable attributes defined by YAML syntax:

```
---
title: The History of Semicolons
author: Prof. Lexi Graphical
related:
- title: On Neglected Punctuation
  url:   "http://example.com/neglect/"
- title: "Semicolons: Can We Have Too Many?"
  url:   "http://example.com/yes.html"
---

Semicolons are one of our most important,
yet most misunderstood punctuation marks.
```


The History of Semicolons

Related articles:

On Neglected Punctuation: http://example.com/neglect/

Semicolons: Can We Have Too Many?: http://example.com/yes.html

Semicolons are one of our most important, yet most misunderstood punctuation marks.

Figure 2. Rendered HTML Incorporating Metadata

Figure 2 shows what the resulting HTML document looks like when rendered in a browser.

Automating a Complex Document

Since Pandoc filters, using the panflute package, are just Python programs, they can do more than merely modify the document translation: they can perform any processing you desire. In particular, as Python is a good "glue language", in which it's easy to invoke external processes and perform system functions, you can allow the document to trigger this processing and include the results in the finished product.

This final section is a case study showing how to use Pandoc filters and templates to automate away the tedious and error-prone tasks that happen to be involved in creating a certain complex document. I hope this detailed example will make the principles clear, so that you take away the ability to apply these techniques in creating your own documents, even though they are unlikely to be similar to my particular case.

The example is an ebook about gnuplot (http://gnuplot.info), the open-source plotting program. The book consists almost entirely of example gnuplot scripts and their output, side by side, with a paragraph or so of explanation for each example. On other words, it is similar to a recipe book. I enforce a firm rule: each script must work as presented, and the figure displayed with it must be the exact output of the script. This is more troublesome than it might sound at first. One winds up with hundreds of scripts and hundreds of image files. If I decide to alter one of the example scripts, I must ensure that the image printed next to it is the output of the new script and not a stale one. I use boldface highlighting and other presentational details in the printing of the example scripts to help the reader, but this involves markup that can't be fed to gnuplot.

The system that I describe here allows me simply to type the gnuplot scripts along with the text, adding special characters for boldface highlighting. A collection of filters and the output template take care of all the rest. They do the following: 1) replace my highlighting character with the LaTeX command for boldface; 2) take care of line continuations; 3) compute a checksum of the executable version of the script (with formatting characters removed); 4) define a hypertarget in the document using this checksum; 5) include the executable version of the script in the output as a PDF attachment; 6) process the script with gnuplot and save the resulting graph as a PNG, using the checksum in the name, if the image file does not already exist; 7) include the PNG in the output; and 8) create an index of plots, linking to their locations in the book using the hypertargets in step 4.

Using this system, I can make alterations to the examples at will, without having to worry about breaking anything or keeping track of what goes where. Any change in the script will lead to a changed checksum, so the program knows to run gnuplot on the changes and make a new figure. After using it to help process almost 200 pages of text, it's pretty well tested, and I feel that the time spent setting it up was well worth the resulting savings in headaches and tedium, allowing me to concentrate on the more enjoyable aspects of writing the book. In addition to the above, there are extra steps for those gnuplot examples that create animations. In that case, another filter creates a set of animation frames, and the system uses ImageMagick (http://www.imagemagick.org/script/index.php) to stitch them together into a movie. It creates a poster frame for the movie, attaches the movie to the PDF, copies the movie file to the publisher's server and creates a link to the movie file in the book.

Using the techniques for writing filters and templates described above, you already know how to construct a system like this. The only new idea here is using the Python filter to call out to external programs.

