

LINUX JOURNAL™

Since 1994: The Original Magazine of the Linux Community

SMARTPHONES,
LINUX
AND THE
INTERNET
OF THINGS

JULY 2015 | ISSUE 255 | www.linuxjournal.com

**BLUETOOTH
HACKS**
for Linux
Users

MOBILE



**HANDY
MySQL
COMMANDS**
for Admins

A LOOK AT
3D PRINTING
HARDWARE

WORKING WITH
FUNCTIONS IN
SHELL SCRIPTS

LUWRAIN
Accessibility
Solution for
the Blind

HOW-TO

MODEL
RELATIONSHIPS
IN DJANGO



WATCH:
ISSUE
OVERVIEW

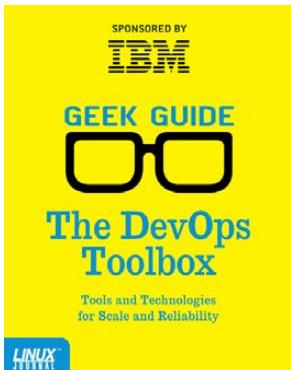


NEW!

Linux Journal eBook Series

GEEK GUIDES

The DevOps Toolbox: Tools and Technologies for Scale and Reliability



By Bill Childers

Introducing *The DevOps Toolbox: Tools and Technologies for Scale and Reliability* by Linux Journal Virtual Editor Bill Childers.

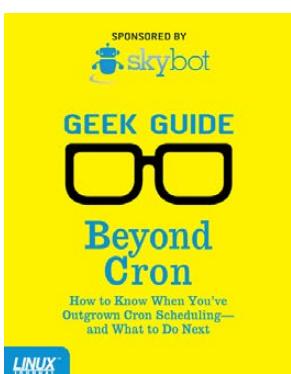
When I was growing up, my father always said, “Work smarter, not harder.” Now that I’m an adult, I’ve found that to be a core concept in my career as a DevOps engineer and manager. In order to work smarter, you’ve got to have good tools and technology in your corner doing a lot of the repetitive work, so you and your team can handle any exceptions that occur. More important, your tools need to have the ability to evolve and grow over time according to the changing needs of your business and organization.

In this eBook, I discuss a few of the most important tools in the DevOps toolbox, the benefits of using them and some examples of each tool. It’s important to not consider this a review of each tool, but rather a guide to foster thinking about what’s appropriate for your own organization’s needs.

Register today to receive your complimentary copy of The DevOps Toolbox:
<http://linuxjournal.com/devops-toolbox-guide>

Beyond Cron

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



By Mike Diehl

If you’ve spent any time around UNIX, you’ve no doubt learned to use and appreciate cron, the ubiquitous job scheduler that comes with almost every version of UNIX that exists. Cron is simple and easy to use, and most important, it just works. It sure beats having to remember to run your backups by hand, for example.

But cron does have its limits. Today’s enterprises are larger, more interdependent, and more interconnected than ever before, and cron just hasn’t kept up. These days, virtual servers can spring into existence on demand. There are accounting jobs that have to run after billing jobs have completed, but before the backups run. And, there are enterprises that connect Web servers, databases, and file servers. These enterprises may be in one server room, or they may span several data centers.

Register today to receive your complimentary copy of Beyond Cron:
<http://linuxjournal.com/beyond-cron-guide>





NOW HIRING
suse.com/jobs

GRAB HOLD
OF A DYNAMIC,
COLORFUL CAREER.

POWER
CHAMELEON THE



CONTENTS

JULY 2015
ISSUE 255

MOBILE

FEATURES

58 Bluetooth Hacks

Bluetooth technology offers many undiscovered possibilities both for practical usage and fun.

Alexander Tolstoy

70 Linux and the Internet of Things

A look at the factors that made the IoT possible.

