LINUX JOURNAL

Since 1994: The Original Magazine of the Linux Community

Develop Feature-Rich Apps with Command-Line Tools

OCTOBER 2017 | ISSUE 282
http://www.linuxjournal.com

PROGRAMMING

Scripting Project with Bash and PHP

+

How to Use Threading in Python

Introducing Subutai, a New Kind of Cloud

Ubuntu and Bash as a Windows Program

WATCH: ISSUE OVERVIEW

CONTENTS

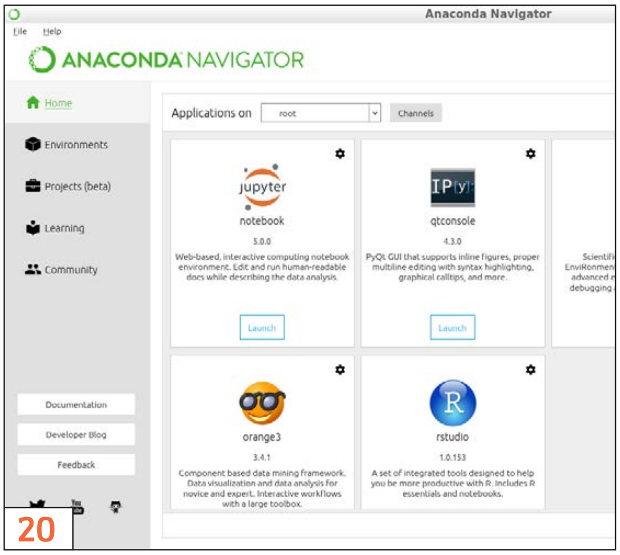
OCTOBER 2017
ISSUE 282

FEATURES

Cover Image: © Can Stock Photo / undrey

COLUMNS

IN EVERY ISSUE

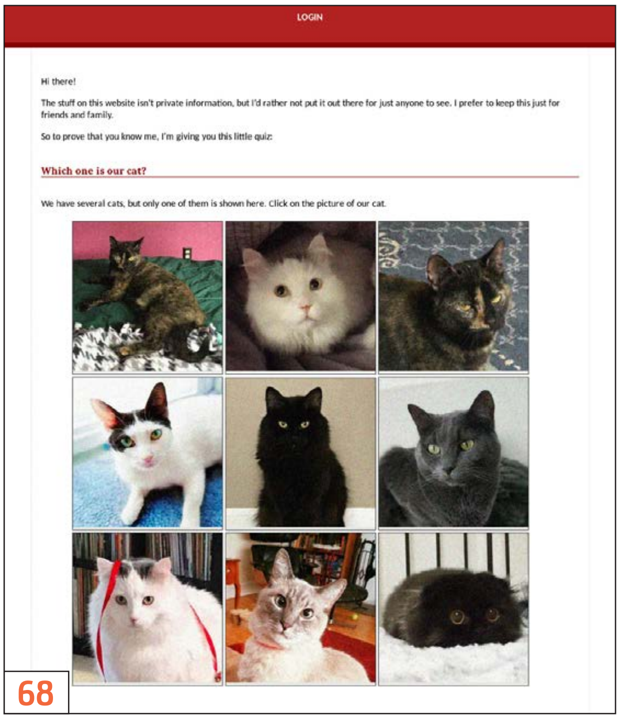


Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.

A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.



Hi there!

The stuff on this website isn't private information, but I'd rather not put it out there for just anyone to see. I prefer to keep this just for friends and family.

So to prove that you know me, I'm giving you this little quiz:

Which one is our cat?

We have several cats, but only one of them is shown here. Click on the picture of our cat.

ON THE COVER
- Develop Feature-Rich Apps with Command-Line Tools, p. 86
- Scripting Project with Bash and PHP, p. 68
- How to Use Threading in Python, p. 30
- Introducing Subutai, a New Kind of Cloud, p. 98
- Ubuntu and Bash as a Windows Program, p. 38

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

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

ENJOY:

Timely delivery

Off-line reading

Easy navigation

Phrase search and highlighting

Ability to save, clip and share articles

Embedded videos

Android & iOS apps, desktop and e-Reader versions

SUBSCRIBE TODAY!

LINUX JOURNAL

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

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

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

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

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

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

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

LINUX is a registered trademark of Linus Torvalds.

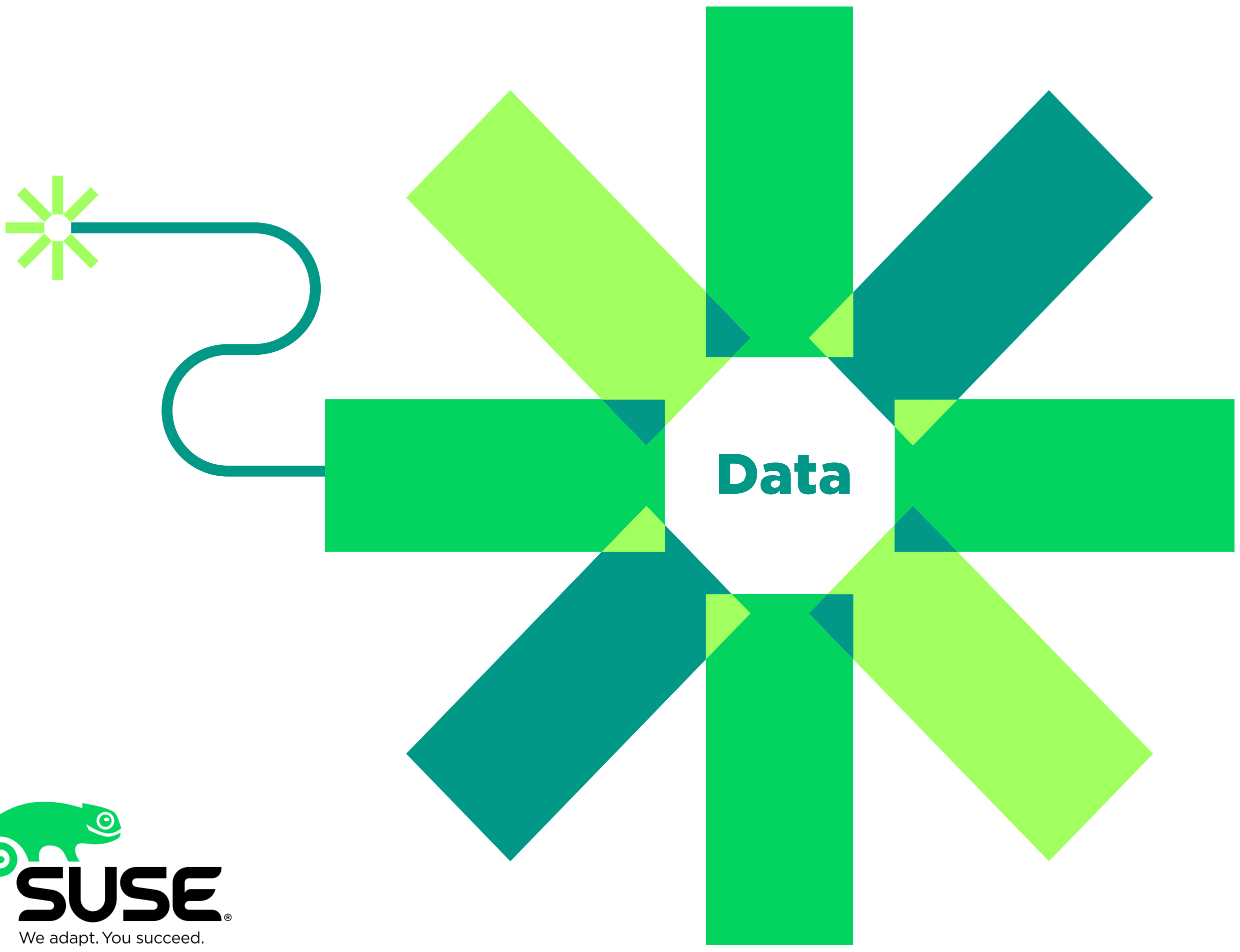
STORAGE
R∃DEFINED:

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.

Bash and Cats

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.

VIDEO: Shawn Powers runs through the latest issue.

If someone asked me how the internet stays running, I'd probably say something like, "Bash scripts and cat photos." Because really, those two things pretty much encompass the human online experience. Bash scripts are quick snippets of timing-saving code, and cat photos are, well, photos of fluffy kitties. Most days, that's enough. Although I won't say this issue is based on that premise, I'd like to pretend that's the case!

Reuven M. Lerner starts things off with how to create multithreaded applications in Python. (See? Thread/yarn? The whole world is cats!) Usually the tasks we need computers to perform are so simple, the computer can do them one at a time faster than we can blink. But, what if those tasks are time-consuming? Multithreading is a great way to force computers to do more work at once.

Dave Taylor follows with an article on Bash scripting inside Windows. Yes, even in Windows Bash scripts rule. I think my theory is proving itself.

Kyle Rankin talks about taking a vacation again this month. Specifically, he talks about purchasing a cheap laptop that can be taken on vacation or used when on call without concern about it getting stolen or damaged. Sure, most tech folks spend a few minutes on vacation solving a work issue, but why tote your fancy everyday computer along with you, when a flimsy keyboard and outdated CPU will work perfectly fine for five minutes of work?

In fact, if you read my article this month on

Ansible playbooks, you might need less than five minutes to solve problems back home. Ansible automation really starts to shine when playbooks are added to the mix.

Jim Hall takes my cat theory to a very literal degree. This month, he shows how to create your own CAPTCHA system, but rather than proving human-ness, your CAPTCHA can test personal information. In Jim's case, his CAPTCHA displays a bunch of cat photos, and you need to pick which cat is his. For friends, it's a simple test. For strangers? Not so much. Might it be tempting to answer incorrectly in order to see more cats? Yes. But, that's one of the dangers of using cats. Their fluffiness is hard to resist.

Andy Carlson finishes off the issue with an indepth look at Bash itself. We all know a handful of tricks Bash can do easily, but what about things like encryption? Our ever-vigilant scripting language is so versatile, even seasoned professionals like myself are likely to find useful tips and tricks we never realized were possible.

Speaking of tips and tricks, this issue of *Linux Journal* might focus on fluff and scripting, but it's also full of the same tech tips and useful programs you've come to expect every month. We also have new product announcements, updates about the tech world, and insight on technology in general. This was a fun issue of *Linux Journal*, and it contains far more cat photos than most. Although, that might be a shortcoming of the other issues, as a certain number of cat photos should be considered baseline, no?■

RETURN TO CONTENTS

PREVIOUS
Current_Issue.tar.gz

NEXT
Editors' Choice

diff -u

What's New in Kernel Development

The **OOM** (out-of-memory) killer targets processes that seem to be responsible for zotzing a system due to taking too much RAM. It relies on various policies to make a best guess about which process to kill, and then it kills it, on the hope that the system will then be usable again. Sometimes it guesses right, and sometimes it doesn't. OOM policies always are under ongoing development and very difficult to get right.

Roman Gushchin recently wanted to address the fact that the OOM killer didn't recognize virtual systems as such. Instead of killing the whole container, it would kill only a single process inside the container, which might just render the entire container unusable. Roman posted some patches to allow the entire cluster of processes in a container to be considered one single "job", which the OOM killer could kill or not, according to its policies. This way, containers wouldn't have to worry about possibly finding one of their processes mysteriously missing.

Michal Hocko and others didn't like this, because on an abstract level, it did represent a change to OOM killer policies. Instead of process A being killed, process B would be killed instead. If Roman was going to change OOM killer policy, Michal said, there needed to be a whole consideration of how these policies would affect various workloads.

Roman, joined by **Johannes Weiner**, argued that the patches were not related to policy, but simply redefined what was meant by a "job", so that a whole container could be seen as a discrete entity and targeted appropriately. Johannes said, "all we want is the OOM policy, whatever it is, applied to cgroups."

Ultimately, Roman may have some hurdles to jump over before any patch could be accepted. It's crucial that the OOM killer remain as useful as possible for the widest possible variety of use cases, so any change must be considered carefully. But it's also clear that **cgroups** will need to be handled properly by the OOM killer as well, or else we could start to see weird breakages in virtual systems, without a good way to address the problem.

Intel's **Ross Zwisler** recently said that it soon would be very common for a given device to have a variety of different types of memory installed, with different speeds and other characteristics. The kernel, and especially user code, would need to have some way of controlling, or at least suggesting, which type of memory they preferred to run on.

He said that the Intel folks were leaning toward a **SysFS**-based solution, in which various SysFS files would indicate ranges of memory, and the speed and other characteristics associated with those ranges. He wanted some guidance from the kernel people about how best to expose that information, and whether SysFS was the right approach, and so on.

No one had any problem with Ross' approach, at least not at that stage—although it wouldn't be utterly uncommon for Intel to put in a lot more work, only to have **Linus Torvalds** or some other

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.

high-level developer raise objections later.

In general, the thread had the feel of Intel simply alerting folks to a new hardware direction and making sure its future patches in the area would at least not come as a surprise to anyone. Mission accomplished.

There's a new security framework in town, and its name is **SARA**. **Salvatore Mesoraca** has begun implementing it as a general-purpose framework that can be used to put together a variety of security policies, according to the needs of any given system. One of its main claims to fame is that it's "stackable", meaning it can interoperate cleanly with other security systems.

Stackability is useful because there are a bunch of people working in this area, and it would be nightmarish if they all had to coordinate their efforts before their patches could be deemed acceptable for inclusion in the main kernel tree. Although, of course, if any of them could be merged together, that would be seen as a benefit rather than a drawback.

That issue came up when Salvatore announced one of his SARA sub-modules as being a **WX** protection system—ensuring that a piece of memory could be either executable or writable, but not both. **Mickaël Salaün** also had implemented a similar security feature and suggested merging the two. Ultimately, the two systems seemed too different to merge, but their stackability meant that Mickaël could make use of SARA features to shore up the gaps in his own work.—Zack Brown drupalize.me

Instant Access to Premium Online Drupal Training

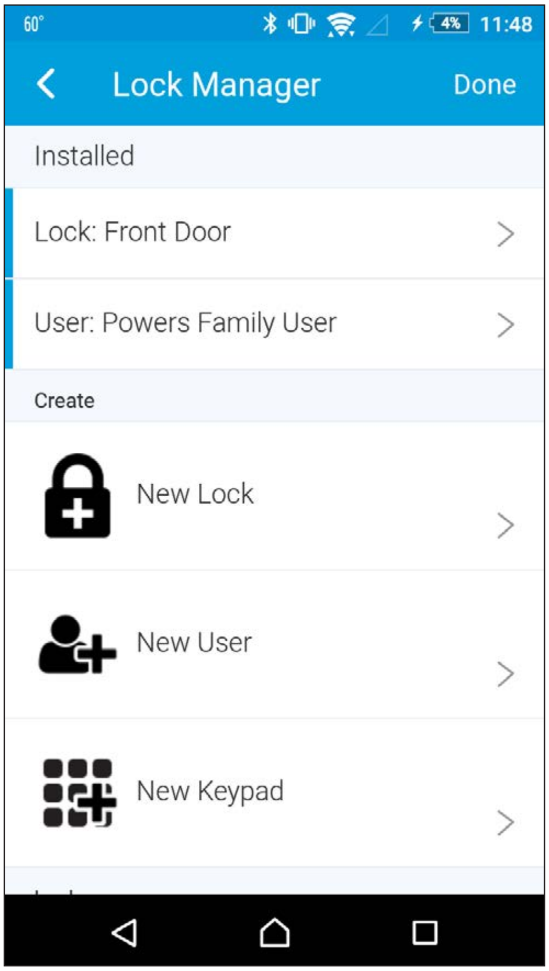
- ✓ *Instant access to hundreds of hours of Drupal training with new videos added every week!*
- ✓ *Learn from industry experts with real world experience building high profile sites*
- ✓ *Learn on the go wherever you are with apps for iOS, Android & Roku*
- ✓ *We also offer group accounts. Give your 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 = c
        sel.tex
        this.fo
      }
      else if (
        var sta
        var end
        var scr
        this.va
        this.fo
        // High
```

Rekey Your House—from Hawaii



I've mentioned the SmartThings home automation system I use in my house before, but I specifically wanted to highlight a free app that integrates with the Android SmartThings app to provide far more advanced features than most lock manufacturers include with their own systems.

Although it does require you to use the SmartThings system, all the apps and configurations are stored in the cloud, so you can control the system from any Android device. You can configure multiple users with multiple codes, allow certain codes to work during certain times, or even make "burner" codes that will work only once then disable themselves.

The Lock Manager application is incredibly powerful and fairly easy to install if you follow the directions at the website: https://github.com/ethayer/lock-manager. We recently had a situation at our house that required us to change the locks from afar, and I'm happy to say, Lock Manager did the trick! The only "gotcha" we found is that once you make a change, it's important to lock and unlock the door remotely manually (using SmartThings app) to make sure the new codes are uploaded properly. If you have a lock like we do (the Kwikset 910 Z-Wave deadbolt), be sure to get Lock Manager; it will change the way you think about digital key codes!—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.

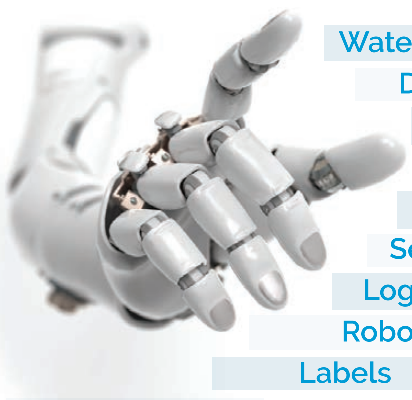
WHY ATTEND?

- **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.

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

140+ COMPANIES
35 LEADING SPEAKERS
12 COUNTRIES
185+ ATTENDEES

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

#PRECISIONAGVISION
PRECISIONAGVISION.COM

Two Ears, Three Headsets

During my normal workday, I'm connected to various computers and phones. I used to have wired USB headsets for each computer, and I'd have to swap them on and off every few minutes. Recently, I've started using wireless headsets, because the audio quality is finally to the point where I can use them for meetings and even some recording without worrying I'll sound like someone talking in the subway on a Bluetooth headset.

To get to this point, I've purchased a shameful number of headsets. Two of them I'm happy to say are well worth their significant investment.

The Logitech H820e is a single earpiece (my preference, as you can see in the photo I wear multiples at once!) headset with a flexible boom for the microphone. The sound quality on this is great for listening and for speaking, and although the big fluffy earpiece does get a bit hot and sweaty, it's tolerable for a few hours at a time. Rather than Bluetooth, this headset uses DECT technology, which makes for great connectivity, even during trips to the bathroom. (I know this from experience.) My office tends to be quiet, so the noise-cancellation features aren't really something I take advantage of, but this Logitech unit indeed works in noisy environments too. I've used this headset all day and never had an issue with the charge running out. It's important to put it on the charger base every night, however, because I've come to work in the morning and found it dead on my desk when I forgot.

The other headset I regularly use is the Jabra Pro 935 for computers. This headset is a bit less expensive and a bit more comfortable (less ear sweat) than the Logitech. I find the audio quality to be a little less during recordings, however, so it's not one I normally use for business meetings or video recording. Still, it's the headset I tend to use more often for listening to things, because it's very comfortable, and I can wear it for hours without even noticing it's there. The Jabra uses a "2G4" wireless technology that I'm not really familiar with. It has close to the same range as the Logitech with DECT technology, however, and both headsets blow away even the best Bluetooth headsets when it comes to range and quality.

If you're like me and like to be connected without the need for wires draping all over, I'm happy to report the current line of wireless headsets is well worth the investment. Most use standard USB cables for connection to the computer, and wireless connectivity is handled simply by plugging the headset in to the charger/transmitter base. I don't necessarily recommend wearing three headsets at a time like I do (I always wear my Trekz Titanium bone conduction headset as well), but if you don't mind the Borg-like appearance, it's more than possible!—Shawn Powers

Networking: a Horse of a Different Color

When it comes to networking devices in your house or office, nothing beats a solid Ethernet cable. During the past few years, Wi-Fi has gotten to the point that it can be a truly reliable second-place option for connecting devices, but is it really the second-best way?

My family is buying a farm (literally, not figuratively), and the barn is

200+ feet away from the house. I could get a wireless bridge pair to get Wi-Fi out there, but since the barn already is wired for power, I thought maybe a Powerline solution would make more sense. They're cheaper, and if reliable, far simpler to deploy. So I bought a TP-Link AV2000 pair of adapters, and I have to admit, I'm sold.

I won't talk about setting them up, because quite honestly, it's so simple it seems *too* simple. Once they're set up, it's like plugging a cable in to a network switch. Quite frankly, they just work! I haven't tested the claimed 2000Mbps bandwidth, because that's far more than I need anyway. As far as reliable connections go, however, the Powerline option is actually well worth checking out. It's possible your situation will prove to be a bad candidate, because things like vacuum cleaners, air conditioners or any other motorized device can introduce noise on the line that will disrupt or slow down the signal. My advice is to try it, but keep the package in good shape in case you need to return it.—Shawn Powers

---

Using Python for Science

In past articles, I've looked at several ways you could use Python to do scientific calculations, but I've never actually covered how to set up and use Python itself in a way that makes scientific work easier. So in this article, I explore Anaconda, a Python distribution that is targeted at scientific and data research.

The default installation includes a large number of Python modules that are useful when doing data science—or really any other type of scientific computing. Installation is relatively easy. You can find download links on the main Anaconda site (https://www.continuum.io) that will allow you to choose between Mac OS X, Windows and Linux.

For Linux, you can choose between Python 2.X and 3.X, as well as between 32-bit or 64-bit executables. Now that Python 3.X has matured more, my default suggestion has changed. Unless you have a specific reason to do something differently, I suggest that you default on downloading and using Python 3.X. Once it is downloaded, you can make the downloaded file executable or you can run it directly by using bash, like this:

```
bash ./Anaconda3-4.4.0-Linux-x86_64.sh
```

You'll need to accept the license agreement to finish the installation.

The installer will ask for an installation location, defaulting on the anaconda3 directory within your home directory. It will unpack everything there and then ask if you want its bin directory added to your PATH environment variable. It's important to remember this if you use Python scripts to do system administration tasks. If you just run the command `python`, it will default to the one installed by Anaconda.

One of the core technologies that makes Anaconda unique is the conda package management system. Conda can be used to manage all of the modules and other software installed when you installed

Anaconda. To manage updates, simply run the following commands:

```
conda update --all
```

You also can update individual packages selectively by using their package names in the above command rather than the `--all` option. To install a new Python module, such as opencv, use the command:

```
conda install opencv
```

That command will check on all the requirements and make sure all the dependencies are correct.

If you can't remember (or don't know) what a particular module name might be in the conda packaging scheme, you can do a search with a command like the following:

```
conda search --names-only open
```

This will return a list of all of the conda package names that have the text "open" in them.

You always can check to see what already has been installed by using the `list` option to conda.

If you have finished with some experimental code and want to remove a particular package that you no longer need, you can uninstall them with the following command:

```
conda remove opencv
```

All of those commands have several more options that I haven't covered here, but you can find many more details by looking at their help pages.

Another really powerful tool, especially when working on multiple projects, is the enhanced management of virtual environments that is possible with Anaconda. When you are doing research computations, you often have to start with explorations into your problem area. You definitely don't want any of those exploratory tasks to interfere with any currently ongoing work. So the best option is to set up a separate, isolated, environment where it is safe to destroy things with no fear of losing earlier work. This is handled by virtual environments. Python has had virtual environments for some time, but managing them can be unintuitive for some people. Anaconda has included a set of tools to help simplify the process.

When you install Anaconda, you actually are operating within a default environment already. In order to create a new one, you would use the command:

```
conda create --name project1
```

In order to activate this new environment, run the command:

```
source activate project1
```

Now, everything you do, with regard to Python and conda, will take place within this environment. For example, if you run the command `conda list` within this environment, you'll see that there are no packages installed. If you now install a package, it will exist only within this environment. This way, you can have an isolated environment that will contain only the Python modules you need for that particular project.

If you already have an environment that you have been working with, but you want to extend it in some manner, you can clone this starting environment with the command:

```
conda create --name project2 --clone project1
```

As you work with this environment, conda keeps track of the history of changes that you have applied to it. You can get the list of those changes with:

```
conda list --revisions
```

This way, you always can revert back to some previous revision with the following command, where X is the revision number you want to revert to):

```
conda install --revision X
```

Once you are done with your work for the day, you can deactivate a given environment with:

```
source deactivate
```

When you are completely finished with a particular environment, you can permanently delete it with:

```
conda remove --name project2 --all
```

Just be sure that you are deleting the correct environment. You don't want to destroy all of your hard work accidentally.

You can get a list of all of the environments managed by conda with the command:

```
conda info --envs
```

If you are working on a project collaboratively, you probably don't want to have to send an entire environment to someone else, as that simply would take too much bandwidth. You also don't want to send a list of handwritten instructions on how to re-create it, as humans are famous for forgetting steps. Instead, conda includes the following command that you can use to create a descriptive file:

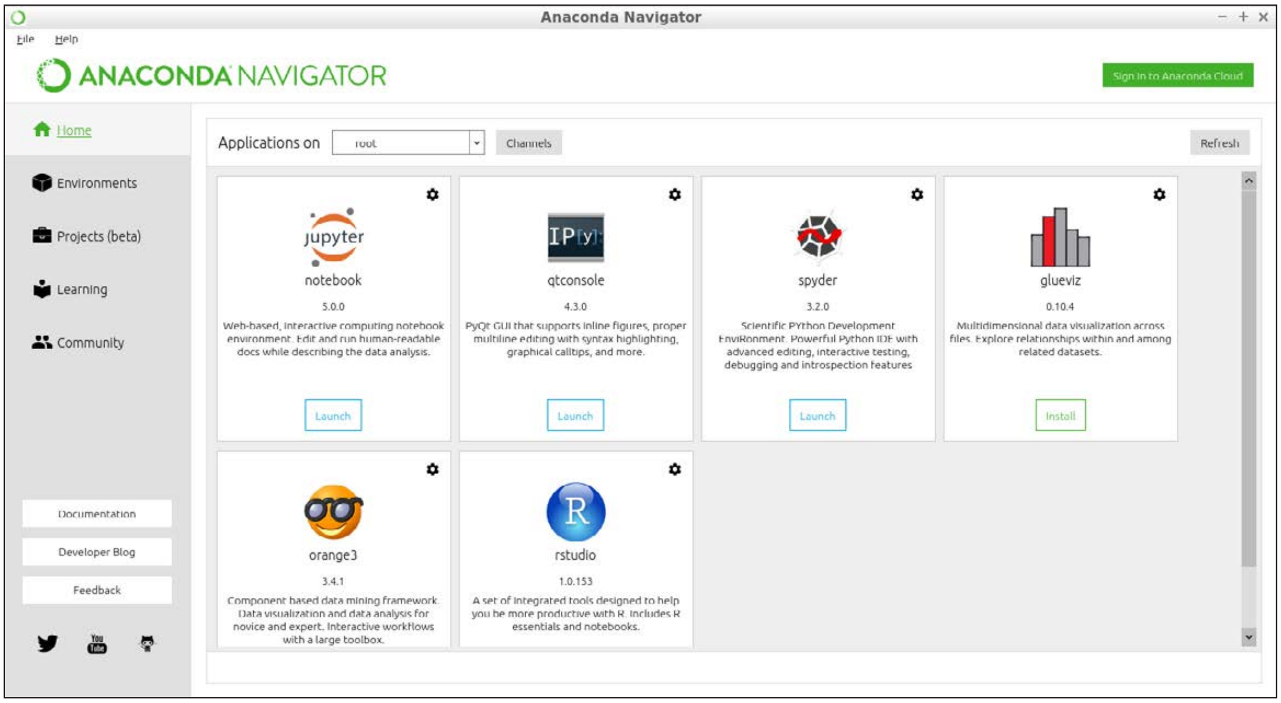
```
conda list --explicit >> project1.txt
```

You can send this file to your collaborators and have them run this:

```
conda create --name my_project1 --file ./project1.txt
```

That will allow them to re-create your project environment.

All of these commands have been managed on the command line,



Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

Multidimensional data visualization across files. Explore relationships within and among related datasets.

Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.

A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.

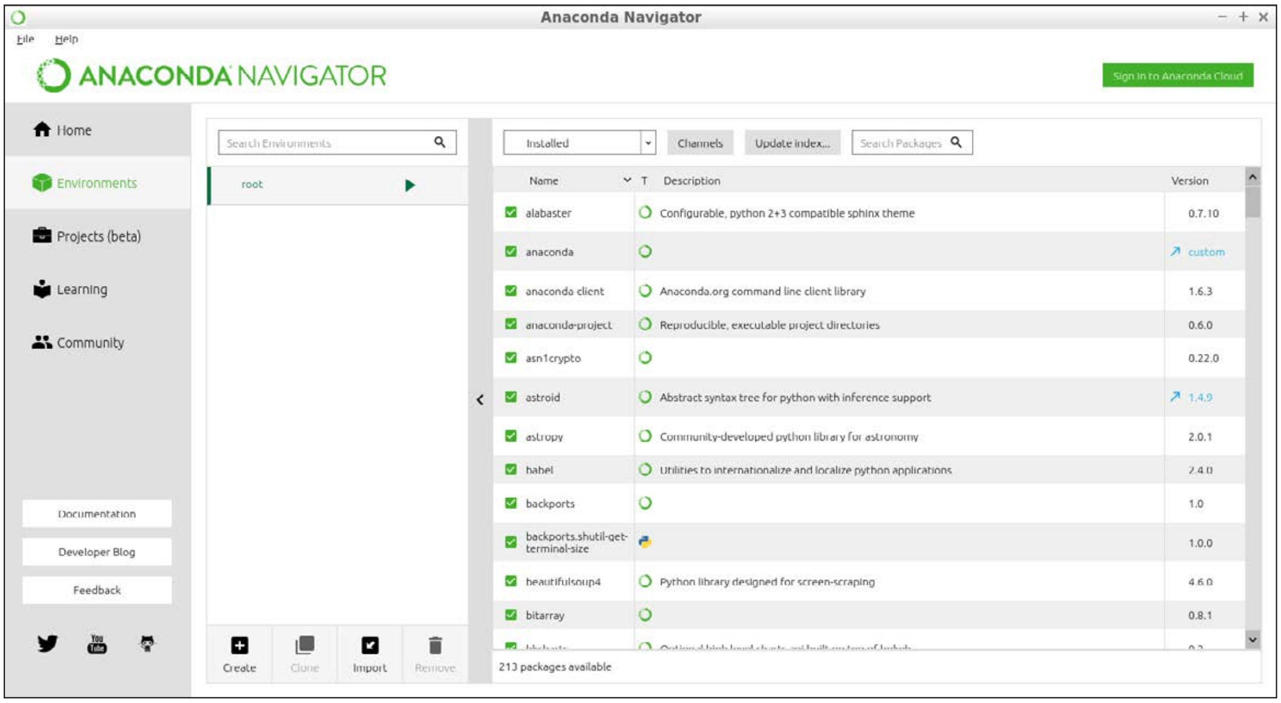
Figure 1. The Anaconda Navigator provides a graphical interface for interacting with your installation of Anaconda.

but not everyone is comfortable working that way. For those people, Anaconda includes the Anaconda Navigator. You can start it with the command `anaconda-navigator`.

On the first page of the application, you'll see launchers for the major Python software that is available through Anaconda. This includes packages like spyder, orange3 and the jupyter notebook. If they haven't been installed yet, you'll see a button labelled "Install" rather than "Launch".

The second page allows you to manage environments within your Anaconda installation. From here, you can manage the installed Python modules, create new environments or clone existing ones. You even can import projects from specification files to create a new copy of an environment. The right-hand side of the window displays Python modules, and you filter based on whether the list is those installed, update-able or yet to be installed.

There is a third page, currently in beta, which manages projects. Projects are a way of organizing larger pieces of code and sharing them



Name	T	Description	Version
✓ alabaster		Configurable, python 2+3 compatible sphinx theme	0.7.10
✓ anaconda			↗ custom
✓ anaconda client		Anaconda.org command line client library	1.6.3
✓ anaconda-project		Reproducible, executable project directories	0.6.0
✓ asn1crypto			0.22.0
✓ astroid		Abstract syntax tree for python with inference support	↗ 1.4.9
✓ astropy		Community-developed python library for astronomy	2.0.1
✓ babel		Utilities to internationalize and localize python applications	2.4.0
✓ backports			1.0
✓ backports.shutil-get-terminal-size			1.0.0
✓ beautifulsoup4		Python library designed for screen-scraping	4.6.0
✓ bitarray			0.8.1

213 packages available

Figure 2. Anaconda allows for managing environments within your installation.

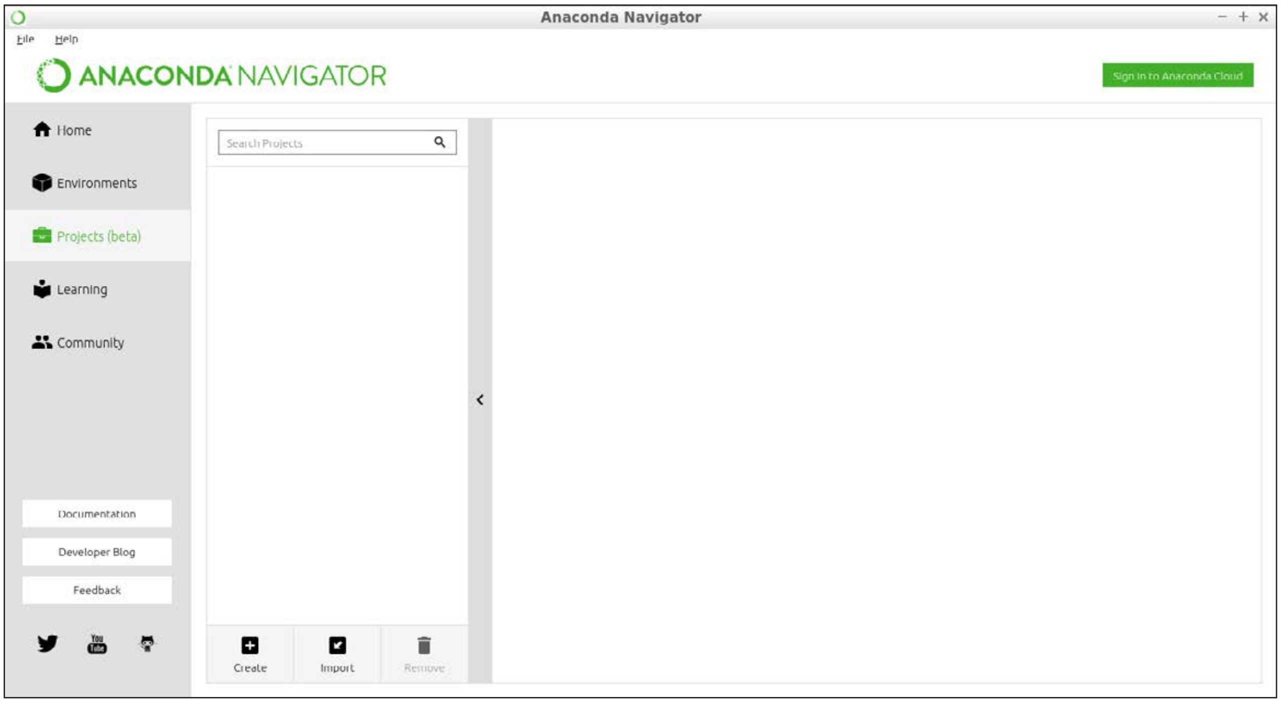


Figure 3. Anaconda also helps you manage larger projects, along with environments.

with others. Sharing is made easier with the Anaconda Cloud. Once you have an account on Anaconda Cloud, you can upload projects, environments, packages and jupyter notebooks. Once they have been uploaded, you can share them with other people around the globe much more easily. Although you can log in and work with the Anaconda Cloud in a web browser, the Anaconda Navigator allows you to log in directly from there and be able to interact with your materials stored online.

This was a short introduction, but hopefully I covered enough to help you better organize your scientific code. In future articles, I plan to dig a bit more into actually doing some scientific work with Python and taking advantage of these organizational tools.—Joey Bernard

THEY SAID IT

Never continue in a job you don't enjoy. If you're happy in what you're doing, you'll like yourself, you'll have inner peace. And if you have that, along with physical health, you will have had more success than you could possibly have imagined.
—Johnny Carson

It is curious that physical courage should be so common in the world and moral courage so rare.
—Mark Twain

The secret of happiness is to make others believe they are the cause of it.
—Al Batt

Farming looks mighty easy when your plow is a pencil, and you're a thousand miles from the corn field.
—Dwight D. Eisenhower

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

PREVIOUS
UpFront

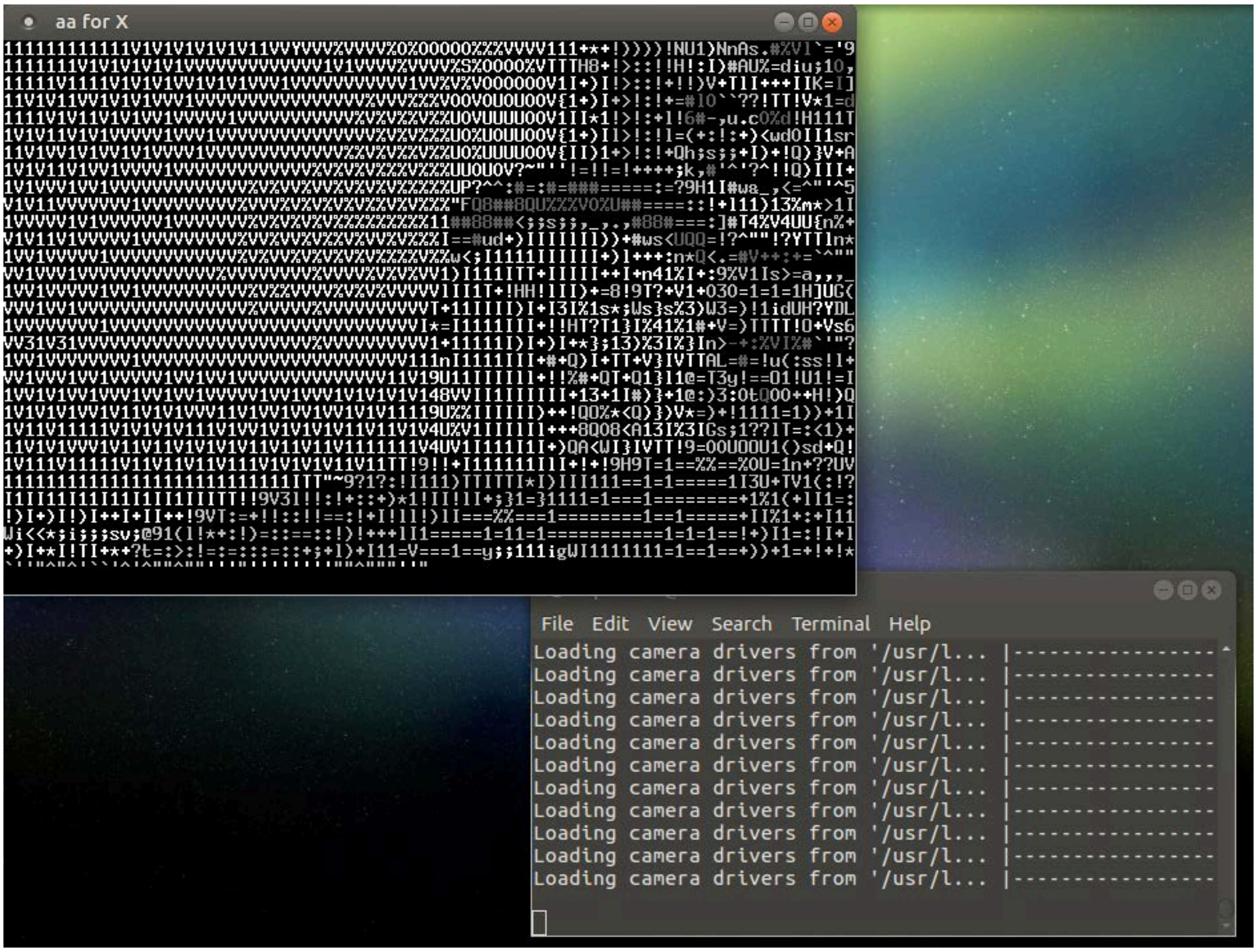
NEXT
Reuven M. Lerner's
At the Forge

When a Webcam Just Doesn't Cut It

With my obsession—er, I mean hobby—regarding BirdCam, I've explored a great number of camera options. Whether that means trying to get Raspberry Pi cameras to focus for a macro shot of a feeder or adjusting depth of field to blur out the neighbor's shed, I've fiddled with just about every webcam setting there is. Unfortunately, when it comes to lens options, nothing beats a DSLR for quality. Thankfully, there's an app for that.

The gphoto2 suite of drivers and apps allows a huge list of DSLR cameras to function as image or video capture devices inside Linux. There's a compatibility list at http://gphoto.org/proj/libgphoto2/support.php, and even if your camera isn't listed, it's likely you'll be able to use it in some manner. For example, my Nikon Coolpix P610 isn't officially supported, but I'm able to get some images from it regardless. In fact, it even supports previewing a photo in ASCII art. That might not be a useful feature, but I found it incredibly fun to play with.

In all reality, if you want to automate taking pictures while using a real DSLR camera, gphoto2 is probably the tool you want. Whether it's a short-term setup on a tripod or a future super-high-def BirdCam (send me a link!), be sure to check it out. In fact, gphoto2 is so cool