Here is the markup for one of the examples in the book. I wrap the gnuplot script in a code block with a `gnc` attribute:

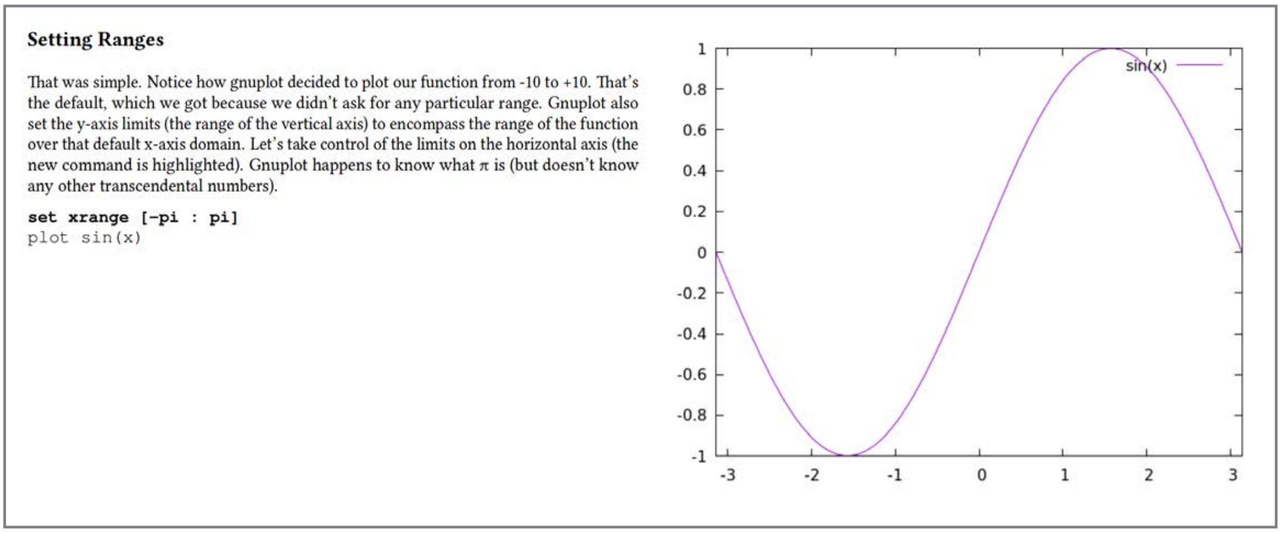
~~~
```gnc
@set xrange [-pi : pi]@
plot sin(x)
```
~~~

The @ characters are used to delimit what I want printed in boldface. The following shows the filter function that processes these code blocks:

```
#!/usr/bin/python3.5
import panflute as pf
import subprocess
c = subprocess.run
import re
import zlib
import os




def action(elem, doc):
    if isinstance(elem, pf.CodeBlock):
        if 'gnc' in elem.classes:
            # pff will hold the checksum
            pff = str(zlib.adler32(bytes(elem.text.replace('@', ''),
             ➥'utf-8')))
            #The name used for the
            #gnuplot output image:
            pfn = pff + '.png'
            #Check if we've done this one:
            if pfn not in os.listdir():
                dscript = elem.text.replace('@', '')
                script = '''set term pngcairo\nset out
                 ➥"{}"\n{}'''.format(pfn, dscript)
                with open(pff + '.gn', 'w') as scriptfile:
                    scriptfile.write(dscript)
```

```
                #Execute gnuplot on the script:
                c('''echo '{}' | gnuplot'''.format(script), shell = True)
            if doc.format == 'latex':
                #Lots of escaping needed:
                mt = elem.text.replace('{', '\\{')
                mt = mt.replace('}', '\\}')
                #LaTeX boldface:
                mt = re.sub('@(.*?)@', r'\\textbf{\1}', mt)
                mt = mt.replace('\\\n', '\\textbackslash\n')
                #for newlines in gnuplot labels, etc.

                mt = mt.replace('\\\\n', '\\textbackslash{n}')
                return pf.RawBlock(
                '\n\hypertarget{'+pff+'}{}\\begin{Verbatim}\n'+mt+
                '\n\end{Verbatim}\n\\textattachfile[mimetype=text/'
                +'plain]{' + pff + '.gn}{' + '\\framebox'+'
                {Open script}}\n\plt{' + pfn + '}',
                format = 'latex')
            else:
                return [elem, pf.RawBlock(
                    '<br /><img alt = "" src = "' + pfn + '" />',
                    format = 'html')]

if __name__ == '__main__':
    pf.toJSONFilter(action)
```