Andrey Katsman



ON THE COVER

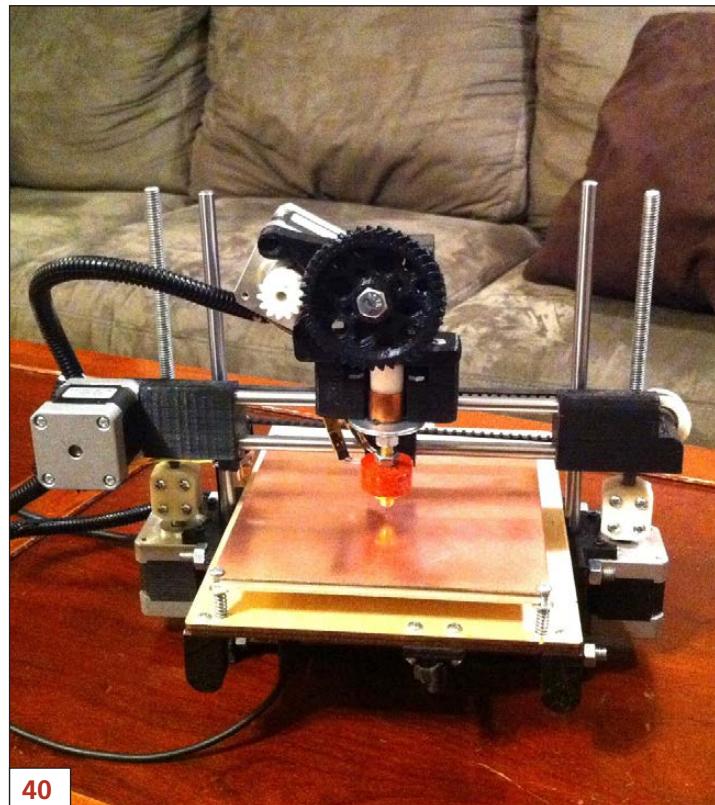
- Smartphones, Linux and the Internet of Things, p. 70
- Bluetooth Hacks for Linux, p. 58
- Luwrain: Accessibility Solution for the Blind, p. 82
- Handy MySQL Commands for Admins, p. 46
- How-To: Model Relationships in Django, p. 28
- A Look at 3D Printing Hardware, p. 40
- Working with Functions in Shell Scripts, p. 36

INDEPTH

82 Introducing Luwrain

Accessibility solutions
are changing.

Michael Pozhidaev



COLUMNS

28 Reuven M. Lerner's At the Forge

Model Relationships in Django

36 Dave Taylor's Work the Shell

Working with Functions:
Towers of Hanoi

40 Kyle Rankin's Hack and /

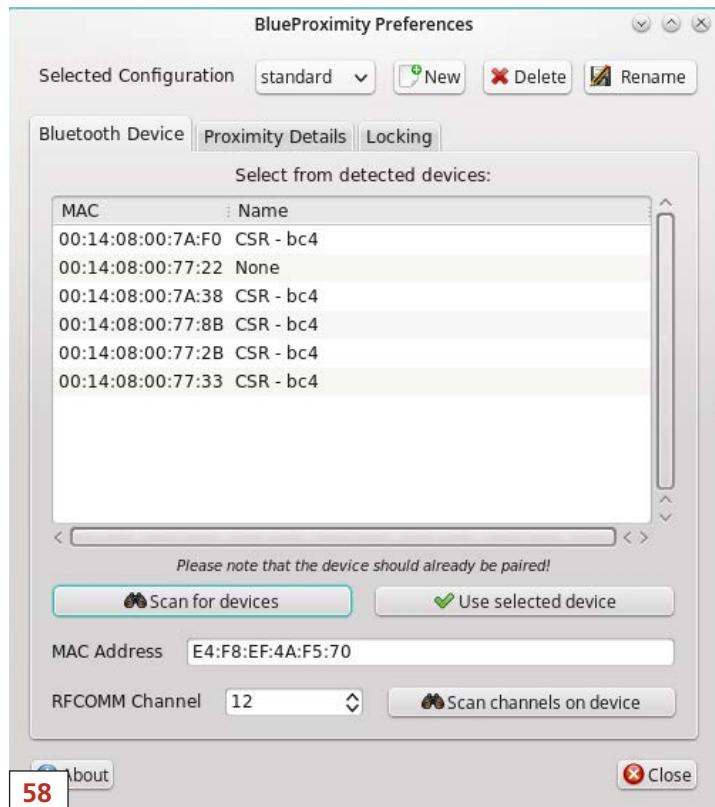
What's New in 3D Printing, Part II:
the Hardware

46 Shawn Powers' The Open-Source Classroom

MySQL—Some Handy Know-How

92 Doc Searls' EOF

Privacy Is Personal



IN EVERY ISSUE

8 Current_Issue.tar.gz

12 Letters

16 UPFRONT

26 Editors' Choice

54 New Products

95 Advertisers Index

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	Gerrick Antikajian gerrick@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.