```
Loading camera drivers from '/usr/l... |-----------------
Loading camera drivers from '/usr/l... |-----------------
Loading camera drivers from '/usr/l... |-----------------
Loading camera drivers from '/usr/l... |-----------------
Loading camera drivers from '/usr/l... |-----------------
Loading camera drivers from '/usr/l... |-----------------
Loading camera drivers from '/usr/l... |-----------------
Loading camera drivers from '/usr/l... |-----------------
Loading camera drivers from '/usr/l... |-----------------
Loading camera drivers from '/usr/l... |-----------------
Loading camera drivers from '/usr/l... |-----------------
```

and powerful, I'm giving it this month's Editors' Choice award. And if I can get my wife to loan me her Canon DSLR with the 50mm prime lens, BirdCam might rise to a whole new level!

—Shawn Powers

RETURN TO CONTENTS

Threading in Python

Threads can provide concurrency, even if they're not truly parallel.

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.

IN MY LAST ARTICLE, I took a short tour through the ways you can add concurrency to your programs. In this article, I focus on one of those forms that has a reputation for being particularly frustrating for many developers: threading. I explore the ways you can use threads in Python and the limitations the language puts upon you when doing so.

The basic idea behind threading is a simple one: just as the computer can run more than one process at a time, so too can your process run more than one thread at a time. When you want your program to do something in the background, you can launch a new thread. The main thread continues to run in the foreground, allowing the program to do two (or more) things at once.

What's the difference between launching a new process and a new thread? A new process is completely independent of your existing process, giving you more stability (in that the processes cannot affect or corrupt one another) but also less flexibility (in that data cannot easily flow from one thread to another). Because multiple threads within a process share data, they can work with one another more closely and easily.

For example, let's say you want to retrieve all of the data from a variety of websites. My preferred Python package for retrieving data

Listing 1. retrieve1.py

```
#!/usr/bin/env python3

import requests
import time

urls = [one_line.strip()
        for one_line in open('urls.txt')]

length = {}

start_time = time.time()

for one_url in urls:
    response = requests.get(one_url)
    length[one_url] = len(response.content)

for key, value in length.items():
    print("{0:30}: {1:8,}".format(key, value))


end_time = time.time()

total_time = end_time - start_time

print("\nTotal time: {0:.3} seconds".format(total_time))
```

from the web is the "requests" package, available from PyPI. Thus, I can use a `for` loop, as follows:

```
length = {}

for one_url in urls:
    response = requests.get(one_url)
    length[one_url] = len(response.content)

for key, value in length.items():
    print("{0:30}: {1:8,}".format(key, value))
```

How does this program work? It goes through a list of URLs (as strings), one by one, calculating the length of the content and then storing that content inside a dictionary called `length`. The keys in `length` are URLs, and the values are the lengths of the requested URL content.

So far, so good; I've turned this into a complete program (retrieve1.py), which is shown in Listing 1. I put nine URLs into a text file called urls.txt (Listing 2), and then timed how long retrieving each of them took. On my computer, the total time was about 15 seconds, although there was clearly some variation in the timing.

Listing 2. urls.txt

```
http://lerner.co.il
http://LinuxJournal.com
http://en.wikipedia.org
http://news.ycombinator.com
http://NYTimes.com
http://Facebook.com
http://WashingtonPost.com
http://Haaretz.co.il
http://thetech.com
```

Improving the Timing with Threads

How can I improve the timing? Well, Python provides threading. Many people think of Python's threads as fatally flawed, because only one thread actually can execute at a time, thanks to the GIL (global interpreter lock). This is true if you're running a program that is performing serious calculations, and in which you really want the system to be using multiple CPUs in parallel.

However, I have a different sort of use case here. I'm interested in retrieving data from different websites. Python knows that I/O can take a long time, and so whenever a Python thread engages in I/O (that is, the screen, disk or network), it gives up control and hands use of the GIL over to a different thread.

In the case of my "retrieve" program, this is perfect. I can spawn a separate thread to retrieve each of the URLs in the array. I then can wait for the URLs to be retrieved in parallel, checking in with each of the threads one at a time. In this way, I probably can save time.

Let's start with the core of my rewritten program. I'll want to implement the retrieval as a function, and then invoke that function along with one argument—the URL I want to retrieve. I then can invoke that function by creating a new instance of `threading.Thread`, telling the new instance not only which function I want to run in a new thread, but also which argument(s) I want to pass. This is how that code will look:

```
for one_url in urls:
    t = threading.Thread(target=get_length, args=(one_url,))
    t.start()
```

But wait. How will the `get_length` function communicate the content length to the rest of the program? In a threaded program, you really must not have individual threads modify built-in data structures, such as a list. This is because such data structures aren't thread-safe, and doing something such as an "append" from one thread might cause all sorts of problems.

However, you can use a "queue" data structure, which is thread-safe, and thus guarantees a form of communication. The function can put its results on the queue, and then, when all of the threads have completed their run, you can read those results from the queue.

Here, then, is how the function might look:

```
from queue import Queue

queue = Queue()

def get_length(one_url):
    response = requests.get(one_url)
    queue.put((one_url, len(response.content)))
```

As you can see, the function retrieves the content of `one_url` and then places the URL itself, as well as the length of the content, in a tuple. That tuple is then placed in the queue.

It's a nice little program. The main thread spawns a new thread, each of which runs `get_length`. In `get_length`, the information gets stuck on the queue.

The thing is, now it needs to retrieve things from the queue. But if you do this just after launching the threads, you run the risk of reading from the queue before the threads have completed. So, you need to "join" the threads, which means to wait until they have finished. Once the threads have all been joined, you can read all of their information from the queue.

There are a few different ways to join the threads. An easy one is to create a list where you will store the threads and then append each new thread object to that list as you create it:

```
threads = [ ]

for one_url in urls:
    t = threading.Thread(target=get_length, args=(one_url,))
    threads.append(t)
    t.start()
```

You then can iterate over each of the thread objects, joining them:

```
for one_thread in threads:
    one_thread.join()
```

Listing 3. retrieve2.py

```
#!/usr/bin/env python3

import requests
import time
import threading
from queue import Queue

urls = [one_line.strip()
        for one_line in open('urls.txt')]

length = {}
queue = Queue()
start_time = time.time()
threads = [ ]

def get_length(one_url):
    response = requests.get(one_url)
    queue.put((one_url, len(response.content)))

# Launch our function in a thread
print("Launching")
for one_url in urls:
    t = threading.Thread(target=get_length, args=(one_url,))
    threads.append(t)
    t.start()

# Joining all
print("Joining")
for one_thread in threads:
    one_thread.join()

# Retrieving + printing
print("Retrieving + printing")
while not queue.empty():
    one_url, length = queue.get()
    print("{0:30}: {1:8,}".format(one_url, length))

end_time = time.time()

total_time = end_time - start_time

print("\nTotal time: {0:.3} seconds".format(total_time))
```

Note that when you call `one_thread.join()` in this way, the call blocks. Perhaps that's not the most efficient way to do things, but in my experiments, it still took about one second—15 times faster—to retrieve all of the URLs.

In other words, Python threads are routinely seen as terrible and useless. But in this case, you can see that they allowed me to parallelize the program without too much trouble, having different sections execute concurrently.

Considerations

The good news is that this demonstrates how using threads can be effective when you're doing numerous, time-intensive I/O actions. This is especially good news if you're writing a server in Python that uses threads; you can open up a new thread for each incoming request and/or allocate each new request to an existing, pre-created thread. Again, if the threads don't really need to execute in a truly parallel fashion, you're fine.

But, what if your system receives a very large number of requests? In such a case, your threads might not be able to keep up. This is particularly true if the code being executed in each thread is CPU-intensive.

In such a case, you don't want to use threads. A popular option—indeed, *the* popular option—is to use processes. In my next article, I plan to look at how such processes can work and interact.■

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

Ubuntu Linux and Bash as a Windows Program!

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.

An Ubuntu Bash shell as a Windows app? Fantastic! Here's how to proceed.

PREVIOUS
Reuven M. Lerner's
At the Forge

NEXT
Kyle Rankin's
Hack and /

MICROSOFT HAS RULED THE OPERATING SYSTEM WORLD FOR MANY YEARS, and it's staggering to see how many computers run Microsoft Windows in all its many variants. In May 2017, Microsoft announced it had an installed base of 1.4 billion. Sure, there also are much more than a billion devices now running Android, but if you're running a desktop or laptop computer, odds remain that it's running some version of Microsoft Windows.

Microsoft hasn't just ignored the rest of the OS world—although it sure took a long time for them to bail on Windows Mobile and accept Android—so it was with delight that I read that the next version of Windows 10 would include a simple app that lets you run a Linux shell.

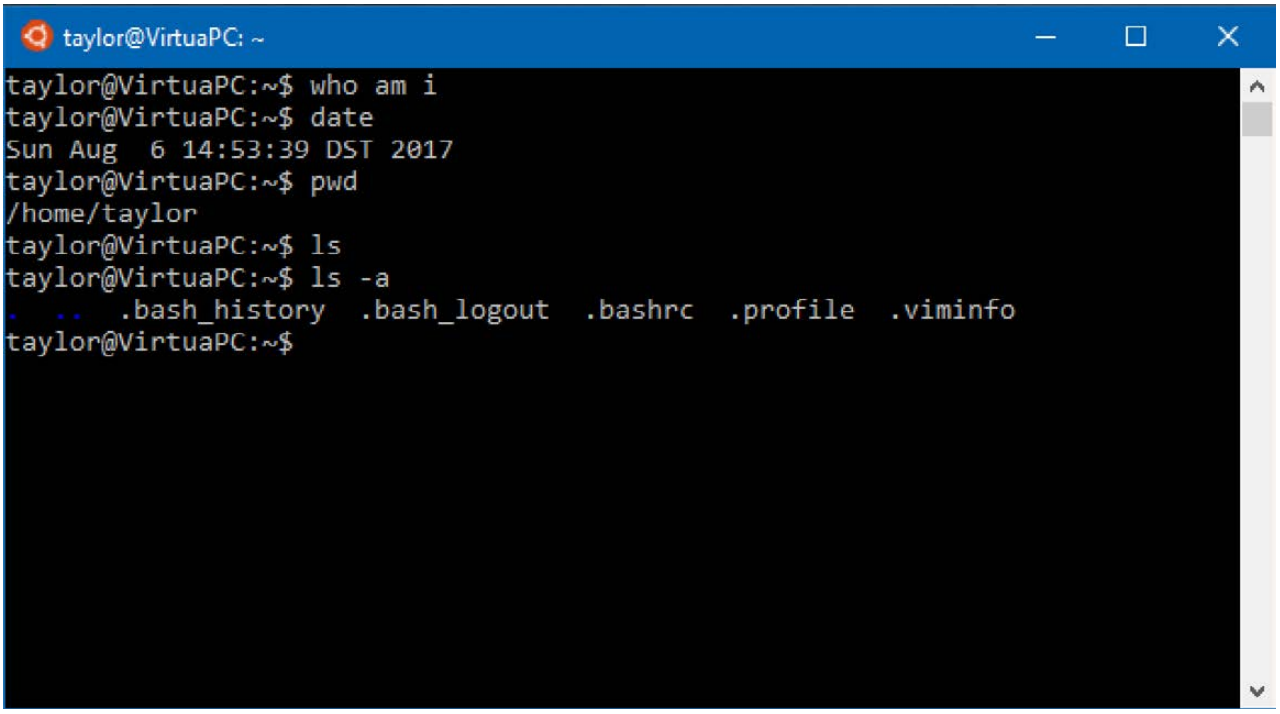
No hassles with complex installations, no dual booting, no virtual machines you need to configure—just a simple app to find in the Microsoft Store and a program to click and run whenever you want to expand your knowledge of Bash or Linux.

Sort of.

At the time of this writing, the app requires that you're in the early release program (it's free to join, and you'll become a beta tester for the next version of Windows 10), but once you're signed up and running the early release version, it is indeed a download-and-go program.

By the time you read this article, however, what I'm running as an early release of Win10 should be the latest public update, so that roadblock will vanish.

To get started, you'll need to enable the Windows Subsystem for Linux (which also lets you run other flavors of Linux and is pretty cool), but start by searching in the Microsoft Store for "Ubuntu" to find Ubuntu Linux on Windows. The latest version of Ubuntu is supported in the app too: 16.04 LTS.



```
taylor@VirtuaPC:~$ who am i
taylor@VirtuaPC:~$ date
Sun Aug  6 14:53:39 DST 2017
taylor@VirtuaPC:~$ pwd
/home/taylor
taylor@VirtuaPC:~$ ls
taylor@VirtuaPC:~$ ls -a
.  ..  .bash_history  .bash_logout  .bashrc  .profile  .viminfo
taylor@VirtuaPC:~$
```

Figure 1. Ubuntu as a Windows App? Excellent!

Click to launch it once installed, and you'll be confronted with a window like the one shown in Figure 1.

It's interesting to note that some standard Linux commands return nothing—like `who am i`—but most of them work fine, and there even are dot files in my newly created home directory /home/taylor.

This is a Bash login shell. You can confirm your own shell a couple different ways, but I like this command: `ps $$`.

Are you curious about how the system identifies itself? `uname -a` is the standard Linux command to get the version, which reveals this interesting info:

```
$ uname -a
Linux VirtuaPC 4.4.0-43-Microsoft #1-Microsoft
 ➥Wed Dec 31 14:42:53 PST 2014
x86_64 x86_64 x86_64 GNU/Linux
```

VirtuaPC is indeed a virtual machine system (you can learn more about it at the Microsoft.com download center, if you're curious), and it's all part of the install, all hidden from us users. Thank goodness; VM installs can be tricky to configure properly.

What about Bash itself? It's easy to identify versions by using the `--version` flag:

```
$ bash --version
GNU bash, version 4.3.11(1)-release (x86_64-pc-linux-gnu)
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later
 ➥<http://gnu.org/licenses/gpl.html>

This is free software; you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
```

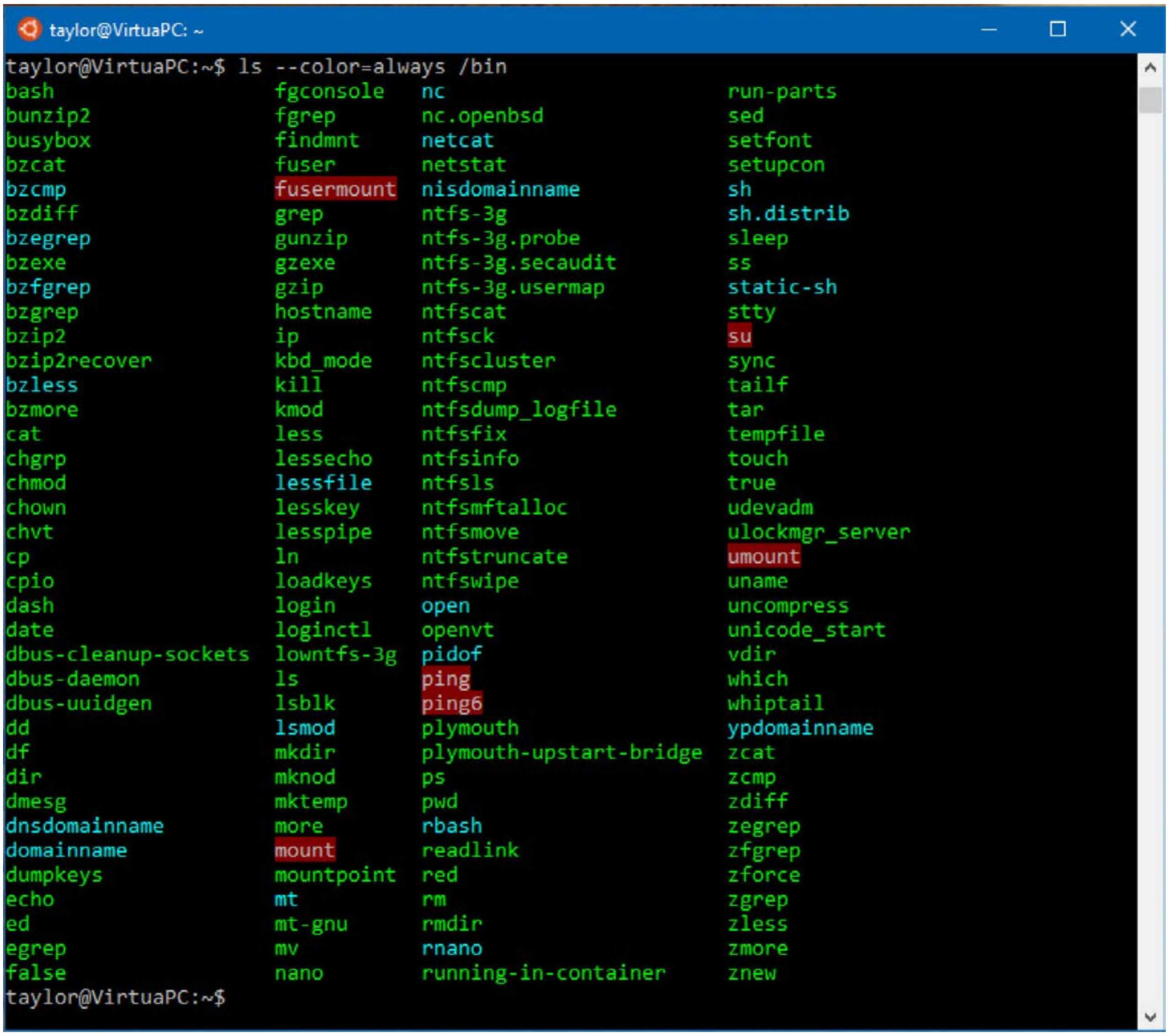
Version 4.3.11 is reasonably current, although the GNU archive shows the latest version is 4.4 (you can go to http://ftp.gnu.org/gnu/bash to see the latest).

Better yet, you're not quite in the late 1980s with the command line either, as the default TERM identifies itself to programs not as ANSI but as xterm-256color. This means you can have colors included in ls output, grep results and so on. Whether it looks nice and is legible is another story.

Let's give it a whirl—see Figure 2.

Changing the font size, window color scheme and so on is tricky because there's no UI to the Ubuntu program in Windows. It's also hard to get into the X Window System underlying the Ubuntu server that's running on your Windows system, so the standard trick of changing settings in .Xdefaults won't work.

Fortunately, you at least can improve the color scheme somewhat with the setterm command, then axe the default aliases to remove the clumsy



taylor@VirtuaPC: ~