I've pointed out the key steps in code comments. The filter inserts code in the output, such as `\plt`, that refers to macros defined in the final template. These are simple affairs that handle some formatting, such as inserting page breaks and including the figures at the correct width. The code imports the zlib package for the checksums and uses the subprocess module to run external programs. It performs filesystem operations using the included os module. The filter inserts commands to use the LaTeX `attachfile` package to make PDF attachments, which I include in my output template. The `RawBlock`s are a panflute data structure for "raw text" and must include the intended output format as a second argument.



Setting Ranges

That was simple. Notice how gnuplot decided to plot our function from -10 to +10. That's the default, which we got because we didn't ask for any particular range. Gnuplot also set the y-axis limits (the range of the vertical axis) to encompass the range of the function over that default x-axis domain. Let's take control of the limits on the horizontal axis (the new command is highlighted). Gnuplot happens to know what π is (but doesn't know any other transcendental numbers).

```
set xrange [-pi : pi]
plot sin(x)
```

Figure 3. A Fragment from the Book

Figure 3 shows a page from the book where this early recipe appears.

And, finally, here is a fragment of the LaTeX source corresponding to the book fragment, where you can see how the code checksum is used:

```
\hypertarget{setting-ranges}{\subsection{Setting
Ranges}\label{setting-ranges}}

That was simple. Notice how gnuplot decided to plot
our function from -10 to +10. That's the default,
which we got because we didn't ask for any
particular range. Gnuplot also set the y-axis
limits (the range of the vertical axis) to
encompass the range of the function over that
default x-axis domain. Let's take control of the
limits on the horizontal axis (the new command is
highlighted). Gnuplot happens to know what π is
(but doesn't know any other transcendental
numbers).


\hypertarget{3260812063}{}\begin{Verbatim}
\textbf{set xrange [-pi : pi]}
```

```
plot sin(x)
\end{Verbatim}

\plt{3260812063.png}
```

You can use similar techniques to write filters and templates for anything imaginable: perhaps processing images, creating logs, using the results of calculations within documents, checking links or incorporating information scraped from websites. You can trigger all of these actions through markup that you define to extend Pandoc's Markdown language, using the panflute API.

I've found that Pandoc, used with Python-programmed filter functions, is a powerful system for authoring complex documents. I routinely translate single input files to HTML, PDF (through LaTeX) and docx for Word-using publishers, with no additional work. For those of us who write for a variety of publications, or simply want to be able to repurpose our manuscripts on occasion, a workflow based on Pandoc is the Holy Grail of authoring systems. ■

Dr Lee Phillips is a theoretical physicist and writer who lives in McLean, Virginia. He has worked on projects for the Navy, NASA and DOE on laser fusion, fluid flow, plasma physics and scientific computation. He writes about science and free software for scientists, and uses Linux exclusively in his work.

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

RETURN TO CONTENTS

Heirloom Software: the Past as Adventure

"Legacy" software can be more than a euphemism—sometimes it is art worth restoring. ERIC S. RAYMOND

PREVIOUS
Feature: Technical Writing with Pandoc and Panflute

Through the years, I've spent what might seem to some people an inordinate amount of time cleaning up and preserving ancient software. My Retrocomputing Museum page archives any number of computer languages and games that might seem utterly obsolete (http://www.catb.org/retro).

I preserve this material because I think there are very good reasons to care about it. Sometimes these old designs reveal unexpected artistry, surprising approaches that can help us break free of assumptions and limits we didn't know we were carrying.

But just as important, cultures understand themselves through their history and their artifacts, and this is no less true of programming cultures than of any other kind. If you're a computer hacker, great works of heirloom software are your heritage as surely as Old Master paintings are a visual artist's; knowing about them enriches you and

Therefore, I've always been more interested in forward-porting heirloom source code so it can be run and studied in modern environments.

helps solidify your relationship to your craft.