REGISTER TODAY!

24th USENIX Security Symposium

AUGUST 12-14, 2015 • WASHINGTON, D.C.

The USENIX Security Symposium brings together researchers, practitioners, system administrators, system programmers, and others interested in the latest advances in the security of computer systems and networks. The Symposium will include a 3-day technical program with more than 65 refereed paper presentations, invited talks, posters, a panel discussion on security and privacy research ethics, and Birds-of-a-Feather sessions. Featured speakers/sessions include:

Keynote Address by Richard Danzig, *member of the Defense Policy Board, The President's Intelligence Advisory Board, and the Homeland Security Secretary's Advisory Council*

Invited Talk: "Using Formal Methods to Eliminate Exploitable Bugs" by Katherine Fisher, *Tufts University*

Invited Talk: "Machine vs. Machine: Lessons from the First Year of Cyber Grand Challenge" by Mike Walker, *DARPA*

Invited Talk: "Preventing Security Bugs through Software Design" by Christopher Kern, *Google*

The following co-located events will precede the Symposium on August 10–11, 2015:

3GSE '15: 2015 USENIX Summit on Gaming, Games, and Gamification in Security Education

CSET '15: 8th Workshop on Cyber Security Experimentation and Test

FOCI '15: 5th USENIX Workshop on Free and Open Communications on the Internet

HealthTech '15: 2015 USENIX Summit on Health Information Technologies

Safety, Security, Privacy, and Interoperability of Health Information Technologies

HotSec '15: 2015 USENIX Summit on Hot Topics in Security

JETS '15: 2015 USENIX Journal of Election Technology and Systems
(Formerly EVT/WOTE)

WOOT '15: 9th USENIX Workshop on Offensive Technologies

www.usenix.org/sec15



Stay Connected...



twitter.com/USENIXSecurity



www.usenix.org/facebook



www.usenix.org/youtube



www.usenix.org/linkedin



www.usenix.org/gplus



www.usenix.org/blog



SHAWN POWERS

Infinite Portal to the Universe, 12-Hour Battery Life

A few months back, I accidentally left my phone on the nightstand, and spent an entire day without my cell phone. It was a terrible experience. Don't get me wrong, I fully understand the concept of taking a vacation from technology. I also realize there are times when using a mobile device is inappropriate. The terrible part was that although I functioned perfectly fine, being disconnected was incredibly inconvenient. It wasn't until that day that I finally realized just how much more I can accomplish simply by being connected. As technologists, we all celebrate the awesome world of mobile computing, and this month, we devote *Linux Journal* to the concept as well.

Reuven M. Lerner and I both tackle

databases this month. Reuven describes how to handle table relationships inside Django. If you're a Python programmer, Django is an incredibly powerful Web framework, and most Web apps rely on databases for their back-end data storage. My article is far less developer-centric, as I discuss how to interact with a MySQL database directly from the command line. Programs like PHPMyadmin and Adminer are great, but it's important for a system administrator to know how the command-line interface works as well.

Dave Taylor goes back to the basics with shell scripting this month and explains how to take advantage of functions. Separating parts of code into reusable functions is a very efficient way to program in any language, and with shell scripting, it also makes the code much easier to read and understand. Instead of focusing on software, in this issue, Kyle Rankin focuses on hardware.



VIDEO:

Shawn Powers runs through the latest issue.

Specifically, he talks about the current 3D printing hardware in part two of his four-part series on 3D printing. Whether you've been fiddling with 3D printing since it became mainstream a few years ago or are just getting into the game now, Kyle's series will bring you up to date in no time.