```
taylor@VirtuaPC:~$ ls --color=always /bin
bash                  fgconsole   nc                       run-parts
bunzip2               fgrep       nc.openbsd               sed
busybox               findmnt     netcat                   setfont
bzcat                 fuser       netstat                  setupcon
bzcmp                 fusermount  nisdomainname            sh
bzdiff                grep        ntfs-3g                  sh.distrib
bzegrep               gunzip      ntfs-3g.probe            sleep
bzexe                 gzexe       ntfs-3g.secaudit         ss
bzfgrep               gzip        ntfs-3g.usermap          static-sh
bzgrep                hostname    ntfscat                  stty
bzip2                 ip          ntfsck                   su
bzip2recover          kbd_mode    ntfscluster              sync
bzless                kill        ntfscmp                  tailf
bzmore                kmod        ntfsdump_logfile         tar
cat                   less        ntfsfix                  tempfile
chgrp                 lessecho    ntfsinfo                 touch
chmod                 lessfile    ntfsls                   true
chown                 lesskey     ntfsmftalloc             udevadm
chvt                  lesspipe    ntfsmove                 ulockmgr_server
cp                    ln          ntfstruncate             umount
cpio                  loadkeys    ntfswipe                 uname
dash                  login       open                     uncompress
date                  loginctl    openvt                   unicode_start
dbus-cleanup-sockets  lowntfs-3g  pidof                    vdir
dbus-daemon           ls          ping                     which
dbus-uuidgen          lsblk       ping6                    whiptail
dd                    lsmod       plymouth                 ypdomainname
df                    mkdir       plymouth-upstart-bridge  zcat
dir                   mknod       ps                       zcmp
dmesg                 mktemp      pwd                      zdiff
dnsdomainname         more        rbash                    zegrep
domainname            mount       readlink                 zfgrep
dumpkeys              mountpoint  red                      zforce
echo                  mt          rm                       zgrep
ed                    mt-gnu      rmdir                    zless
egrep                 mv          rnano                    zmore
false                 nano        running-in-container     znew
taylor@VirtuaPC:~$
```

Figure 2. The Contents of /bin, in Glorious 256color

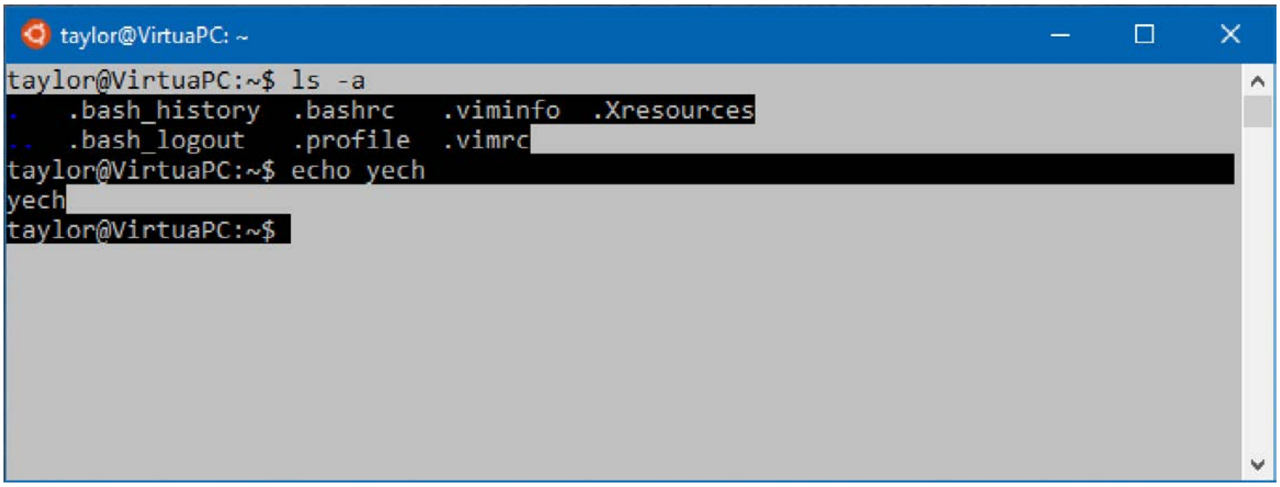
ANSI colors. This command produces a far more readable terminal screen:

```
setterm -term linux -back white -fore black -clear
```

Ubuntu or Microsoft, someone has set it so that "white" is actually a very light grey, so the results are quite attractive. The problem is, run a command that knows how to output in color, and things get pretty broken pretty fast, as you can see in Figure 3 when I tried an `ls -a`.

Open up the ~/.bashrc file, however, and you'll find that commands like `ls` have aliases that force the use of color. You also can check this with the `alias` command:

```
$ alias
alias alert='notify-send --urgency=low -i "$([ $? = 0 ] && echo
terminal || echo error)" "$(history|tail -n1|sed -e
 ➥'\''s/^\s*[0-9]\+\s*//;s/[;&|]\s*alert$//'\'')"'alias
 ➥egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias grep='grep --color=auto'
alias l='ls -CF'
alias la='ls -A'
alias ll='ls -alF'
alias ls='ls --color=auto'
```



```
taylor@VirtuaPC:~$ ls -a
.    .bash_history  .bashrc    .viminfo  .Xresources
..   .bash_logout   .profile   .vimrc
taylor@VirtuaPC:~$ echo yech
yech
taylor@VirtuaPC:~$
```

Figure 3. New Color Scheme, Overwritten by LS

The solution is to go into the ~/.bashrc and comment out (preface each line with a # symbol) all of these aliases, and you'll be very close to Windows/Ubuntu configuration nirvana.

The only piece left that I kept hitting was the default ability of the vi editor (yeah, not emacs) to use color. Open up a file like my .profile, and it's out-of-control color soup and completely unreadable, as you can see in Figure 4.

Again, it's fixable. This time, you want to create a new file in your home directory called .vimrc that contains this single line:

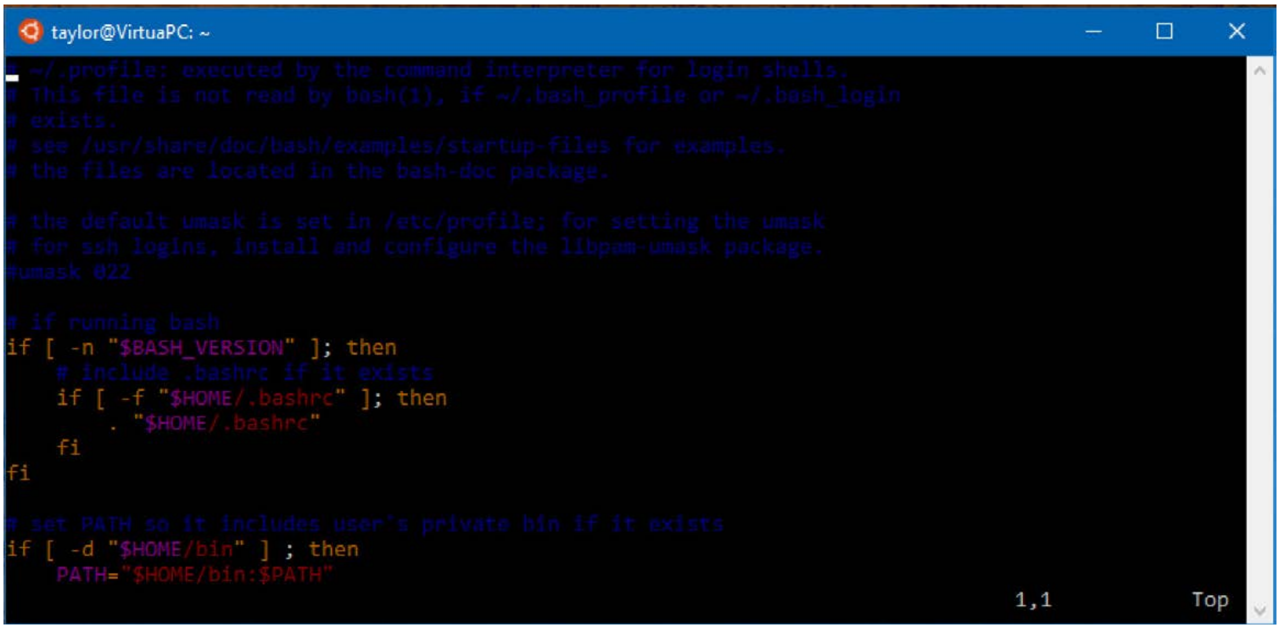
```
syntax off
```

Confusingly, `syntax` is the parsing and display of different file elements in different colors. Can't read the screen to create this file? In vi type this:

```
:syntax off
```

and it'll turn off all that crazy color stuff.

Those few tweaks will get you much farther toward being able to



```
# ~/.profile: executed by the command interpreter for login shells.
# This file is not read by bash(1), if ~/.bash_profile or ~/.bash_login
# exists.
# see /usr/share/doc/bash/examples/startup-files for examples.
# the files are located in the bash-doc package.

# the default umask is set in /etc/profile; for setting the umask
# for ssh logins, install and configure the libpam-umask package.
#umask 022

# if running bash
if [ -n "$BASH_VERSION" ]; then
    # include .bashrc if it exists
    if [ -f "$HOME/.bashrc" ]; then
        . "$HOME/.bashrc"
    fi
fi

# set PATH so it includes user's private bin if it exists
if [ -d "$HOME/bin" ] ; then
    PATH="$HOME/bin:$PATH"
                                                            1,1           Top
```

Figure 4. Vi's color scheme is ghastly!

use and easily read the Ubuntu window in Windows until there’s an actual settings or preferences option that becomes available.

Give it a try, and if you find out how to make the typeface bigger or any other useful configuration tweaks, send them along, and I’ll revisit this in a few months with the best user submissions.■

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

RETURN TO CONTENTS

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.

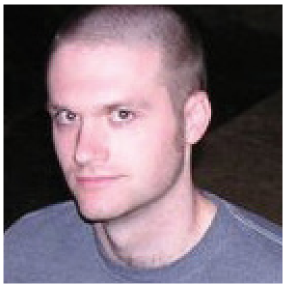
Get a job. Give a job. And invest in the future of Drupal when you do it.

Drupal™ Jobs

Drupal™ ASSOCIATION

On-Call or Travel Laptops

Why stress over losing that expensive personal or work laptop? Buy a cheap one for risky situations.



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.

PREVIOUS
Dave Taylor's
Work the Shell

NEXT
Shawn Powers'
The Open-Source
Classroom

IN THE AUGUST 2017 ISSUE, I WROTE ABOUT HOW TO PREPARE FOR A VACATION SO YOU AREN'T DISTURBED BY A WORK EMERGENCY. As part of that article, I described how to prepare your computer:

> Even better than taking a backup, leave your expensive work computer behind and use a cheaper more disposable machine for travel, and just restore your important files and settings for work on it before you leave and wipe it when you return. If you decide to go the disposable computer route, I recommend working one or two full work days

> on this computer before the vacation to make sure all of your files and settings are in place.

It turns out that this advice works not just for travel but also for a laptop you take with you while on call. So in this article, I elaborate on the above advice and describe some strategies for choosing and setting up an appropriate laptop to take with you while on call or traveling.

Why Choose a Different Laptop?

I was faced with the dilemma of choosing a travel laptop when I went on vacation a few months ago. I needed to be reachable while on vacation, just in case, but I knew I didn't want to lug around and cross borders with an expensive company laptop. There are a number of reasons why this is a good idea, and most of the reasons you would want to use a separate, cheap laptop for travel also apply for an on-call laptop.

Less Concern over Loss, Damage or Theft

Although it's true that your laptop might get lost, stolen or damaged while you commute to work, it's much more likely to happen outside your normal work routine. While you are on call, you might take your laptop to restaurants, bars, events or a friend's house, and because you are outside your normal routine, it's more likely that it will be stolen or that you might accidentally leave it behind. Also when you are commuting to work, you likely have some kind of backpack or case for your laptop, but outside work, you may be more likely just to throw your laptop in the trunk of your car.

While traveling, especially traveling abroad, you are most definitely outside your normal routine, and a laptop is even more likely to get lost, damaged or stolen. The more expensive laptop you have with you, the more enticing of a target, and the more you have to lose. Also, with increased security around airports and customs these days, laptops are more likely to be inspected, confiscated or forced into checked luggage. Plus, if you do have to put your laptop in checked luggage, these days, you must lock your luggage with keys that security agents can unlock. Unfortunately, there are many stories of unscrupulous airport employees who have taken advantage of this fact to steal high-value items from luggage while it's out of its owner's possession.

An Immediate Backup for Your Work Laptop

Having a second laptop that's ready at any moment to take over work duties adds an extra backup in case your work laptop itself breaks. Instead of being out of commission while you are waiting for a replacement, you immediately can resume work on your backup. It also provides you with a backup in case you leave your work laptop at the office.

How to Choose Your Laptop

The key to a good on-call or travel laptop is to get something cheap. As computers have continued to get faster, the fact is that many people can get their general work done (especially in a pinch) with laptops that are many years old. This is especially true on a Linux desktop, even if you aren't someone who spends a decent amount of time on a terminal.

Used Thinkpads are a great choice for travel laptops, because they have good Linux compatibility and are rugged and easy to repair with replacement parts that are easy to find. Because so many organizations have used them as company laptops, you almost always can find a used one cheap on an auction site. Keep an eye out for a model that is listed as having no OS. Those laptops tend to be cheaper because people want to avoid having to install an OS, but as Linux users, we would just overwrite the OS anyway! I've consistently found that if I'm patient, I can get a Thinkpad with reasonable specs for less than $50 on auction sites. If you are willing to splurge on extra RAM or an SSD, these old machines can be surprisingly speedy.

Another option, especially if you want a more portable laptop, is a Chromebook. Although these machines normally are designed to run a limited, secured OS that centers on Google services, they also can run Linux well once you switch into developer mode. Some people use cheap Chromebooks as their default travel computers since they just want to check Gmail and browse the web while traveling. Personally, I found a used Acer C710 for $40 and was able to add RAM and an SSD from a spare Thinkpad, and it turned out to be a rather capable Qubes-compatible machine.

Setting Up Your Laptop

I use Qubes both on my work and personal laptops, and I've long used its built-in backup and restore tool whenever I travel to make sure I have a fresh backup in case my laptop is lost or stolen. Now that I rely on a separate laptop for travel, I just restore that fresh backup onto my travel machine and test it by working on it for a day before the trip. This also means I can selectively restore only the files and settings (appVMs in my case) that are relevant for the situation. In the case of its use as an on-call computer, I don't have to worry as much about fresh backups as long as all of my VPN, SSH and other credentials are kept up to date.

Since most people don't use Qubes, just take advantage of whatever tool you prefer to back up your laptop (you do back up your laptop regularly, don't you?) and restore onto your spare computer as regularly as you need to keep important files up to date. Given that you are doing this to protect against the laptop being lost or stolen, be sure to enable full disk encryption when you install the OS to help protect your sensitive files just in case. For those of you who are extra security-conscious, you can take the additional step of wiping and re-installing your OS whenever you return from a long trip, just in case you are worried about any malware that found its way on your computer while you were on untrusted networks.

Conclusion

In general, I highly recommend selecting a cheap laptop for your on-call and travel computer. You will find you have extra peace of mind knowing that not only will it be inexpensive to replace your laptop if it's lost, broken or stolen, but also that you when you return home, you can get on your regular computer and get right back to work.■

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

RETURN TO CONTENTS

Ansible, Part III: Playbooks

Playbooks make Ansible even more powerful than before.

PREVIOUS
Kyle Rankin's Hack and /

NEXT
New Products

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.

TO BE QUITE HONEST, IF ANSIBLE HAD NOTHING BUT ITS AD-HOC MODE, IT STILL WOULD BE A POWERFUL AND USEFUL TOOL FOR AUTOMATING LARGE NUMBERS OF COMPUTERS. In fact, if it weren't for a few features, I might consider sticking with ad-hoc mode and adding a bunch of those ad-hoc commands to a Bash script and be done with learning. Those few additional features, however, make the continued effort well worth it.

Tame the Beast with YAML

Ansible goes out of its way to use an easy-to-read configuration file for making "playbooks", which are files full of separate Ansible "tasks". A task is basically an ad-hoc command written out in a configuration file that makes it more organized and easy to expand.

The configuration files use YAML, which stands for "Yet Another Markup Language". It's an easy-to-read markup language, but it does rely on whitespace, which isn't terribly common with most config files. A simple playbook looks something like this:

```
---

- hosts: webservers
  become: yes
  tasks:
    - name: this installs a package
      apt: name=apache2 update_cache=yes state=latest

    - name: this restarts the apache service
      service: name=apache2 enabled=yes state=restarted
```

The contents should be fairly easy to identify. It's basically two ad-hoc commands broken up into a YAML configuration file. There are a few important things to notice. First, every filename ends with .yaml, and every YAML file must begin with three hyphen characters. Also, as mentioned above, whitespace matters. Finally, when a hyphen should precede a section and when it should just be spaced appropriately often is confusing. Basically every new section needs to start with a - symbol, but it's often hard to tell what should be its own section. Nevertheless, it starts to feel natural as you create more and more playbooks.

The above playbook would be executed by typing:

```
ansible-playbook filename.yaml
```

And that is the equivalent of these two commands:

```
ansible webservers -b -m apt -a "name=apache2
 ➥update_cache=yes state=latest"
ansible webservers -b -m service -a "name=apache2
 ➥enabled=yes state=restarted"
```

Handling Your Handlers

But a bit of organization is really only the beginning of why playbooks are so powerful. First off, there's the idea of "Handlers", which are tasks that are executed only when "notified" that a task has made a change. How does that work exactly? Let's rewrite the above YAML file to make the second task a handler:

```
---

- hosts: webservers
  become: yes
  tasks:
    - name: this installs a package
      apt: name=apache2 update_cache=yes state=latest
      notify: enable apache

  handlers:
    - name: enable apache
      service: name=apache2 enabled=yes state=started
```

On the surface, this looks very similar to just executing multiple tasks. When the first task (installing Apache) executes, if a change is made, it notifies the "enable apache" handler, which makes sure Apache is enabled on boot and currently running. The significance is that if Apache is already installed, and no changes are made, the handler never is called. That makes the code much more efficient, but it also means no unnecessary interruption of the already running Apache process.

There are other subtle time-saving issues with handlers too—for example, multiple tasks can call a handler, but it executes only a single time regardless of how many times it's called. But the really significant thing to remember is that handlers are executed (notified) only when an Ansible task makes a change on the remote system.

Just the Facts, Ma'am

Variable substitution works quite simply inside a playbook. Here's a simple example:

```
---

- hosts: webservers
  become: yes
  vars:
    package_name: apache2
  tasks:
    - name: this installs a package
      apt: "name={{ package_name }} update_cache=yes state=latest"
      notify: enable apache

  handlers:
    - name: enable apache
      service: "name={{ package_name }} enabled=yes state=started"
```

It should be fairly easy to understand what's happening above. Note that I did put the entire module action section in quotes. It's not always required, but sometimes Ansible is funny about unquoted variable substitutions, so I always try to put things in quotes when variables are involved.

The really interesting thing about variables, however, are the "Gathered Facts" about every host. You might notice when executing a playbook that the first thing Ansible does is "Gathering Facts...", which completes without error, but doesn't actually seem to do anything. What's really happening is that system information is getting populated into variables that can be used inside a playbook. To see the entire list of "Gathered Facts", you can execute an ad-hoc command:

```
ansible webservers -m setup
```

You'll get a *huge* list of facts generated from the individual hosts. Some of them are particularly useful. For example, `ansible_os_family` will return something like "RedHat" or "Debian" depending on which distribution you're using. Ubuntu and Debian systems both return

If you're creating a playbook and want to be notified of things along the way, the debug module is really your friend.

"Debian", while Red Hat and CentOS will return "RedHat". Although that's certainly interesting information, it's really useful when different distros use different tools—for example, apt vs. yum.

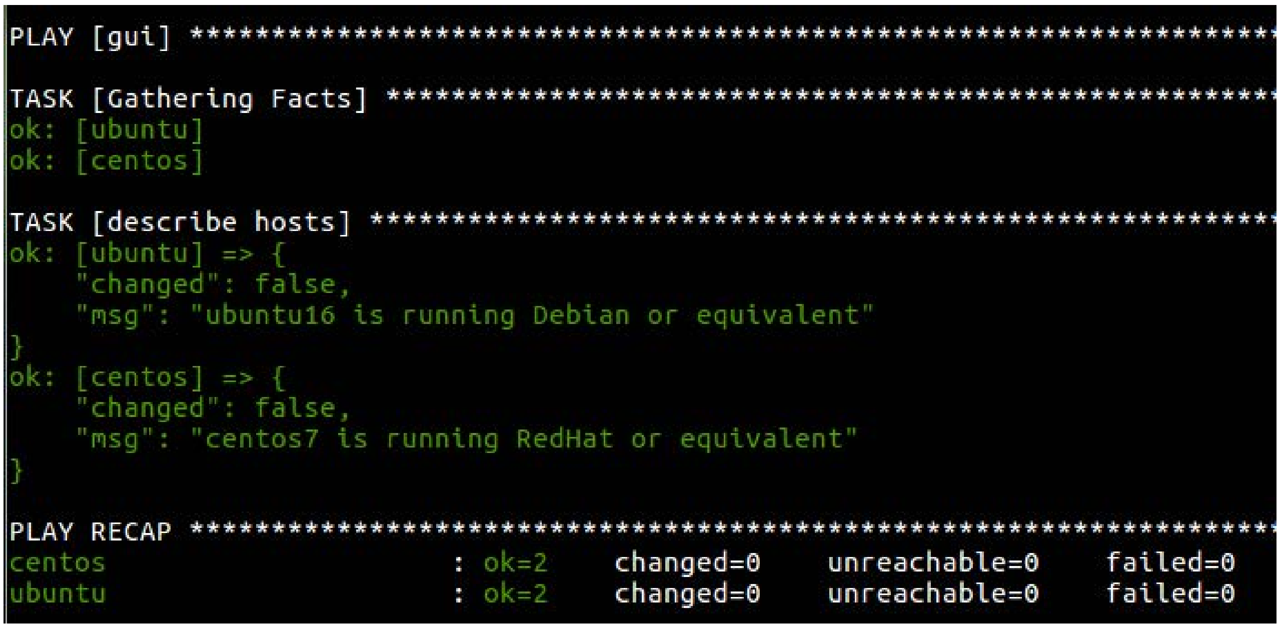
Getting Verbose

One of the frustrations of moving from Ansible ad-hoc commands to playbooks is that in playbook mode, Ansible tends to keep fairly quiet with regard to output. With ad-hoc mode, you often can see what is going on, but with a playbook, you know only if it finished okay, and if a change was made. There are two easy ways to change that. The first is just to add the `-v` flag when executing `ansible-playbook`. That adds verbosity and provides lots of feedback when things are executed. Unfortunately, it often gives so much information, that usefulness gets lost in the mix. Still, in a pinch, just adding the `-v` flag helps.

If you're creating a playbook and want to be notified of things along the way, the debug module is really your friend. In ad-hoc mode, the debug module doesn't make much sense to use, but in a playbook, it can act as a "reporting" tool about what is going on. For example:

```
---

- hosts: webservers
  tasks:
   - name: describe hosts
     debug: msg="Computer {{ ansible_hostname }} is running
      ➥{{ ansible_os_family }} or equivalent"
```

The above will show you something like Figure 1, which is incredibly



```
PLAY [gui] ********************************************************************

TASK [Gathering Facts] ********************************************************
ok: [ubuntu]
ok: [centos]

TASK [describe hosts] *********************************************************
ok: [ubuntu] => {
    "changed": false,
    "msg": "ubuntu16 is running Debian or equivalent"
}
ok: [centos] => {
    "changed": false,
    "msg": "centos7 is running RedHat or equivalent"
}

PLAY RECAP ********************************************************************
centos                     : ok=2    changed=0    unreachable=0    failed=0
ubuntu                     : ok=2    changed=0    unreachable=0    failed=0
```

Figure 1. Debug mode is the best way to get some information on what's happening inside your playbooks.

useful when you're trying to figure out the sort of systems you're using. The nice thing about the debug module is that it can display anything you want, so if a value changes, you can have it displayed on the screen so you can troubleshoot a playbook that isn't working like you expect it to work. It is important to note that the debug module doesn't do anything other than display information on the screen for you. It's not a logging system; rather, it's just a way to have information (customized information, unlike the verbose flag) displayed during execution. Still, it can be invaluable as your playbooks become more complex.

If This Then That

Conditionals are a part of pretty much every programming language. Ansible YAML files also can take advantage of conditional execution, but the format is a little wacky. Normally the condition comes first, and then if it evaluates as true, the following code executes. With Ansible, it's a little backward. The task is completely spelled out, then a `when` statement is added at the end. It makes the code very readable, but as someone who's been using if/then mentality his entire career, it feels funny. Here's a slightly more complicated playbook. See if you can parse out what would happen in an environment with both Debian/Ubuntu and Red Hat/CentOS systems:

```
---

- hosts: webservers
  become: yes
  tasks:
    - name: install apache this way
      apt: name=apache2 update_cache=yes state=latest
      notify: start apache2
      when: ansible_os_family == "Debian"

    - name: install apache that way
      yum: name=httpd state=latest
      notify: start httpd
      when: ansible_os_family == "RedHat"

  handlers:
    - name: start apache2
      service: name=apache2 enabled=yes state=started

    - name: start httpd
      service: name=httpd enabled=yes state=started
```

Hopefully the YAML format makes that fairly easy to read. Basically, it's a playbook that will install Apache on hosts using either yum or apt based on which type of distro they have installed. Then handlers make sure the newly installed packages are enabled and running.

It's easy to see how useful a combination of gathered facts and conditional statements can be. Thankfully, Ansible doesn't stop there. As with other configuration management systems, it includes most features of programming and scripting languages. For example, there are loops.

Play It Again, Sam

If there is one thing Ansible does well, it's loops. Quite frankly, it supports so many different sorts of loops, I can't cover them all here. The best way to figure out the perfect sort of loop for your situation is to read the Ansible documentation directly at http://docs.ansible.com/ansible/latest/playbooks_loops.html.

For simple lists, playbooks use a familiar, easy-to-read method for doing multiple tasks. For example:

```
---

- hosts: webservers
  become: yes

  tasks:
    - name: install a bunch of stuff
      apt: "name={{ item }} state=latest update_cache=yes"
      with_items:
        - apache2
        - vim
        - chromium-browser
```

This simple playbook will install multiple packages using the apt module. Note the special variable named `item`, which is replaced with the items one at a time in the `with_items` section. Again, this is pretty easy to understand and utilize in your own playbooks. Other loops work in similar ways, but they're formatted differently. Just check out the documentation for the wide variety of ways Ansible can repeat similar tasks.

Templates

One last module I find myself using often is the template module. If you've ever used mail merge in a word processor, templating works similarly. Basically, you create a text file and then use variable substitution to create a custom version on the fly. I most often do this for creating HTML files or config files. Ansible uses the Jinja2 templating language, which is conveniently similar to standard variable substitution in play books themselves. The example I almost always use is a custom HTML file that can be installed on a remote batch of web servers. Let's look at a fairly complex playbook and an accompanying HTML template file.

Here's the playbook:

```
---

- hosts: webservers
  become: yes

  tasks:
   - name: install apache2
     apt: name=apache2 state=latest update_cache=yes
     when: ansible_os_family == "Debian"

   - name: install httpd
     yum: name=httpd state=latest
     when: ansible_os_family == "RedHat"

   - name: start apache2
     service: name=apache2 state=started enable=yes
     when: ansible_os_family == "Debian"

   - name: start httpd
     service: name=httpd state=started enable=yes
     when: ansible_os_family == "RedHat

   - name: install index
     template:
       src: index.html.j2
       dest: /var/www/html/index.html
```

Here's the template file, which must end in .j2 (it's the file referenced in the last task above):

```
<html><center>
<h1>This computer is running {{ ansible_os_family }},
and its hostname is:</h1>
```