For exactly re-creating historical computing experiences, not much can beat running the original binary executables on a software emulator for their host hardware. There are small but flourishing groups of re-creationists who do that sort of thing for dozens of different historical computers.

But that's not what I'm here to write about today, because I don't find that kind of museumization very interesting. It doesn't typically yield deep insight into the old code, nor into the thinking of its designers. For that—to have the experience parallel to appreciating an Old Master painting fully—you need not just a running program but source code you can *read*.

Therefore, I've always been more interested in forward-porting heirloom source code so it can be run and studied in modern environments. I don't necessarily even consider it vital to retain the original language of implementation; the important goals, in my view, are 1) to preserve the original design in a way that makes it possible to study that design as a work of craft and art, and 2) to replicate as nearly as possible the UI of the original so casual explorers not interested in dipping into source code can at least get a feel for the experiences had by its original users.

Now I'll get specific and talk about *Colossal Cave Adventure*.

This game, still known as ADVENT to many of its fans because it was written on an operating system that supported only monocase filenames at most six characters long, is one of the great early classics of software. Written in 1976–77, it was the very first text adventure game. It's also the direct ancestor of every rogue-like dungeon simulation, and through those the indirect ancestor of a pretty large percentage of the games being written even today.

If you're of a certain age, the following opening sequence will bring back some fond memories:

```
Welcome to Adventure!!  Would you like instructions?

> n

You are standing at the end of a road before a small brick
building.
Around you is a forest.  A small stream flows out of the
building and
down a gully.

> in

You are inside a building, a well house for a large spring.

There are some keys on the ground here.

There is a shiny brass lamp nearby.

There is food here.

There is a bottle of water here.

>
```

From this beginning, the game develops with a wry, quirky, humorous and somewhat surrealistic style—a mode that strongly influenced the folk culture of computer hackers that would later evolve into today's Open Source movement.

For a work of art that was the first of its genre, ADVENT's style seems in retrospect startlingly mature. The authors weren't fumbling for an idiom that would later be greatly improved by later artists more sure of themselves; instead, they achieved a consistent (and, at the time, unique) style that would be closely emulated by pretty much everyone who followed them in text adventures, and not much improved on *as style* even though the technology of the game engines improved by leaps and bounds, and the range of subjects greatly widened.

ADVENT was artistically innovative—and with an architecture ahead of its time as well. Though the possibility had been glimpsed in research languages (notably LISP) as much as a decade earlier, ADVENT is one of the earliest programs still surviving to be organized as a complex, declaratively specified data structure walked by a much simpler state machine. This is a design style that is underutilized even today.

The continuing relevance of ADVENT's actual concrete source code, on the other hand, is quite a different matter. The implementation aged much more rapidly—and badly—than the architecture, the game or its prose.

ADVENT was originally written under TOPS-10, a long-defunct operating system for the DEC PDP-10 minicomputer. The source for the original version still exists (you can find it and other related resources at the Interactive Fiction Archive: http://www.ifarchive.org), but it tends to defeat attempts to appreciate it as a work of programming art because it's written in an archaic dialect of FORTRAN with (by actual count) more than 350 gotos in its 2.4KLOC of source code.

Preserving that original FORTRAN is, therefore, good for establishing provenance (as historians think about these things) but doesn't do a whole lot for that I've suggested as the cultural purposes of keeping these artifacts around. For that, a faithful translation into a more modern language would be far more useful.

As it happens, Don Woods' 1977 version of ADVENT was translated into C less than two years after it was written. You can still play it—and read the code—as part of the BSD Games package. Alas, while that translation is serviceable for building and running the program, it's not so great for reading. It is less impenetrable than the FORTRAN, but was not moved fully to idiomatic C and reads a bit strangely to a modern eye. (To be fair to the translators, the C language was still in its childhood in 1977, and its modern idioms weren't all that well developed yet.)