Although many folks still think "Bluetooth" is the name of the wireless earpieces often seen on folks in power suits, we all know that it's actually just the wireless communication technology. Almost all mobile devices and computers have Bluetooth built in nowadays, and Alexander Tolstoy discusses how to take advantage of that availability. He shows a handful of hacks you can do to make use of that personal area network space with such a funny name.

Andrey Katsman takes a deep look at, well, everything. Specifically he discusses The Internet of Things, or how our entire world is beginning to join in a giant network. Andrey talks about microcontrollers, connectivity and how Linux is playing a vital role in not really taking over the world, but rather becoming what "the world" means. If you enjoy living in the future as much as I do, you don't want to miss his article.

Finally, Michael Pozhidaev dives into an important and fascinating topic. How do you best create a user interface for a computer that is effective and useful for the blind? So much of how we design

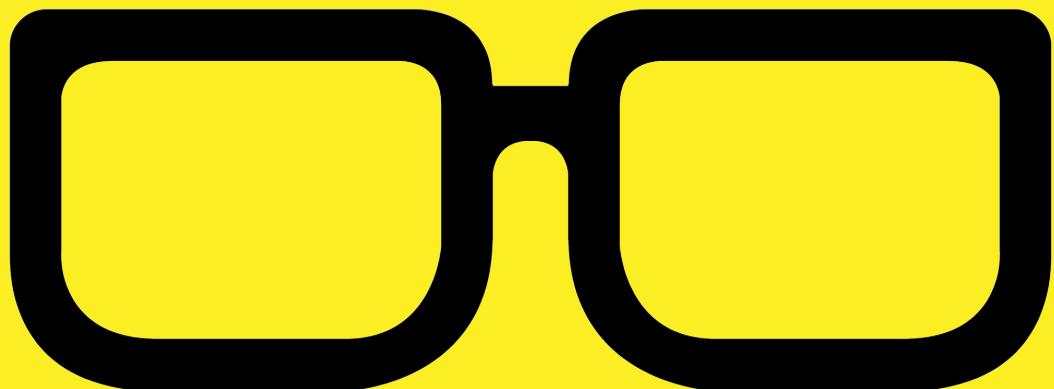
interfaces is based on visual layouts, non-visual functionality often takes a big hit. Michael introduces us to Luwrain this month, which is a Linux distribution designed to forget about eye candy and focus on usability for those who can't see the eye candy anyway. By designing interfaces that are useful and effective for sensory input and output other than visual, ideally we can come up with better user interfaces in general.

Mobile technology is a buzzword that might be starting to lose its meaning in modern culture. We're getting to the point where most technology is mobile, and soon we'll either be wearing or holding in our hands all the technology we interact with on a daily basis. As I write this on a desktop computer, however, I realize we're not quite there. This issue explores a lot of mobile-related topics, and it also contains tech tips, product announcements, and many other useful items that all continue to blur the line between mobile and non-mobile. We hope you enjoy this issue, which you probably are reading on a mobile device! (As long as you remembered to charge it, that is.) ■

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

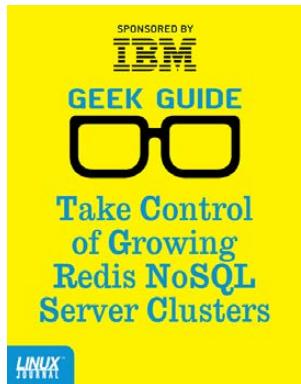
Practical books
for the most technical
people on the planet.

GEEK GUIDES



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

<http://geekguide.linuxjournal.com>



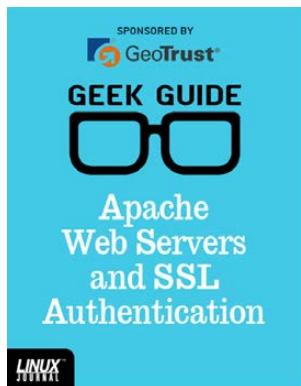
Take Control of Growing Redis NoSQL Server Clusters