```
<h3>{{ ansible_hostname }}</h3>
{# this is a comment, which won't be copied to the index.html file #}
</center></html>
```

This also should be fairly easy to understand. The playbook takes a few different things it learned and installs Apache on the remote systems, regardless of whether they are Red Hat- or Debian-based. Then, it starts the web servers and makes sure the web server starts on system boot. Finally, the playbook takes the template file, index.html.j2, and substitutes the variables while copying the file to the remote system. Note the `{# #}` format for making comments. Those comments are completely erased on the remote system and are visible only in the .j2 file on the Ansible machine.

The Sky Is the Limit!

I'll finish up this series in my next article, where I plan to cover how to build on your playbook knowledge to create entire roles and take advantage of the community contributions available. Ansible is a very powerful tool that is surprisingly simple to understand and use. If you've been experimenting with ad-hoc commands, I encourage you to create playbooks that will allow you to do multiple tasks on a multitude of computers with minimal effort. At the very least, play around with the "Facts" gathered by the ansible-playbook app, because those are things unavailable to the ad-hoc mode of Ansible. Until next time, learn, experiment, play and have fun!■

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

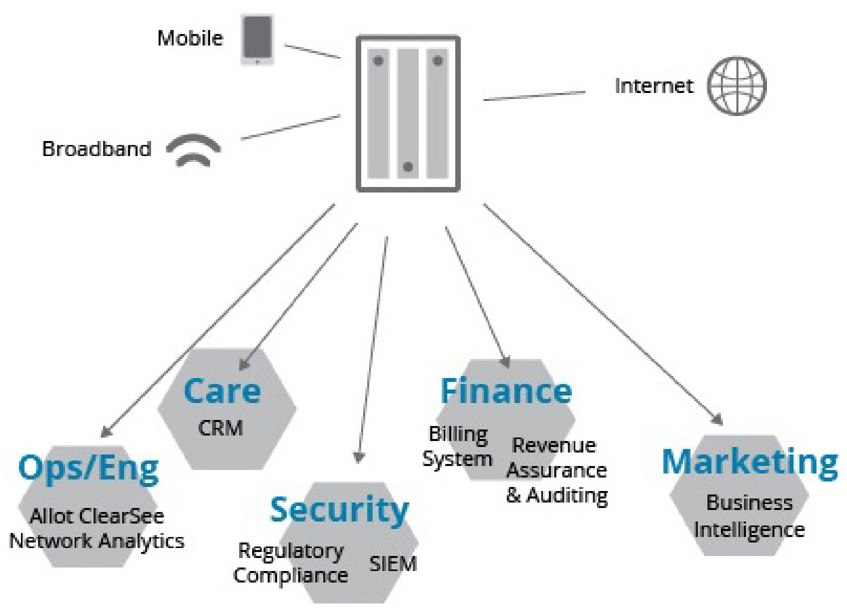
NEXT
Feature: Cool Project:
Create a CAPTCHA
for Your Website

LINBIT's DRBD Top

Many proprietary high-availability (HA) software providers require users to pay extra for system-management capabilities. Bucking this convention and driving down costs is LINBIT, whose DRBD HA software solution, part of the Linux kernel since 2009, powers thousands of digital enterprises. The cost savings originate from LINBIT's DRBD Top, a new software tool to simplify the management of the LINBIT DRBD application. Via DRBD Top's unified graphical interface, administrators can navigate their DRBD resources conveniently without typing multiple commands. Available on Github, DRBD Top provides critical status, assessment and troubleshooting capabilities for administrators who manage HA clusters, especially those with greater than two nodes.
https://www.linbit.com/en

iStorage diskAshur Storage Drives

With software-free setup and operation, the new iStorage diskAshur group of ultra-secure storage drives works across all operating systems, including Linux, macOS, Android, Chrome, thin and zero clients, MS Windows and embedded systems. Available in HDD and SDD versions, these high-speed USB 3.1, PIN-authenticated, hardware-encrypted portable data storage drives feature iStorage's unique EDGE technology. iStorage calls the EDGE technology—short for Enhanced Dual Generating Encryption—super-spy-like, due to the advanced security features that make diskAshur the "most secure data storage drives available on the market". For one thing, without the PIN, there's no way in! diskAshur's dedicated, hardware-based secure microprocessor (Common Criteria EAL4+-ready) employs built-in physical protection mechanisms designed to defend against external tamper, bypass laser attacks and fault injections. The drives feature technology that encrypts both the data and the encryption key, ensuring that private information is secure and protected. Other security features include a brute-force hack defense mechanism, self-destruct feature, unattended auto-lock and a wear-resistant epoxy-coated keypad. The diskAshur drives are elegantly designed and available in four striking colors and in capacity options from 128GB to 5TB.
https://istorage-uk.com



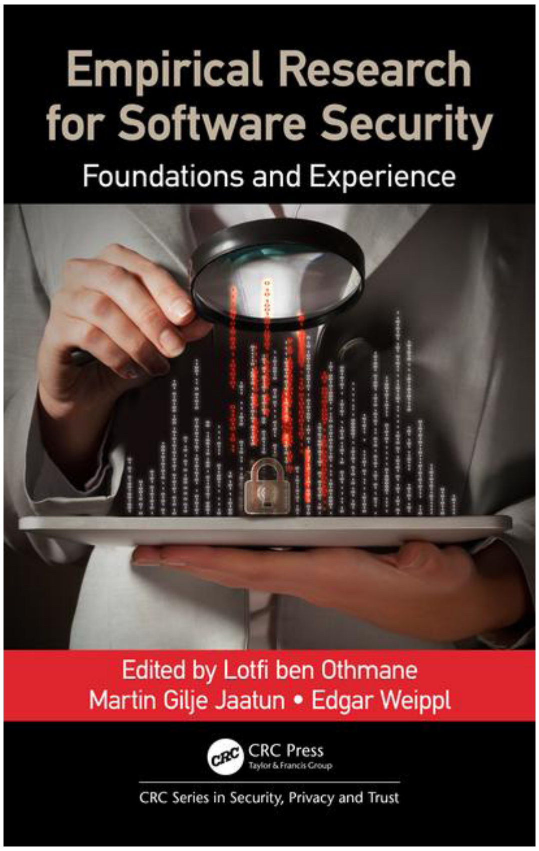
Datamation's "Leading Big Data Companies" Report

The Big Data market is in a period of remarkable transition. If keeping tabs on this dynamic sector is in your wheelhouse, Datamation has made your homework easier by developing "Leading Big Data Companies", a report that provides "a snapshot of a market sector in transition". Ranging from established legacy vendors to start-ups, this report details the numerous strategies that are exploited in today's Big Data landscape. The core technologies employed by this diverse group of vendors include cloud, open source, AI and several others. This report belongs to Datamation's ongoing focus on the latest emerging tech for the enterprise. In the mere seven years that have passed since Yahoo! introduced Hadoop, Big Data has burgeoned in popularity as ever more firms seek insights from the massive amounts of data at their disposal. Because Big Data has matured differently from most technologies in that no single leader has emerged after nearly a decade, the analytics industry finds itself still in growth mode, making it dynamic and challenging for those trying to make sense of it on their own.
http://www.datamation.com

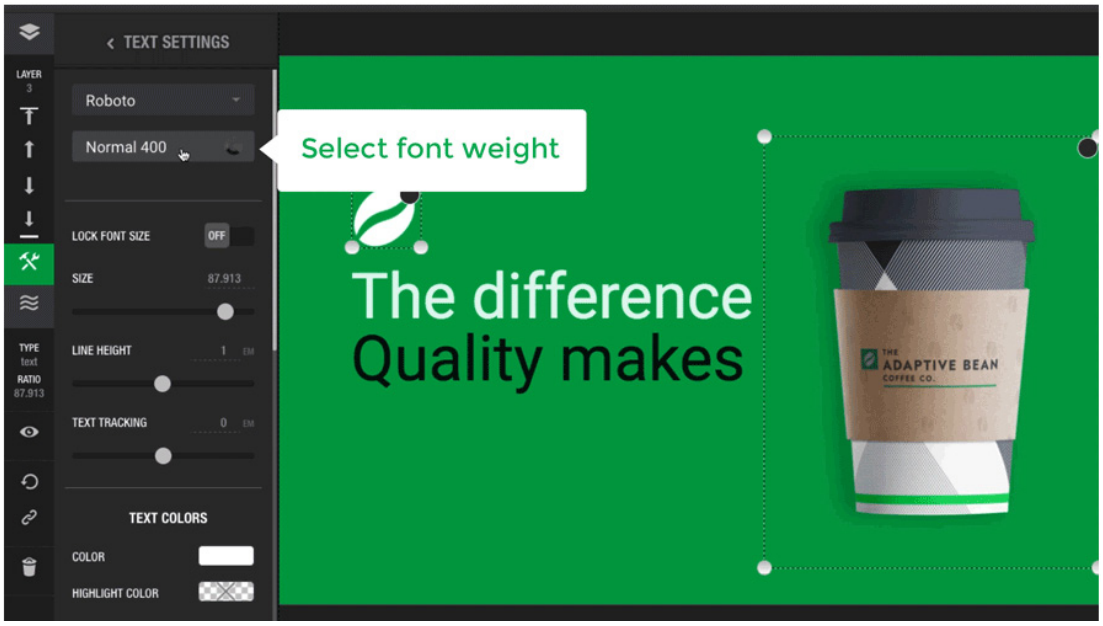
PSSC Labs' PowerServe HPC Servers and PowerWulf HPC Clusters

In its quest to provide customers the latest and best computing solutions that deliver relentless performance with the absolute lowest TCO, PSSC Labs has supercharged two server solutions with next-generation processing power. The breakthrough technology of Intel's new Xeon Scalable Processors has been integrated into PSSC Labs' PowerServe HPC line of servers and the PowerWulf line of HPC clusters, a move that guarantees performance capable of handling cutting-edge computing tasks, such as real-time analytics, virtualized infrastructure and high-performance computing. Besides the advanced architecture, the new processors offer a diverse suite of platform innovations for enhanced application performance including Intel AVX-512, Intel Mesh Architecture, Intel QuickAssist, Intel Optane SSDs and Intel Omni-Path Fabric. Both PSSC Labs solutions are designed for reliable, flexible, HPC solutions targeted at government, academic and commercial environments. Some examples of sectors that will benefit from the new performance include design and engineering, life and physical sciences, financial services and machine/deep learning.
http://www.pssclabs.com

Lotfi ben Othmane, Martin Gilje Jaatun and Edgar Weippl's *Empirical Research for Software Security* (CRC Press)


Empirical Research for Software Security
Foundations and Experience
Edited by Lotfi ben Othmane
Martin Gilje Jaatun • Edgar Weippl
CRC Press Taylor & Francis Group
CRC Series in Security, Privacy and Trust

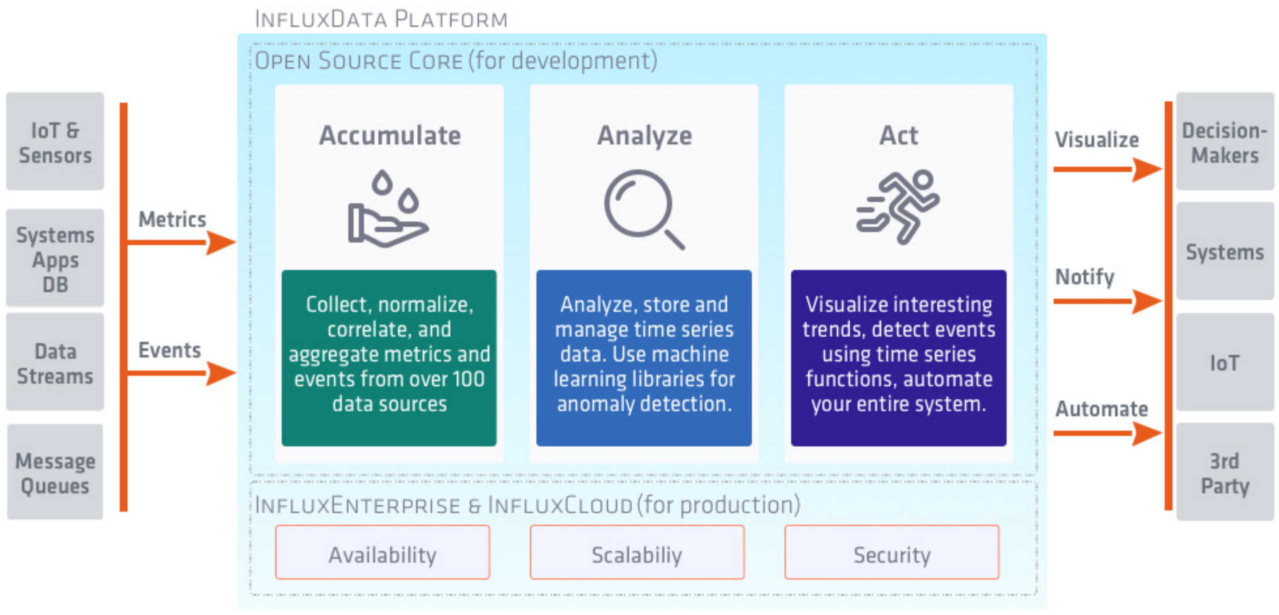
Developing truly secure software is no walk through the park. In an effort to apply the scientific method to the art of secure software development, a trio of authors—Lotfi ben Othmane, Martin Gilje Jaatun and Edgar Weippl—teamed up to write *Empirical Research for Software Security: Foundations and Experience*. The book is a guide for using empirical research methods to study secure software challenges. Empirical methods, including data analytics, allow extraction of knowledge and insights from the data that organizations gather from their tools and processes, as well as from the opinions of the experts who practice those methods and processes. These methods can be used to perfect a secure software development lifecycle based on empirical data and published industry best practices. The book also features examples that illustrate the application of data analytics in the context of secure software engineering.
https://www.crcpress.com



Roboto
Normal 400
LOCK FONT SIZE OFF
SIZE 87.913
LINE HEIGHT 1 EM
TEXT TRACKING 0 EM
LAYER 3
TYPE text
RATIO 87.913

Neuranet's Flexitive

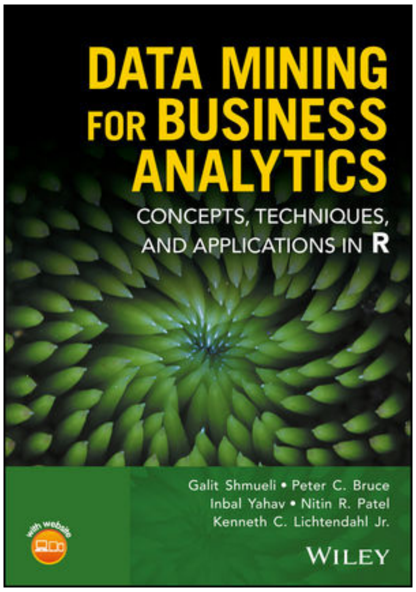
The new Interactive Advertising Bureau (IAB) Standard Ad Unit Portfolio's support for flexible ads is intended to improve the ad experience for users and boost revenue potential for advertisers. An updated solution from Neuranet, its Flexitive 2.0 responsive design software is designed to solve today's design challenges and give media companies and agencies a significant advantage in adapting to the IAB's new Flex Specifications and Lean Guidelines that were released in summer 2017. Flexitive 2.0 is the next generation of Neuranet's HTML5 cloud-based platform that supports more advanced responsive design capabilities for building high-quality, animated HTML5-based designs that adapt to unlimited sizes across any device, operating system, app or browser. The new standards were designed to promote user-friendly digital advertising that can scale across device types easily. Flexitive also incorporates the LEAN principles of lightweight, encrypted, AdChoices-supported and non-invasive advertising. Neuranet emphasizes other key features of Flexitive 2.0, such as an easy-to-use drag-and-drop interface that requires no coding knowledge, unlimited sizing of a single design, two-click creative design variations with instant scaling, export for use in more than 30 ad servers and more.
https://flexitive.com


InfluxData Platform
Open Source Core (for development)
IoT & Sensors
Systems Apps DB
Data Streams
Message Queues
Metrics
Events
Accumulate
Collect, normalize, correlate, and aggregate metrics and events from over 100 data sources
Analyze
Analyze, store and manage time series data. Use machine learning libraries for anomaly detection.
Act
Visualize interesting trends, detect events using time series functions, automate your entire system.
InfluxEnterprise & InfluxCloud (for production)
Availability
Scalabiliy
Security
Visualize
Notify
Automate
Decision-Makers
Systems
IoT
3rd Party

InfluxData

What is ephemeral data, you ask? InfluxData can supply the answer, because handling it is the business of the company's InfluxData open-source platform that is custom-built for metrics and events. Ephemeral data is transitory, existing only briefly, and is becoming vital for modern applications built where containers, microservices and sensors can come and go and are intermittently connected. The updated InfluxData 1.3 Platform can handle a billion (yes, with a "b"!) unique time series, making it easier to handle ephemeral data coming from containers or adding and removing sensors in IoT-tracking systems. InfluxData addresses the explosion of data points and sources, monitoring and controls requiring nanosecond precision coming from sensors and microservices. The InfluxData platform provides a comprehensive set of tools and services to accumulate metrics and events data, analyze the data and act on the data via powerful visualizations and notifications. New features in release 1.3 include time-series indexing, high-availability anomaly detection, query language improvements and automatic cluster rebalancing. InfluxData calls the new release "one of the most significant technical advancements in the platform to date".
https://www.influxdata.com

Galit Shmueli et al.'s *Data Mining for Business Analytics* (Wiley)


DATA MINING FOR BUSINESS ANALYTICS
CONCEPTS, TECHNIQUES, AND APPLICATIONS IN R
Galit Shmueli • Peter C. Bruce
Inbal Yahav • Nitin R. Patel
Kenneth C. Lichtendahl Jr.
with website
WILEY

The updated 5th edition of the book *Data Mining for Business Analytics* from Galit Shmueli and collaborators is a standard guide to data mining and analytics that adds two new co-authors and a trove of new material vis-á-vis its predecessor. R is a free, open-source and popularity-gaining software environment for statistical computing and graphics. Trailing with the subtitle *Concepts, Techniques, and Applications in R*, the new 5th edition of *Data Mining for Business Analytics* continues to provide an applied approach to data-mining concepts and methods, using the R software as a canvas on which to illustrate. With the book, readers learn how to implement a variety of popular data-mining algorithms in R to tackle business problems and opportunities. Material covered in-depth includes both statistical and machine-learning algorithms for prediction, classification, visualization, dimension reduction, recommender systems, clustering, text mining and network analysis. The new 5th edition includes material from business, government, a dozen case studies demonstrating applications for the data-mining techniques described, and exercises in each chapter that help readers gauge and expand their comprehension and competency of the material. *Data Mining for Business Analytics* can serve as either a text book or a reference for analysts, researchers and practitioners working with quantitative methods in myriad fields.
http://wiley.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

Cool Project: Create a CAPTCHA for Your Website

Jim's CAPTCHA uses photos of his cats, so only friends and family are allowed to view his personal website. You could modify the CAPTCHA to use other photos, like pictures of doors or trees, but I think everyone agrees that the internet needs more photos of cats. Note, this method works for dogs as well.

JIM HALL

PREVIOUS
New Products

NEXT
Feature: Developing Console Applications with Bash

I run a personal web server that I use for various projects and experiments. Over time, it's also become a convenient place for me to post other information to share with friends and family. For example, after much prompting by my family, I finally put up a "wish list" page for Christmas and birthday gift ideas.

Most of what I share on this personal website is intended for friends and family only, not for the general public. It's not sensitive information or private data, but I'd rather not put it out there for just anyone to see.

For a while, I considered protecting my web pages using a proper login system, but I didn't want to manage user names and passwords for all my friends and family who wanted to access my personal website. I might have implemented OpenID or some other decentralized authentication protocol to allow visitors to log in using another service. And although that would be the technically correct thing to do, I thought it was overkill to require a user name and password just for my friends and family to view my wish-list page.

So instead, I wrote a simple web system that asks visitors to demonstrate that they know me. This isn't a formal login system; rather, it's a form of CAPTCHA.

The concept of any CAPTCHA is to present some simple test for an *intended* audience to answer, but that's difficult for anyone else. Most CAPTCHAs try to present a trivial puzzle, such as clicking on any photos that contain water (like fountains or lakes). These CAPTCHAs often are used in online comment systems to verify that the person leaving the comment is a real person and not a bot.

In my case, I wanted to create a CAPTCHA that was simple for friends and family to answer, but difficult for others to figure out. And I decided to do it in a totally adorable way—by using photos of my cats.

My CAPTCHA

My CAPTCHA asks visitors to click on the photo of one our cats against a "forest" of other cats that don't belong to me. If you know me pretty well, this is easy. Obviously, my friends and family know what my cats look like. Selecting the photo of my cat is a simple demonstration that you know me; strangers would be unlikely to select the correct image.

Using photos of my cat isn't a perfect solution. It's entirely possible that attackers could use a brute-force method and keep clicking on photos of cats until they recognize which cats are mine. But this isn't a proper login system; it's just a CAPTCHA. I'm not protecting sensitive information like social security numbers. I merely want to prevent some random Joe User from seeing what books and t-shirts I'm asking for at Christmas.

In the simplest case, creating a cat CAPTCHA involves creating a random "forest" of cat photos, where only one of the cats belongs to you. You can create this "forest" in any number of ways. One easy way to do it in PHP is to create two lists: all images of my cats and all images of other people's cats. Pick one photo of my cat at random from the "my cat" list and several photos at random from the "not my cat" list, then shuffle the result.

So, you need to get started with photos of cats. This is the easiest step. Go through your photo collection and find pictures of your cats:

Figure 1. Jim's Cat

Figure 2. Jim's Other Cat

contented cats lying in the sun, sleepy cats napping on cushions, playful cats prowling the backyard, confused cats wondering why you're always pointing a camera at them.

Then, crop and resize your cat photos to the appropriate dimensions. So you can use the photos more easily in the CAPTCHA, crop the photos to be square, with the cat taking up most of the photo. In GIMP, this is easily done if you set the Crop tool to use a fixed aspect ratio of "1:1". At this stage, don't worry about the size of the cat photos; I'll describe how to set that in a later step.

Next, you need photos of other cats. You can find those from a variety of online photos, but I went on Facebook and looked for any cat photos posted by my friends. I found about 50 other cat photos this way. Just make sure the other cats don't look too much like your cats, and same as the previous step, crop the other cat photos so they are square.

To use these cat photos in a CAPTCHA, you want to ensure that the visitor cannot guess which cat is yours based on the photo's filenames. Photos named something like mycat_1.jpg can provide clear clues that the cat might belong to you. Instead, name all your photos in the same generic way. I prefer to rename my cat CAPTCHA photos according to the MD5 signature of the photo. This provides suitable non-predictability; you can't guess which photos are my cats and which are not simply based on the photo filename.

To make your pictures more alike, strip your cat photos of any EXIF data. While humans won't see the EXIF data, a bot might. EXIF data could provide a telltale hint that a photo is or isn't your cat. So make them all the same and omit EXIF data.

Note: if your CAPTCHA photos don't change over time, attackers could use a brute-force method to try all of the photos until they guess the MD5 signatures of the correct images. To protect against this, I add some random noise to the photos. I'm sure you can do this in GIMP, but I found it was easiest to use ImageMagick's Convert tool to add noise while I resized the cat photos:

```
convert large_photo.jpg -strip -resize 250x250 +noise
 ➥Laplacian small_photo.jpg
```

ImageMagick has several random noise generators, but I find the Laplacian method doesn't disturb the image too much.

You easily can automate ImageMagick with a script, and that's what I do. A simple Bash script processes all of my image files every day to resize them and add noise while renaming them to something random. This also merges the photos into a single directory and creates the lists of "my cats" and "not my cats". Put the list files in a separate directory, preferably outside the document root. On my personal web server, my document root is under /var/www/html, but I keep my list files in /var/www/etc/cats.

This script runs on my personal web server every day:

```
#!/bin/sh
etcdir=/var/www/etc/cats/
cachedir=/var/www/html/captcha/cache/
```

```
catsdir=$HOME/cats
tempimg=/tmp/cat.jpg


for d in mycats notmycats ; do
 ( cd $catsdir/$d
 >$etcdir/$d.list
 for img in *.jpg ; do
  convert $img -strip -resize 250x250 +noise Laplacian $tempimg
  hashval=$( md5sum $tempimg | awk '{print $1}' )
  mv $tempimg $cachedir/$hashval.jpg
  echo $hashval.jpg >> $etcdir/$d.list
 done )
done
```

This script adds all cat photos saved in $HOME/cats/mycats to the "my cats" list and all cat photos saved in $HOME/cats/notmycats to the "not my cats" list. ImageMagick removes all `EXIF` data (`-strip`), resizes all photos to the same size (`-resize 250x250`) and adds some random noise (`+noise Laplacian`). At the same time, all cat photos are mixed together with random filenames in the /var/www/html/captcha/cache directory.

Writing the PHP Code

Once you have the cat images, you need to write some PHP code that displays the cat images as a CAPTCHA. For my CAPTCHA, I prefer to display nine cat images in a 3x3 grid. One photo is my cat, and the other eight cats are not.

The first step is for the PHP script to know which cats are mine and which cats are not. That's where the mycats.list and notmycats.list files come in.

The `file()` function in PHP reads an entire file into an array—for example:

```
<?php
    $myCats = file('/var/www/etc/cats/mycats.list');
    $notMyCats = file('/var/www/etc/cats/notmycats.list');
?>
```

The `file()` function will include the newline characters at the end of each line, plus any empty lines that might be in the file. So you need to provide a few options to read the list files without the extra stuff:

```
<?php
    $myCats = file('/var/www/etc/cats/mycats.list',
     ➥FILE_IGNORE_NEW_LINES | FILE_SKIP_EMPTY_LINES);
    $notMyCats = file('/var/www/etc/cats/notmycats.list',
     ➥FILE_IGNORE_NEW_LINES | FILE_SKIP_EMPTY_LINES);
?>
```

PHP has a handy function `array_rand()` that picks random entries out of an array. This function returns the indices of random entries, which are easily used as array references. In the simplest case, you can select one cat randomly from the $myCats array like this:

```
<?php
    $myCats[ array_rand($myCats) ]
?>
```

Then build a new array of eight cats selected at random from the `$notMyCats` array and one random cat from the `$myCats` array by stacking the elements in the `array()` function. This builds a new array `$randCats` from the elements provided. As you can see, the new `$randCats` array is in a predictable order, with the correct cat always in the last position. So after building the array, randomize the order of the array elements with the `shuffle()` function:

```
<?php
    $notMyCatsIdx = array_rand($notMyCats, 8);
    $randCats = array(
            $notMyCats[ $notMyCatsIdx[0] ],
            $notMyCats[ $notMyCatsIdx[1] ],
            $notMyCats[ $notMyCatsIdx[2] ],
            $notMyCats[ $notMyCatsIdx[3] ],
            $notMyCats[ $notMyCatsIdx[4] ],
            $notMyCats[ $notMyCatsIdx[5] ],
            $notMyCats[ $notMyCatsIdx[6] ],
            $notMyCats[ $notMyCatsIdx[7] ],
            $myCats[ array_rand($myCats) ]
    );
    shuffle($randCats);
?>
```

Once you have randomly selected eight incorrect cats and one correct cat in the $randCats array, you can display the nine cat photos on a CAPTCHA web page. You'll use HTML styles later to ensure the photos are displayed as a 3x3 grid. In my CAPTCHA implementation, each cat photo is a separate Submit button in a web form. On the captcha/index.php CAPTCHA page, iterate through the `$randCats` array using the `foreach()` method:

```
<form method="POST">
<?php
    foreach ( $randCats as $cat ) {
            echo<<<EOF
<button type="submit" name="cat" value="$cat">
<img src="$cat" alt="image of a cat" />
</button>
EOF;
    }
?>
<input type="hidden" name="captcha" value="catCAPTCHA" />
</form>
```

The hidden input "captcha" with the value "catCAPTCHA" will be used later to detect that the user submitted the web form and not simply visited the web page.

Validating the CAPTCHA is fairly straightforward: read the POST data to the web form and search for the cat CAPTCHA value in the `$myCats` array. PHP's `in_array()` function checks whether a value is in an array, so you can use this function to simplify the lookup. At the top of the captcha/index.php page, add code to detect the posting of the cat CAPTCHA, then search for the cat CAPTCHA value:

```
<?php
 /* test if the visitor guessed the correct cat */


 if (strcmp( $_POST['captcha'], 'catCAPTCHA' ) == 0) {
  $cat = $_POST['cat'];


  if (in_array($cat, $myCats) === TRUE) {
   /* success */
  }
  else {
   /* failure */
  }
 }
?>
```

The Completed Captcha

With these pieces, it's pretty simple to complete the CAPTCHA page.

I've parameterized the cookie information via the cookie-conf.php file. The parameter file retrieves the cookie value from a separate /var/www/etc/cats/catCAPTCHA.cookie file. Also, cookie-conf.php defines a standard cookie name, path and domain, and sets the cookie expiry to one day.

This CAPTCHA page is just an example. You can improve the system by checking the value of the `redir=` URL parameter before redirecting the user there to prevent cross-site attacks. And, you might add further checks to detect multiple CAPTCHA attempts from the same user within too short a time frame.

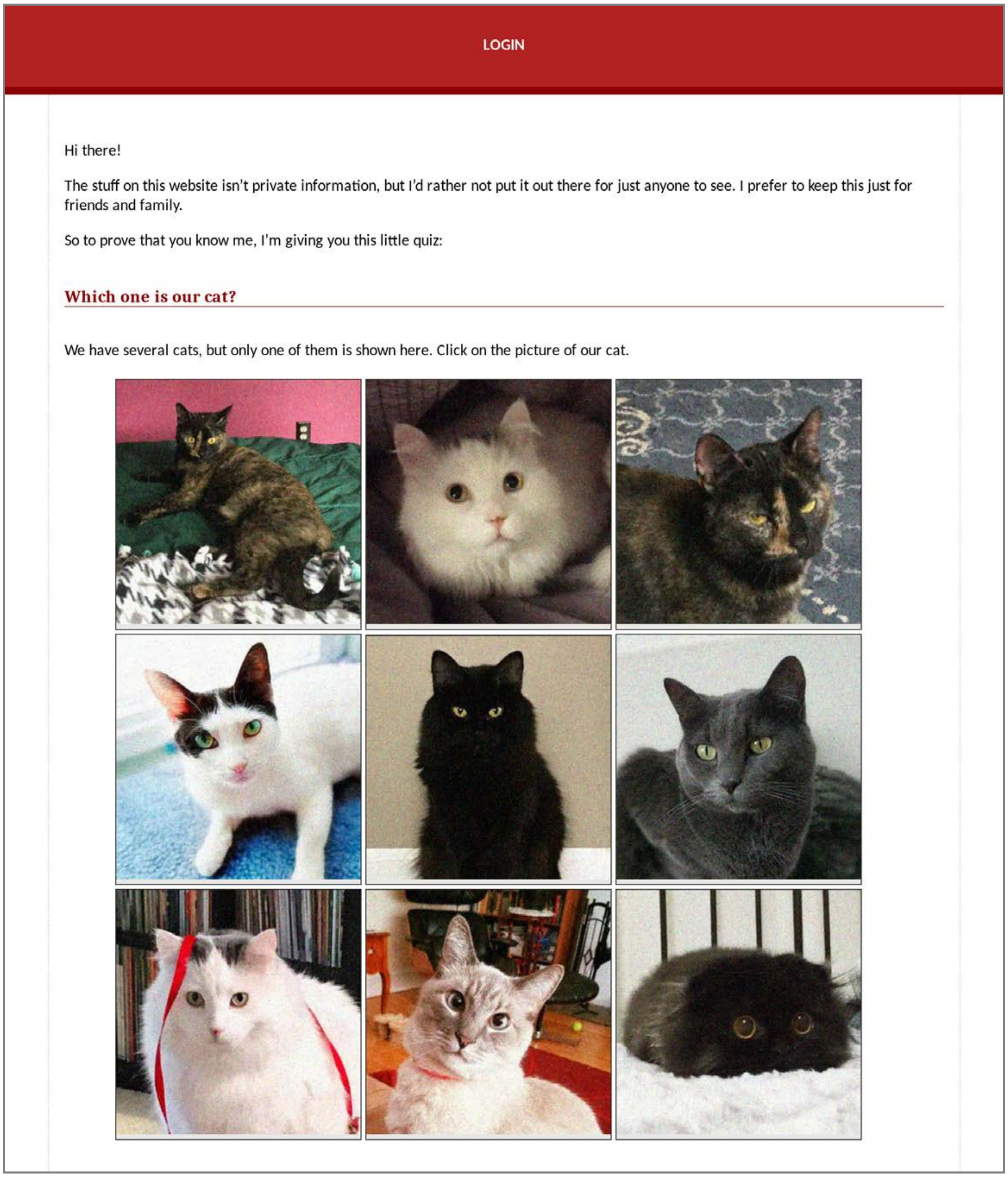
Listing 1 shows the completed CAPTCHA page.
This CAPTCHA page defines a few styles, both to make the page content

Listing 1. Completed CAPTCHA Page

```
<?php
 require ('/var/www/etc/cats/cookie-conf.php');


 /* initialize values */


 $myCats = file('/var/www/etc/cats/mycats.list',
  ➥FILE_IGNORE_NEW_LINES | FILE_SKIP_EMPTY_LINES);
 $notMyCats = file('/var/www/etc/cats/notmycats.list',
  ➥FILE_IGNORE_NEW_LINES | FILE_SKIP_EMPTY_LINES);


 $CAPTCHAsuccess = FALSE;
 $CAPTCHAmessage = '<p>Hi there!</p>';


 /* test if the visitor guessed the correct cat */


 if (strcmp( $_POST['captcha'], 'catCAPTCHA' ) == 0) {
  $cat = $_POST['cat'];


  if (in_array($cat, $myCats) === TRUE) {
   $CAPTCHAsuccess = TRUE;
   $CAPTCHAmessage = '<p class="success">Success!</p>';
   setcookie($cookieName, $cookieValue, $cookieExpire,
    ➥$cookiePath, $cookieDomain);


   $redir = $_GET['redir'];
  }
  else {
   $CAPTCHAsuccess = FALSE;
   $CAPTCHAmessage = '<p class="error">No, that is not
    ➥our cat. Try again?</p>';
  }
 }
?>
<!DOCTYPE html>
<html>
<head>
 <title>Which one is our cat?</title>
 <link rel="shortcut icon" href="/favicon.png" />
 <meta charset="UTF-8">
 <meta name="theme-color" content="firebrick">
 <meta name="viewport" content="width=device-width" />
<?php
 /* if CAPTCHA was correct, and there is a redirect URL,
    then redirect */


 if ( ($CAPTCHAsuccess) && (strlen($redir) > 0) ) {
  echo "<meta http-equiv='refresh' content='0;URL=$redir' />";
 }
?>
 <style>
body {
 background-color: white;
 color: black;
 font-family: sans-serif;
 margin: 0;
}
header {
 background-color: firebrick;
 border-bottom: .5em solid darkred;
 color: white;
}
 header h1 {
 font-size: 1em;
 margin: 0;
 padding: 2em 0;
 text-align: center;
 text-transform: uppercase;
 }
main {
 border-bottom: 1em solid gray;
}
main section {
 border-left: 1px dotted lightgray;
 border-right: 1px dotted lightgray;
 margin: 0 auto;
 padding: 2em 1em;
 max-width: 900px;
}
 main h2 {
 border-bottom: 1px solid darkred;
 color: darkred;
 font-family: serif;
 font-size: 1.1em;
 margin: 2em 0;
 }
 p.success {
 color: green;
 }
 p.error {
 color: red;
 }
 form {
 margin: 1em auto;
 max-width: 800px;
 }
 form button {
 border: 1px solid #333;
 margin: 2px;
 padding: 0;
 -moz-appearance: none;
 -webkit-appearance: none;
 }
 </style>
</head>
<body>
<header>
 <h1>Login</h1>
</header>
<main>
 <section>
<?php
 echo $CAPTCHAmessage;


 if ( ! $CAPTCHAsuccess ) {
  echo<<<EOF
<p>The stuff on this website isn't private information, but
I'd rather not put it out there for just anyone to see. I
prefer
to keep this just for friends and family.</p>


<p>So to prove that you know me, I'm giving you
this little quiz:</p>


<h2>Which one is our cat?</h2>
```

```
<p>We have several cats, but only one of them is shown here.
Click on the picture of our cat.</p>

<form method="POST">
EOF;

  $notMyCatsIdx = array_rand($notMyCats, 8);
  $randCats = array(
   $notMyCats[ $notMyCatsIdx[0] ],
   $notMyCats[ $notMyCatsIdx[1] ],
   $notMyCats[ $notMyCatsIdx[2] ],
   $notMyCats[ $notMyCatsIdx[3] ],
   $notMyCats[ $notMyCatsIdx[4] ],
   $notMyCats[ $notMyCatsIdx[5] ],
   $notMyCats[ $notMyCatsIdx[6] ],
   $notMyCats[ $notMyCatsIdx[7] ],
   $myCats[ array_rand($myCats) ]
  );
  shuffle($randCats);

  foreach ( $randCats as $cat ) {
   echo<<<EOF
<button type="submit" name="cat" value="$cat">
<img src="cache/$cat" alt="image of a cat" />
</button>
EOF;
  }

  echo<<<EOF
<input type="hidden" name="captcha" value="catCAPTCHA" />
</form>
EOF;
 }
?>
 </section>
</main>
</body>
</html>
```

look nice and to ensure that the CAPTCHA images are displayed in a 3x3 grid. Since each cat photo is 200x200 pixels, setting the page width to 700 pixels forces the CAPTCHA images to wrap to the next line after three



Hi there!

The stuff on this website isn't private information, but I'd rather not put it out there for just anyone to see. I prefer to keep this just for friends and family.

So to prove that you know me, I'm giving you this little quiz:

Which one is our cat?

We have several cats, but only one of them is shown here. Click on the picture of our cat.

Figure 3. Cats CAPTCHA

photos. This effectively presents them as a 3x3 grid.

The `-moz-appearance: none;` and `-webkit-appearance: none;` styles for the form buttons erase any extra decoration from Mozilla and Safari web browsers. Without those styles, an iPad will add rounded edges to the buttons, which would ruin the effect of the CAPTCHA.

The other styles provide a pleasant viewing experience. For example, the styles apply a red banner at the top of the page as well as different styling for text and headings.

Rotating the CAPTCHA Cookie

One final improvement remains. Remember, when the visitor correctly selects the right cat, the CAPTCHA page sets a cookie with a value read from a file. But, why read the cookie value from a file rather than set a static value?

The answer is security. If you keep the cookie value in a separate file, you can change the contents of cookie file whenever you want to "invalidate" the old cookie value. On my personal web server, I set up a daily job that writes a random value to the cookie value. Even I couldn't tell you what the new cookie value will be.

Any web cookie is really just a long string of random letters and numbers. So on my web server, I want to set a different long string every day. I do this in two steps:

1. Create a random file using `dd` and /dev/urand.

2. Generate a hash of that random file using `sha256sum`.

The value of the SHA256 checksum on the random file becomes the new CAPTCHA cookie value. That's 64 letters and numbers to "describe" a random file, which makes a pretty good CAPTCHA cookie value. I run this simple Bash script via cron to overwrite the cookie file with a random value every day:

```
#!/bin/sh
cookiefile=/var/www/etc/cats/catCAPTCHA.cookie
dd if=/dev/urandom bs=1M count=4 of=/tmp/urandom.tmp 2> /dev/null
sha256sum /tmp/urandom.tmp | awk '{print $1}' > $cookiefile
```

Testing for the CAPTCHA

Once you've put the CAPTCHA page in place, your other web pages need to recognize when a visitor has "passed" the CAPTCHA test. You easily can test for the CAPTCHA by examining cookies.

Referring back to the completed CAPTCHA page, when the visitor correctly selects the right cat, the CAPTCHA page sets a cookie with a specific value. So if the cookie exists and contains that value, any PHP page can determine if the visitor "passed" the CAPTCHA. I define a function to do this for me in the cookie-conf.php page:

```
<?php
 $cookieName = 'catCAPTCHA';
 $cookieValue = file_get_contents('/var/www/etc/cats/catCAPTCHA.cookie');
 $cookieExpire = time() + 86400; /* 86400 = (3600 * 24) = 1 day */
 $cookiePath = '/';
 $cookieDomain = 'freedos.org';


 function is_catCAPTCHA_ok()
 {
 /* shortcut to determine if a login is successful ..
    uses the cookie value */


  global $cookieName, $cookieValue;


  if ( (isset($_COOKIE)) &&
       (isset($_COOKIE[$cookieName])) &&
       (strcmp($_COOKIE[$cookieName], $cookieValue) == 0) ) {
   return TRUE;
  }
  else {
   return FALSE;
  }
 }
?>
```

In any PHP page that I want to protect with the cat CAPTCHA, I simply include the cookie-conf.php file and call `is_catCAPTCHA_ok()` to test the CAPTCHA. In a more complete implementation, you might instead redirect non-validated visitors to the CAPTCHA page, setting the `redir=` parameter so that users automatically return to the correct page. Here's a simple example to test the CAPTCHA:

```
<?php
 require ('/var/www/etc/cats/cookie-conf.php');
?>
<!DOCTYPE html>
<html>
<head>
 <title>Test page</title>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width" />
</head>
<body>


<?php
 if ( is_catCAPTCHA_ok() ) {
  echo 'cat CAPTCHA is set';
 }
 else {
  echo 'cat CAPTCHA is <b>not</b> set';
 }
?>


</body>
</html>
```

CAPTCHAs Are Easy

If you aren't protecting very sensitive information, consider a simple CAPTCHA. In my case, I wanted to keep random strangers from seeing my Christmas wish list. That's not exactly "Fort Knox" material. So a straightforward CAPTCHA does the job.

A good CAPTCHA is easy for certain people to solve and difficult for others to solve. For my web pages, I wanted to let in friends and family, and it's easy enough to do that with a CAPTCHA that uses photos of my cats.

Every time visitors load the CAPTCHA page, they see a different set of nine cats. Eight of these cats do not belong to me; only one of them is the correct cat. The correct cat is different each time and appears in a random location. The first time you load the page, my cat might be in the upper-left position. The next time, you might find my cat in the lower-right or somewhere else. This is a "good enough" test to demonstrate that you know me. Only my friends and family should know what my cats look like.

In practice, it takes me about a second to find my cat among the "forest" of other cats. And my friends and family say they like seeing the cats whenever they access my personal website. They say it's just adorable, and I agree.■

Jim Hall is an advocate for free and open-source software, best known for his work on the FreeDOS Project. Jim earned his MS in Scientific and Technical Communication from the University of Minnesota, focusing on the usability of open-source software. At work, Jim is the Chief Information Officer for Ramsey County, Minnesota.

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

RETURN TO CONTENTS

DEVELOPING CONSOLE APPLICATIONS WITH **BASH**

Bring the power of the Linux command line into your application development process.

ANDY CARLSON

PREVIOUS
Feature: Cool Project: Create a CAPTCHA for Your Website

NEXT
Guest EOF

As a novice software developer, the one thing I look for when choosing a programming language is this: is there a library that allows me to interface with the system to accomplish a task? If Python didn't have Flask, I might choose a different language to write a web application. For this same reason, I've begun to develop many, admittedly small, applications with Bash. Although Python, for example, has many modules to import and extend functionality, Bash has thousands of commands that perform a variety of features, including string manipulation, mathematic computation, encryption and database operations. In this article, I take a look at these features and how to use them easily within a Bash application.

Reusable Code Snippets

Bash provides three features that I've found particularly useful when creating reusable functions: aliases, functions and command substitution. An alias is a command-line shortcut for a long command. Here's an example:

```
alias getloadavg='cat /proc/loadavg'
```

The alias for this example is `getloadavg`. Once defined, it can be executed as any other Linux command. In this instance, `alias` will dump the contents of the /proc/loadavg file. Something to keep in mind is that this is a static command alias. No matter how many times it is executed, it always will dump the contents of the same file. If there is a need to vary the way a command is executed (by passing arguments, for instance), you can create a function. A function in Bash functions the same way as a function in any other language: arguments are evaluated, and commands within the function are executed. Here's an example function:

```
getfilecontent() {
    if [ -f $1 ]; then
        cat $1
    else
        echo "usage: getfilecontent <filename>"
    fi
}
```

This function declaration defines the function name as `getfilecontent`. The `if/else` statement checks whether the file specified as the first function argument (`$1`) exists. If it does, the contents of the file is outputted. If not, usage text is displayed. Because of the incorporation of the argument, the output of this function will vary based on the argument provided.

The final feature I want to cover is command substitution. This is a mechanism for reassigning output of a command. Because of the versatility of this feature, let's take a look at two examples. This one involves reassigning the output to a variable:

```
LOADAVG="$(cat /proc/loadavg)"
```

The syntax for command substitution is `$(command)` where "command" is the command to be executed. In this example, the `LOADAVG` variable will have the contents of the /proc/loadavg file stored in it. At this point, the variable can be evaluated, manipulated or simply echoed to the console.

Text Manipulation

If there is one feature that sets scripting on UNIX apart from other environments, it is the robust ability to process text. Although many text processing mechanisms are available when scripting in Linux, here I'm looking at `grep`, `awk`, `sed` and variable-based operations. The `grep` command allows for searching through text whether in a file or piped from another command. Here's a `grep` example:

```
alias searchdate='grep
 ➥"[0-9][0-9][0-9][0-9]-[0-9][0-9]-[0-9][0-9]"'
```

The alias created here will search through data for a date in the YYYY-MM-DD format. Like the `grep` command, text either can be provided as piped data or as a file path following the command. As the example shows, search syntax for the `grep` command includes the use of regular expressions (or regex).

When processing lines of text for the purpose of pulling out delimited fields, `awk` is the easiest tool for the job. You can use `awk` to create verbose output of the /proc/loadavg file:

```
awk '{ printf("1-minute: %s\n5-minute: %s\n15-minute:
 ➥%s\n",$1,$2,$3); }' /proc/loadavg
```

For the purpose of this example, let's examine the structure of the /proc/loadavg file. It is a single-line file, and there are typically five space-delimited fields, although this example uses only the first three fields. Much like Bash function arguments, fields in `awk` are references as variables are named by their position in the line (`$1` is the first field and so on). In this example, the first three fields are referenced as arguments to the `printf` statement. The `printf` statement will display three lines, and each line will contain a description of the data and the data itself. Note that each `%s` is substituted with the corresponding parameter to the `printf` function.

Within all of the commands available for text processing on Linux, `sed` may be considered the Swiss army knife for text processing. Like `grep`, `sed` uses regex. The specific operation I'm looking at here involves regex substitution. For an accurate comparison, let's re-create the previous `awk` example using `sed`:

```
sed 's/^\([0-9]\+\.[0-9]\+\) \([0-9]\+\.[0-9]\+\)
 ➥\([0-9]\+\.[0-9]\+\).*$/1-minute: \1\n5-minute:
 ➥\2\n15-minute: \3/g' /proc/loadavg
```

Since this is a long example, I'm going to separate this into smaller parts. As I mentioned, this example uses regex substitution, which follows this syntax: s/search/replace/g. The "s" begins the definition of the substitution statement. The "search" value defines the text pattern you want to search for, and the "replace" value defines what you want to replace the search value with. The "g" at the end is a flag that denotes global substitution within the file and is one of many flags available with the substitute statement. The search pattern in this example is:

```
^\([0-9]\+\.[0-9]\+\) \([0-9]\+\.[0-9]\+\)
 ➥\([0-9]\+\.[0-9]\+\).*$
```

The caret (^) at the beginning of the string denotes the beginning of a line of text being processed, and the dollar sign ($) at the end of the string denotes the end of a line of text. Four things are being searched for within this example. The first three items are:

```
\([0-9]\+\.[0-9]\+\)
```

This entire string is enclosed with escaped parentheses, which makes the value within available for use in the replace value. Just like the `grep` example, the `[0-9]` will match a single numeric character. When followed by an escaped plus sign, it will match one or more numeric characters. The escaped period will match a single period. When you put this whole expression together, you get an pattern for a decimal digit.

The fourth item in the search value is simply a period followed by an asterisk. The period will match any character, and the asterisk will match zero or more of whatever preceded it. The replace value of the example is:

```
1-minute: \1\n5-minute: \2\n15-minute: \3
```

This is largely composed of plain text; however, it contains four unique special items. There are newline characters that are represented by the slash-"/n". The other three items are slashes followed by a number. This number corresponds to the patterns in the search value surrounded by parentheses. Slash-1 is the first pattern in parentheses, slash-2 is the second and so on. The output of this `sed` command will be exactly the same as the `awk` command from earlier.

The final mechanism for string manipulation that I want to discuss involves using Bash variables to manipulate strings. Although this is much less powerful than traditional regex, it provides a number of ways to manipulate text. Here are a few examples using Bash variables:

```
MYTEXT="my example string"
echo "String Length:  ${#MYTEXT}"
echo "First 5 Characters: ${MYTEXT:0:5}"
echo "Remove \"example\": ${MYTEXT/ example/}"
```

The variable named `MYTEXT` is the sample string this example works with. The first `echo` command shows how to determine the length of a string variable. The second `echo` command will return the first five characters of the string. This substring syntax involves the beginning character index (in this case, zero) and the length of the substring (in this case, five). The third `echo` command removes the word "example" along with a leading space.

Mathematic Computation

Although text processing might be what makes Bash scripting great, the need to do mathematics still exists. Basic math problems can be evaluated using either `bc`, `awk` or Bash arithmetic expansion. The `bc` command has the ability to evaluate math problems via an interactive console interface and piped input. For the purpose of this article, let's look at evaluating piped data. Consider the following:

```
pow() {
    if [ -z "$1" ]; then
        echo "usage: pow <base> <exponent>"
    else
        echo "$1^$2" | bc
    fi
}
```

This example shows creating an implementation of the `pow` function from C++. The function requires two arguments. The result of the function will be the first number raised to the power of the second number. The math statement of `"$1^$2"` is piped into the `bc` command for calculation.

Although `awk` does provide the ability to do basic math calculation, the ability for `awk` to iterate through lines of text makes it especially useful for creating summary data. For instance, if you want to calculate the total size of all files within a folder, you might use something like this:

```
foldersize() {
    if [ -d $1 ]; then
```

```
        ls -alRF $1/ | grep '^-' | awk 'BEGIN {tot=0} {
        ➥tot=tot+$5 } END { print tot }'
    else
        echo "$1: folder does not exist"
    fi
    }
```

This function will do a recursive long-listing for all entries underneath the folder supplied as an argument. It then will search for all lines beginning with a dash (this will select all files). The final step is to use `awk` to iterate through the output and calculate the combined size of all files.

Here is how the `awk` statement breaks down. Before processing of the piped data begins, the `BEGIN` block sets a variable named `tot` to zero. Then for each line, the next block is executed. This block will add to `tot` the value of the fifth field in each line, which is the file size. Finally, after the piped data has been processed, the `END` block then will print the value of `tot`.

The other way to perform basic math is through arithmetic expansion. This will take a similar visual for the command substitution. Let's rewrite the previous example using arithmetic expansion:

```
pow() {
    if [ -z "$1" ]; then
        echo "usage: pow <base> <exponent>"
    else
        echo "$[$1**$2]"
    fi
}
```

The syntax for arithmetic expansion is `$[expression]`, where expression is a mathematic expression. Notice that instead of using the caret operator for exponents, this example uses a double-asterisk. Although there are differences and limitations to this method of calculation, the syntax can be more intuitive than piping data to the `bc` command.

Cryptography

The ability to perform cryptographic operations on data may be necessary depending on the needs of an application. If a string needs to be hashed, a file needs to be encrypted, or data needs to be base64-encoded, this all can be accomplished using the `openssl` command. Although `openssl` provides a large set of ciphers, hashing algorithms and other functions, I cover only a few here.

The first example shows encrypting a file using the blowfish cipher:

```
bf-enc() {
    if [ -f $1 ] && [ -n "$2" ]; then
        cat $1 | openssl enc -blowfish -pass pass:$2 > $1.enc
    else
        echo "usage: bf-enc <file> <password>"
    fi
}
```

This function requires two arguments: a file to encrypt and the password to use to encrypt it. After running, this script produces a file named the same as your original but with the file extension of "enc".

Once you have the data encrypted, you need a function to decrypt it. Here's the decryption function:

```
bf-dec() {
    if [ -f $1 ] && [ -n "$2" ]; then
        cat $1 | openssl enc -d -blowfish -pass pass:$2 >
         ➥${1%%.enc}
    else
        echo "usage: bf-dec <file> <password>"
    fi
}
```

The syntax for the decryption function is almost identical to the encryption function with the addition of "-d" to decrypt the piped data and the syntax to remove ".enc" from the end of the decrypted filename.

Another piece of functionality provided by `openssl` is the ability to create hashes. Although files may be hashed using `openssl`, I'm going to focus on hashing strings here. Let's make a function to create an MD5 hash of a string:

```
md5hash() {
    if [ -z "$1" ]; then
        echo "usage: md5hash <string>"
    else
        echo "$1" | openssl dgst -md5 | sed 's/^.*= //g'
    fi
}
```

This function will take the string argument provided to the function and generate an MD5 hash of that string. The `sed` statement at the end of the command will strip off text that `openssl` puts at the beginning of the command output, so that the only text returned by the function is the hash itself.

The way that you would validate a hash (as opposed to decrypting it) is to create a new hash and compare it to the old hash. If the hashes match, the original strings will match.

I also want to discuss the ability to create a base64-encoded string of data. One particular application that I have found this useful for is creating an HTTP basic authentication header string (this contains username:password). Here is a function that accomplishes this:

```
basicauth() {
    if [ -z "$1" ]; then
        echo "usage: basicauth <username>"
    else
        echo "$1:$(read -s -p "Enter password: " pass ;
         ➥echo $pass)" | openssl enc -base64
    fi
}
```

This function will take the user name provided as the first function argument and the password provided by user input through command substitution and use `openssl` to base64-encode the string. This string then can be added to an HTTP authorization header field.

Database Operations

An application is only as useful as the data that sits behind it. Although there are command-line tools to interact with database server software, here I focus on the SQLite file-based database. Something that can be difficult when moving an application from one computer to another is that depending on the version of SQLite, the executable may be named differently (typically either `sqlite` or `sqlite3`). Using command substitution, you can create a fool-proof way of calling `sqlite`:

```
$(ls /usr/bin/sqlite* | grep 'sqlite[0-9]*$' | head -n1)
```

This will return the full file path of the `sqlite` executable available on a system.

Consider an application that, upon first execution, creates an empty database. If this syntax is used to invoke the `sqlite` binary, the empty database always will be created using the correct version of `sqlite` on that system.

Here's an example of how to create a new database with a table for personal information:

```
$(ls /usr/bin/sqlite* | grep 'sqlite[0-9]*$' | head -n1) test.db
 ➥"CREATE TABLE people(fname text, lname text, age int)"
```

This will create a database file named test.db and will create the people table as described. This same syntax could be used to perform any SQL operations that SQLite provides, including SELECT, INSERT, DELETE, DROP and many more.

This article barely scrapes the surface of commands available to develop console applications on Linux. There are a number of great resources for learning more in-depth scripting techniques, whether in

Bash, awk, sed or any other console-based toolset. See the Resources section for links to more helpful information.■

Andy Carlson has worked in IT for the past 13 years doing networking and server administration. He is thankful to have chosen a career that he loves, grows in and learns from. He and his amazing wife have three daughters and a son, and they currently reside in Cincinnati, Ohio. He enjoys playing the guitar and spending time with family and friends.

Resources

The Advanced Bash-Scripting Guide: http://tldp.org/LDP/abs/html/index.html

GNU Awk User's Guide: https://www.gnu.org/software/gawk/manual/html_node/index.html

sed, a Stream Editor: https://www.gnu.org/software/sed/manual/html_node/index.html

My GitHub Gists: https://gists.github.com/bng44270

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

3 YEAR WARRANTY

Zero Defects

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 & enterprise1U 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

Say Hi to Subutai

A new kind of cloud growing in the open-source wild.

ALEX KARASULU

PREVIOUS
Feature: Developing Console Applications with Bash

I learned about Subutai from Philip Sheldrake of the Digital Life Collective (and much else) and thought it deserved attention here in Linux Journal, *so I offered this space for that. Alex Karasulu did most of the writing, but it was a team effort with help from Jon 'maddog' Hall, Philip Sheldrake and Steve Taylor.—Doc Searls*

What Is Subutai?

Subutai is an open-source project and platform that lets anyone share, barter or rent computer resources to create clouds from the edge rather than centralized locations (https://subutai.io). Available devices can attach to these clouds hovering on the edge. We started calling it Social Cloud Computing (https://en.wikipedia.org/wiki/Social_cloud_computing), but technically, Subutai is a dynamic p2p multi-cloud made possible thanks to Lightweight Linux Containers (https://en.wikipedia.org/wiki/LXC) and software-defined networking. Think Amazon's Virtual Private Cloud (https://aws.amazon.com/vpc), but running on your computers and the computers of social contacts who share their computer resources with you. Or, think AirBnB on computers for the people's cloud.

Subutai partners with the Digital Life Collective, a member co-operative that researches, develops, funds and supports what we call "tech we trust"—those technologies that put the individual's autonomy, privacy and dignity first, or that support those technologies that do. Our tech, not their tech.

We work together so Subutai can help prevent our privacy from being compromised and our every action from being analyzed. That means not being tied to large cloud providers and giving us the option to use resources we have on hand.

How Does It Work?

You set how much of your computers' resources you're willing to share with others. Rules and quotas are used to share with contacts from your social-media accounts. Once your network of friends, family and colleagues share with you, the stage is set to create clouds across shared computer resources.

When someone creates a cloud, peer computers authorized to share resources with the cloud's owner swarm together (like bees) to form an n-way virtual private network (VPN). A peer is a group of computers with resources that can be shared with others. A peer can be a rack of computers or a single virtual machine running on your laptop.

Peers contribute resources into the VPN as Linux container hosts. Whatever the underlying hardware, operating system or virtualization technology, resources are presented canonically to environments as containers. The VPN provides secure connectivity between these containers across the internet.

Template containers can be fired up based on Docker images to install infrastructure rapidly. Subutai has a blueprint feature for standing up complicated application stacks over several containers. It's a simple JSON-based templating format similar to Amazon's Cloud Formation templates. Use blueprints to fire up application stacks within environments, and/or build environments using your favorite devops tools for automation. Subutai blueprints support Ansible and other tools out of the box.

You basically have your own virtual data center ready to go and how you build it out is up to you. The best thing is, it's free, open source, and your mileage depends on who you know and your social-media network, not your wallet.

What's with the Name, Dude?

We think Subutai is totally bad-assed mind-explosion-inducing madness: that's why we named it after one of the greatest bad asses of all time, General Subutai (aka The Dog of War). The Apache people working on the project really like the meritocratic approach Subutai used where doers rule regardless of their background. Plus, Jon 'maddog' Hall, the project's principal advisor, and CEO of the company that originally authored Subutai, thought the name was befitting.

What's the State of the Project?

Subutai is ready and mature. We've been coding up a storm using every advantage we can gain from the latest emerging technologies to make the jaw-dropping magic happen. We are shocked and get goose bumps every time we see Subutai working (like when a jet flies over real low); we fire up our own clouds across the world on our machines because we just *can*. It's sick!

Now it's time for us to share its magic with everyone else, especially with our recent 5.0 major release. We're proud to have produced a major Subutai release almost every year for the past five years. But, we never bothered pushing hard to have others use it. Now it's time to go to the next level.

Why Did You Create Subutai?

People are increasingly conscious of compromising their privacy and getting locked in to large cloud providers, while the cost of cloud compute services can become overwhelming for the little guy. Created to address this problem, Subutai allows people to create their own resilient clouds running across computing resources they already own.

Most painful of all, when using AWS or other cloud providers, we kept getting unpleasant end-of-month surprises after forgetting to turn instances off. We wanted something simple, zero cost and independent of the cloud providers for testing and to satisfy our basic OS project infrastructure needs. We did not need premium resources. We just needed a bit more redundancy than running on a single home computer over a residential broadband line.

Beyond sharing and bartering, Subutai allows individuals and commercial entities, big or small, to rent resources in a cloud services sharing economy. By democratizing the cloud, the "race to zero" rapidly accelerates to reduce costs and increase hosting options closer to (or on) the edge. This has the additional advantage of increasing performance and efficiency by locating services and applications closer to where those services and applications are used most.

What Now?

We want to benefit the Open Source community, basically folks like us. In preparation to take on hordes of new participants, we created an open-source rewards program for FOSS projects. Any peer owner can donate computer resources to open-source projects for "goodwill". The system runs purely on goodwill, and participants create it by being good actors:

- Registering and inviting friends to register.
- Setting up peers and keeping them running above some uptime.
- Upgrading quickly when new releases come out.
- Notifying tenants of outages in advance (giving tenants time to self-evacuate).

This is the slow path to accumulating goodwill. A quicker approach is to trade peer resources for goodwill or create goodwill by donating peer resources to FOSS projects. We added GitHub integrations that

There's a lot more to Subutai than just software. We've designed a broadband modem that is also a turnkey Subutai Appliance and can be used for IoT applications.

allow GitHub projects to register with the Subutai Hub. These projects create Subutai organizations that map to their GitHub organizations and specify the resources they need. The Subutai Hub advertises their needs to peer owners that can then contribute peer resources to their favorite GitHub open-source projects. Peer owners create and earn goodwill while being recognized for their contributions to their favorite projects. If applicable, FOSS projects contribute blueprints that cloud owners can use to install the project's products. More information about the program is available on our Subutai FOSS Contribution Program page: https://subutai.io/helping-the-good-guys.html.

What's Next?

There's a lot more to Subutai than just software. We've designed a broadband modem that is also a turnkey Subutai Appliance and can be used for IoT applications. We call it the Liquid Router because we made it the Swiss army knife of IoT gateways: it has Raspberry Pi, Arduino and PMod headers. Yeah, we're f'ing crazy, but the cloud router/IoT gateway will soon do something no other broadband router can do: it will effortlessly allow average broadband users to mine for Subutai's cryptocurrency while sharing, bartering or renting computer resources. Hence, it's obviously also a physical cryptocurrency wallet.

Yes indeed, we just did what you think we did! We dropped the cryptocurrency bomb right here on *LJ*. That brings us to our cryptocurrency aspirations. We're trying to go beyond sharing and bartering to enable renting with Subutai's cryptocurrency token. The system always will operate on goodwill, but eventually will allow direct Subutai coin transactions.

Besides turning cloud computing on the edge into a real sharing economy, we're really excited about the several problems we will solve with blockchain technology. Blockchain smart contracts can be used for service-level agreements and to enforce accountability. We want verified small businesses to leverage Subutai to buy and sell resources to one another using Subutai's cryptocurrency and the blockchain. Before AWS, Amazon was just an e-commerce company. Why can't any verified company sell cloud services regardless of its vertical? We want to start a supernova to birth millions of Amazons that can transact with each other to provide amazing cloud services and applications on and off the edge.

How Can Others Get Involved?

First and foremost, we need users to create peers and environments, then give us feedback: our GitHub issue-trackers are ready. Our intelligent launchers will help you set up a VM on your desktop to share or rent resources. Power users can use scripts with other hypervisors and environments. We need projects on GitHub to register for the OS contribution program and provide blueprints for applications to be made available to all users in the Subutai Bazaar.

Next, we need developers! We need Ethereum blockchain and Solidity-savvy engineers, P2P researchers to help us experiment and optimize DHT and Gossip protocols. We need C/C++, Java, JavaScript and Golang programmers. If you want to get involved as a developer, first give Subutai a try as a user. Then take a look at our list of GitHub projects in the Resources section of this article. Come onto our Slack channel and let

ADVERTISER INDEX

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

ADVERTISER	URL	PAGE #
All Things Open	http://www.AllThingsOpen.org	27
Drupal Jobs	https://jobs.drupal.org/	45
Drupalize.me	http://drupalize.me	13
Peer 1 Hosting	http://go.peer1.com/linux	105
Silicon Mechanics	http://www.siliconmechanics.com	97
SPTech Con	http://www.sptechcon.com	37
SUSE	http://suse.com/storage	7
Vision	http://precisionagvision.com	15

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

us know if you need help diving in. We're here and waiting for you!

Register on the Subutai Hub this month using the *Linux Journal* code LJOCT17, and we'll give you some goodwill to trade cloud resources with others on the system. Happy sharing!■

Alex Karasulu is an entrepreneur with more than 25 years of experience in the software industry and a recognized leader in the Open Source community. He is widely known as the original author of the Apache Directory Server, used by IBM both as the foundation of the Rational Directory Server and also integrated into the Websphere Application Server. Alex co-founded several Apache projects, including MINA, Felix and Karaf, among others, which, along with their communities, thrive independently past his day-to-day involvement in the projects.

RESOURCES

Subutai Main Website: https://subutai.io

Slack: https://subutai-io.slack.com

GitHub Organization: https://github.com/subutai-io

GitHub Projects:

- Peer Management Console: https://github.com/subutai-io/base
- P2P Dæmon: https://github.com/subutai-io/p2p
- Tray Wallet: https://github.com/subutai-io/tray
- CLI Agent: https://github.com/subutai-io/agent
- Browser Plugins for E2E Security: https://github.com/subutai-io/browsers
- Intelligent Installers: https://github.com/subutai-io/launcher
- Snappy Snap: https://github.com/subutai-io/snap
- CDN Dæmon: https://github.com/subutai-io/gorjun
- Liquid Router: https://github.com/subutai-io/liquid-router

RETURN TO CONTENTS

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