Thus, there are still a forbidding number of gotos in the BSD translation. Lots of information is passed around through shared globals in a way that was typical in FORTRAN but was questionable style in C even then. The BSD C code is full of mystery constants inherited from the ancestral FORTRAN source. And there is a serious comprehensibility

When you do a restoration like this, it's not enough merely to make a best effort to preserve original behavior. You ought to be able to *prove you have done so*.

problem around the custom text database that both the original FORTRAN and BSD C versions used—a problem I'll return to later in this article.

Through the late 1970s and early 1980s a lot of people wrote extensions of ADVENT, adding more rooms and treasures. The history of those variants is complicated and difficult to track. Almost lost in the hubbub was that the original authors—Will Crowther and Don Woods—continued to revise their game themselves. The last mainline version—the last release by Don Woods—was *Adventure 2.5* in 1995.

I found *Adventure 2.5* in the Interactive Fiction Archive in late 2016. Two things caught my attention about it. First, I had not previously known that Crowther and Woods *themselves* had shipped a version so extended from the famous original. Second—and unlike the early BSD port—there was nothing resembling what we'd expect in a modern source release to go with the bare code and the Makefile. No manual page. No licensing statement.

Furthermore, the 2.5 code was deeply ugly. It was C, but in worse shape than the BSD port. The comments actually included an apology from Don Woods explaining that it had been mechanically lifted from FORTRAN by a homebrew translator of his own devising—and apologizing for the bad style.

Nevertheless, I saw a possibility—and I wrote Don asking his permission to ship a cleaned-up version under a true open-source license. The reply was some time in coming, but Don not only granted permission speaking for both himself and Will Crowther, he also actively encouraged me to do this thing.

Now a reminder about what I think the goals of heritage preservation ought to be: I felt it was essential that the cleaned-up version should at no point break functional compatibility with what we got from Woods and Crowther. Therefore, the very first thing I did after getting the heirloom source to build clean was add the ability for it to capture command logs for regression testing.

When you do a restoration like this, it's not enough merely to make a best effort to preserve original behavior. You ought to be able to *prove you have done so*. Best practice, then is to start by building a really comprehensive set of regression tests. And that's what I did.

What *we* did, I should say. The project quickly attracted collaborators—most notably Jason Ninneman. The first of Jason's several good ideas was to use coverage-analysis tools to identify gaps in the test suite. Later, Petr Vorpaev, Peje Nilsson and Aaron Traas joined in. By about a month from starting, we could show more than 95% test coverage. And, of course, we ran retrospective testing with the newest version of the test suite on the earliest version we could make read the logs.

That kind of really good test coverage frees your hands. It allowed us to make rapid progress on the *other* prime goal, which was to turn the obfuscated source we started with into a readable work of art that fully revealed the design intentions and astonishing cleverness of the original.

So, all the cryptic magic numbers had to go. The goto-laden spaghetti code had to be restructured into something Don Woods in 2017 wouldn't feel he needed to apologize for. In general, what we aimed to transform the source code into was something we could believe Crowther and Woods—two of the most brilliant hackers of their time—would have written in 1977 if they had then had the tools and best practices of 2017 at their fingertips.

Our most (ahem) adventurous move was to scrap the custom text-database format that Crowther and Woods had used to describe the vocabulary of the game and the topology of *Colossal Cave*.

This—the "complex, declaratively-specified data structure" I mentioned earlier—was the single cleverest feature of the design, and it went all the way back to Crowther's very first version. The dungeon's topology is expressed by a kind of pseudo-code broadly resembling the microcode found underneath a lot of processor architectures; movement consists of dispatching to the sequence of opcodes corresponding to the current room and figuring out which one to fire depending not only on the motion verb the user entered but also on conditionals in the pseudo-code that can test for the presence or absence of objects and their state.

Good luck grokking that from the 2.5 code we started with though.