Author:
Reuven M. Lerner
Sponsor: IBM



Linux in the Time of Malware

Author:
Federico Kereki
Sponsor:
Bit9 + Carbon Black



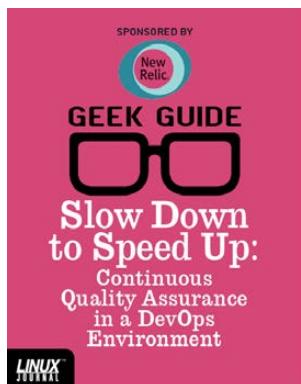
Apache Web Servers and SSL Encryption

Author:
Reuven M. Lerner
Sponsor: GeoTrust



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

Author:
Mike Diehl
Sponsor:
Servers Direct and Seagate



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

Author: Bill Childers
Sponsor: New Relic



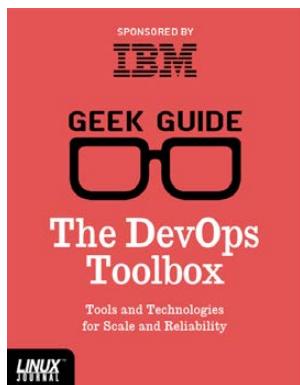
Enterprise Monitoring Success

Author:
Mike Diehl
Sponsor:
InterMapper



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

Author: Mike Diehl
Sponsor: Skybot



The DevOps Toolbox: Tools and Technologies for Scale and Reliability

Author:
Bill Childers
Sponsor: IBM

letters



Pipes and STDs

Just a comment [regarding Shawn Powers' "Pipes and STDs" article in the April 2015 issue]:

```
find / -iname '*.jpg' -exec rm {} \;
```

Will do the trick also.

—Gary Artim

Gary, you are absolutely correct. It's probably even more efficient to do this with find's exec flag. I found it surprisingly difficult to come up with an easy to understand example showing what xargs actually does. While it's not actually the best way to do the job, I hope my example at least demonstrated the concept.—Shawn Powers

An Old Trick

In the May 2015 issue of *Linux Journal* in the Letters section, Jeremy asks for help with a script that recurses directories, as he experiences problems with filenames starting with a dash.

An old trick from the command-line-only times was to prefix relative filenames with ./ to prevent unfortunate misinterpretations of entries starting with a dash. Based on that trick, I have created a modified script that should do what Jeremy set out to do:

```
#!/bin/bash

function recurse_dir()
{
    local Dir
    local DirLevel
    local OldDir

    OldDir="$PWD"
    Dir="$1"

    # Go to the wanted directory
    cd -- "$Dir"

    # Recurse through all files, including "hidden",
    # i.e. starting with a dot
    for Entry in * .*
    do
        # Skip current directory, parent directory
```

```

# and non-existing files

if [ "$Entry" != "." -a "$Entry" != ".." -a -e "$Entry" ]; then

    # Add ./ prefix if needed (i.e. starting with -)

    if [ "${Entry/#-*/-}" == "-" ]; then

        Entry="./$Entry";

    fi

    # A file -- Do whatever is needed with a file!

    if [ -f "$Entry" ]; then

        echo "File:$Entry:";

    fi

    # A directory -- Do whatever is needed with it

    # and recurse down it!

    if [ -d "$Entry" ]; then

        echo "Directory:$Entry:";

        recurse_dir "$Entry"

    fi

fi

done

# Set current directory to initial value

cd "$OldDir"

}

recurse_dir ~

```

The script uses bash-only features to avoid extra/external processes. The current version (as shown) managed to recurse through 12,000 files/directories in five seconds on an old machine.

—Torben Rybner

Hacking a Safe with Bash

I really enjoyed Adam Kosmin's article

"Hacking a Safe With Bash" in the May 2015 issue.

One notoriously difficult part of getting encryption right is all the different ways data can leak out of the encrypted area. For example, the user should be careful not to store his safe on something like Dropbox, because the sync engine may upload temporary or unencrypted files as deleted files for future restoration. Additionally, some common utilities may create temp files in /tmp when working with data (sort, for example).

The ideal is a program that works directly with encrypted data, only unencrypting in memory and never committing plain text to disk. Unfortunately, these are application-specific. Alternate paths to safe creation would be either encrypted filesystems or encrypted containers that can be mounted, such as TrueCrypt, though the user still needs to guard against plain-text leaks outside the safe.