Here are the first two rules as they originally appeared in adventure.text, comprising ten opcodes:

```
3
1       2       2       44      29
1       3       3       12      19      43
1       4       5       13      14      46      30
1       145     6       45
1       8       63
2       1       12      43
2       5       44
2       164     45
2       157     46      6
2       580     30
```

Here's how those rules look, transformed to the YAML markup that our restoration, *Open Adventure* (http://www.catb.org/~esr/advent), now uses:

```
- LOC_START:
    travel: [
      {verbs: [ROAD, WEST, UPWAR], action: [goto, LOC_HILL]},
      {verbs: [ENTER, BUILD, INWAR, EAST], action:
       ➥[goto, LOC_BUILDING]},
      {verbs: [DOWNS, GULLY, STREA, SOUTH, DOWN], action:
       ➥[goto, LOC_VALLEY]},
      {verbs: [FORES, NORTH], action: [goto, LOC_FOREST1]},
      {verbs: [DEPRE], action: [goto, LOC_GRATE]},
    ]
- LOC_HILL:
    travel: [
      {verbs: [BUILD, EAST], action: [goto, LOC_START]},
      {verbs: [WEST], action: [goto, LOC_ROADEND]},
      {verbs: [NORTH], action: [goto, LOC_FOREST20]},
      {verbs: [SOUTH, FORES], action: [goto, LOC_FOREST13]},
      {verbs: [DOWN], action: [speak, WHICH_WAY]},
    ]
```

The concept of using a Python helper to compile a declarative markup like this to C source code to be linked to the rest of the game was maybe just barely thinkable when *Adventure 2.5* was written. YAML didn't exist at all until six years later.

But...designer's intent. That's much easier to see in the YAML version than in what it replaced. Therefore, given the *purpose* of heirloom restoration, YAML is better. Rather like stripping darkened varnish from a Rembrandt—the bright colors beneath may startle if you're used to the obscuring overlayer and think of it as definitive, but they are the truth of the work.

With our choices about what we could change so constrained, you might think the restoration was drudge work, but it wasn't like that at all. It was more like polishing a rough diamond—gradually seeing brilliance emerge from beneath an unprepossessing surface. The grottiness was largely—though not entirely—a consequence of the limitations of the tools Crowther and Woods had at hand. When we cleaned that up, we found genius with only a tiny sprinkling of bugs.

My dev team fixed those bugs, of course. We're hackers; that means we consider heirloom software a living heritage to be improved, not an idol to be worshiped. We certainly didn't think, for example, that Don Woods intended use of the verb "extinguish" on an oil-filled unlit urn to make the oil in it vanish.

Petr Vorpaev, reviewing a draft of this article, observed "Sometimes, we stripped bits of genius off, too. Because it was genius that was used to work around limitations that aren't there any more." He's thinking of a very odd feature of the 2.5 code—it worked around the absence of a string type in old FORTRAN by representing strings in a six-bit-per-character encoding packing five characters into a 32-bit word. That is, of course, a crazy thing to do in C, and we targeted it for removal early.

We added some minor features as well. For example, *Open Adventure* allows some command abbreviations that are standard in text-adventure games today but weren't supported in original ADVENT. By default, our version issues the > command prompt that also has been in common use for decades. And, you can edit your command input with Emacs keystrokes.

But, and this is crucial, all the new features are suppressed by an "oldstyle" option. If you choose that, you get a user experience that even a subject-matter expert would find difficult or impossible to distinguish from the 1995 and 1976–1977 originals.

Some of you might nevertheless be furrowing your brows at this point, wondering "YAML? Emacs keystrokes? Even as options? Yikes...can this really still be *Colossal Cave Adventure*?"

That's a question with a long pedigree. Philosophers speak of the "Ship of Theseus" thought experiment; if Theseus leaves Athens, and on his long voyage each plank and line and spar of the ship is gradually replaced, until not a fragment of the original wood remains when he returns to Athens, is it still the same ship?

The answer is, as any student of General Semantics could tell you, "What do you mean by 'same'?" Identity is not a well defined predicate; it changes according to what kind of predictive problem you are using language to tackle. Same arrangement of bits in the source? Same UI? Same behaviors at some level deeper than UI?