I hope that in the coming years, encryption solutions will be more deeply woven into the operating systems and be less hackish. In the

[LETTERS]

meantime, it's good to have hacks (in the good sense of the word) such as Adam's.

—Andrew Fabbro

Photo of the Month

Thought you might like to see Leonardo getting to grips with Linux at the age of four months. Keep up the great work guys! (I've been reading *LJ* since the early days and using Linux since it came on a bunch of floppies.)



—Nick Taylor



WRITE LJ A LETTER

We love hearing from our readers. Please send us your comments and feedback via <http://www.linuxjournal.com/contact>.

PHOTO OF THE MONTH

Remember, send your Linux-related photos to ljeditor@linuxjournal.com!

LINUX JOURNAL

At Your Service

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

ACCESSING THE DIGITAL ARCHIVE:

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

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

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

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

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

O'REILLY®

OSCON

JULY 20-24, 2015
PORTLAND, OR

“OSCON is where I meet brilliant people and leverage their experiences to continuously improve both my skill sets and the sites I run.”

— PAUL PRZYBORSKI
LEAD DEVELOPER FOR NASA'S EARTH OBSERVATORY

Save 20%
on your ticket

(use code LINUXJ)



Open source software, architecture, frameworks, and tools for today's engineers.

If software is what you do, OSCON is where you want to be. Get better at what you do and rekindle your love of all things digital.



oscon.com | [@oscon](https://twitter.com/oscon) | [#oscon](#)

diff -u

WHAT'S NEW IN KERNEL DEVELOPMENT

There's a slow effort underway to allow virtually any part of the kernel to be extracted into its own shared library, thus enabling users to use any alternative subsystem they please. There's a long history of this, going back to the debate between **micro-kernels** and **monolithic kernels**. Even **Linus Torvalds**, the proponent of the monolithic kernel, believes it's better to abstract features out of the kernel, so long as it can be done without sacrificing speed, stability and other core requirements.

Most recently, **Hajime Tazaki** extracted the entire networking stack from the kernel and converted it into a shared library. This wasn't in itself part of a more generalized attempt to do such things, but while no one objected to the idea, there was considerable debate over the right way to architect the extraction, and this led to the thought that Hajime's idea could be extended to other subsystems beyond the networking stack.

Ultimately, **Richard Weinberger** suggested that the portion of Hajime's code that stubbed out the networking stack so it could be linked with a shared library could be added to the kernel's testing code and used to stub out any arbitrary portion of the kernel.

As it turned out, **Antti Kantee** had been working on a similar type of thing for **NetBSD** for the past eight years, but he cautioned that the maintainability issues could rapidly get out of hand if the design didn't aggressively address maintainability from the start. And this, he felt, would require organizing the deeper kernel infrastructure around the need to stub out portions of the kernel to be turned into dynamic libraries. But at that point, he said, the code would have only a very low maintenance cost.

Antti's recommendations were met with some suspicion. Richard felt that the maintainability issues might go deeper even than Antti cautioned, due to the tremendous

speed of Linux development relative to that of NetBSD. In the end, Richard suggested—and Hajime agreed—that the best approach would be for Hajime to maintain the code, now dubbed **libOS**, as a separate git tree himself, to get an exact measurement of how well it could keep pace with the rest of kernel development.

It's not clear whether Hajime's code ever will get into the kernel. It seems a lot of people like the ability it offers, but there are unanswered questions about how well those abilities could be sustained over time. It may take a year or more to get a better sense of these things, and until the kernel folks know more, they'll be unlikely to accept this code into the kernel.

Beata Michalska has been working on generic filesystem event notification—a kernel interface that any filesystem could use to alert the system to various events, such as being remounted read-only. Beata described four basic categories of events: warnings, errors, information and thresholds. Thresholds would include things like the amount of free space dropping below a set minimum. The kernel could respond to filesystem events

by triggering any desired response.

In response to Beata's patch, **Heinrich Schuchardt** suggested that her code should be expanded to cover a range of filesystem scenarios that it didn't yet—for example, distributed filesystems like **Lustre**, remote filesystems like **Samba**, and any **FUSE**-based filesystems also should be supported, he said. He also suggested that thumb drives and any other automounted filesystem also should trigger an event in Beata's code, and the same for filesystems mounted under virtual machines.