There really isn't one right answer. Those of you predisposed to answer "same" might argue "Hey, it passes the same regression tests." Only, maybe it doesn't now. Remember, we fixed some bugs. On the other hand...if the ship of Theseus is still "the same" after being entirely rebuilt, does it cease to be if we learn that the replacement for one of its parts doesn't replicate a hidden flaw in the original? Or if a few improvements have been added during the voyage that weren't in the original plans?

As a matter of fact, *Adventure* already has come through one entire language translation—FORTRAN to C—with its "identity" (in the way hackers and other people usually think of these things) intact. I think I could translate it to, say, Go tomorrow, and it would still be the same game, even if it's nowhere near the same arrangement of bits.

Furthermore, I can show you the ship's log. If you go to the project repository (https://gitlab.com/esr/open-adventure), you can view each and every small transformation of the code between *Adventure 2.5* and the *Open Adventure* tip version.

There is probably not a lot of work still to be done on this particular project, as long as our objectives are limited to be performing a high-quality restoration of *Colossal Cave Adventure*. As they almost certainly will be; if we wanted to do something substantially *new* in this kind of game, the smart way to do it would not be to code custom C, but to use a language dedicated to implementing them, such as Muddle (aka MDL) or Adventure Definition Language.

I hope some larger lessons are apparent. Although I do think *Colossal Cave Adventure* is interesting as an individual case in itself, I really wrote this article to suggest constructive ways to think about the general issues around restoring heirloom software—why you might want to do it, what challenges and rewards you'll find, and what the best practices are.

Here are the best practices I can identify:

- The goals to hold in mind are 1) making the design intent of the original code available for study, and 2) preserving the oldstyle-mode UI well enough to fool an original user.

- Build your regression-test suite first. You want to be able to *demonstrate* that your restoration is faithful, not just assert it.

- Use coverage tools to verify that your regression tests are good enough to constitute a demonstration.

- Once you have your tests, don't sweat changing tools, languages, implementation tactics or documentation formats. Those are ephemera; good design is what endures.

- Always have an oldstyle option. Gain the freedom to improve by making, and keeping, the promise of fidelity to the original behavior in oldstyle mode.

- *Do* fix bugs. This may conflict with the objective of perfect regression testing, but you're an engineer, not an embalmer. Work around that conflict as you need to.

- Show your work. Your product is not just the restored software but the repository from which it ships. The history in that repository needs to be a continuing demonstration of good judgment and sensitivity to the original design intent of the code.

- Document what you change, including the bug fixes. It is good practice to include maintainer's notes describing your restoration process in detail.
- When in doubt about whether to add a feature, be neither over-eager to put your mark on the code nor a slave to its past. Instead, ask "What's in good taste?"

And while you're doing all this, don't forget to *have fun*. The greatest heirloom works, like *Colossal Cave Adventure*, were more often than not written in a spirit of high-level playfulness. You'll be truer to their intent if you approach restoring them with the same spirit.■

Eric S. Raymond is a wandering anthropologist and trouble-making philosopher. He's been known to write a few lines of code too. Actually, if the tag "ESR" means nothing to you, what are you doing reading this magazine?

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 #
All Things Open	http://www.AllThingsOpen.org	31
Drupal Jobs	https://jobs.drupal.org/	67
Drupalize.me	http://drupalize.me	69
HPC Wallstreet	http://www.flaggmgmt.com/hpc	13
InterDrone	http://www.InterDrone.com	17
Peer 1 Hosting	http://go.peer1.com/linux	119
Silicon Mechanics	http://www.siliconmechanics.com	89
SPTech Con	http://www.sptechcon.com	39
SUSE	http://suse.com/storage	7
Vision	http://precisionagvision.com	15
WiSTEM	http://www.womeninstemconference.com	27

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

peer1 hosting

Where every interaction matters.

break down
your innovation barriers

power your business to its full potential

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

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

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

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

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