Various other folks also offered technical suggestions on how Beata could improve her initial patch, and Beata invited more implementation suggestions for some of the features Heinrich had requested.

One benefit everyone seemed to agree on is that a generic notification system would be highly preferable to each filesystem implementing its own notifications independently. So there was a lot of enthusiasm for Beata's work, although the exact technical details probably will continue to be hammered out for some time.

—**ZACK BROWN**

Non-Linux FOSS: Portable Apps, in the Cloud!

Your favorite apps,
    
carried on a portable or cloud drive,
    
work on PCs everywhere you go.

The concept of PortableApps has been around for a long time. It's a great way to take your Windows apps from computer to computer using a USB drive and never worry about being without your favorite program. Honestly, remembering to carry around a USB drive can be a bit of a pain though. Also, USB drives are generally fairly slow, and working directly from the drive can be cumbersome.

Even with those limitations, the PortableApps.com system is an incredible way to keep the apps you want, with your preferences,

exactly how you want them. The recent cloud changes are what sparked this mention. Although PortableApps still work great on USB drives, it's also added support for cloud-based services like Dropbox and Google Drive. If you want your programs to follow you around

from computer to computer without needing to grab a Flash drive, the new syncing functionality means having all your apps and all your preferences on any computer running Dropbox!

I'm not sure if that makes them cloud apps, or TeleportingApps, or if it's just a really great idea—regardless of what you call the concept, it means fast and efficient computing on the various Windows machines you use on a daily basis.

Download the completely free (beer and speech) installer today at <http://portableapps.com>.

—SHAWN POWERS

Take your Android development skills to the next level!

Whether you're an enterprise developer, work for a commercial software company, or are driving your own startup, if you want to build Android apps, you need to attend AnDevCon!



The Android Developer Conference

July 29-31, 2015

Sheraton Boston

- Choose from more than 75 classes and in-depth tutorials
- Meet Google and Google Development Experts
- Network with speakers and other Android developers
- Check out more than 50 third-party vendors
- Women in Android Luncheon
- Panels and keynotes
- Receptions, ice cream, prizes and more
(plus lots of coffee!)

Right after
Google IO!

Android is everywhere!
But AnDevCon is where
you should be!

Earn your Certificate!

Enhance your skills and professional qualifications as an Android expert with over 23 hours of hardcore Android training!



"There are awesome speakers that are willing to share their knowledge and advice with you."

—Kelvin De Moya, Sr. Software Developer, Intellisys

"Definitely recommend this to anyone who is interested in learning Android, even those who have worked in Android for a while can still learn a lot."

—Margaret Maynard-Reid, Android Developer, Dyne, Inc.



Register Early and Save at www.AnDevCon.com

A BZ Media Event



AnDevCon™ is a trademark of BZ Media LLC. Android™ is a trademark of Google Inc. Google's Android Robot is used under terms of the Creative Commons 3.0 Attribution License.

Android Candy: Google Photos

May 22, 2014



May 9, 2014



Apr 1, 2014



Mar 31, 2014



Google has become the company that we love and can't live without, but at the same time, I think we all worry a little about just how much Google knows about us. With that caveat, it's hard to ignore Google's newest offering: Google Photos.

Using unlimited storage, automatic sorting and face/place/event recognition, Google is taking all the manual work of tagging and hiring googlebots to do the work for us. Is that creepy? Maybe a little. Honestly though, it's hard to scoff at unlimited storage of full-quality photos and videos. Plus, if you're like me, actually finding the time to sort and tag tens of thousands of photos is something you're going to get to "any day now".

If you're not completely freaked out by Google having access to all your personal photos, give it a try today. It's free and seems to be legit. There's an Android app you can download from the Google Play Store, plus an iOS app and a Web application for using on computers. Head over to <http://photos.google.com>, and check it out!—SHAWN POWERS

They Said It

Abundance is, in large part, an attitude.

—Sue Patton Thoele

The trick is to make sure you don't die waiting for prosperity to come.

—Lee Iacocca

The power of illustrative anecdotes often lies not in how well they present reality, but in how well they reflect the core beliefs of their audience.

—David P. Mikkelsen

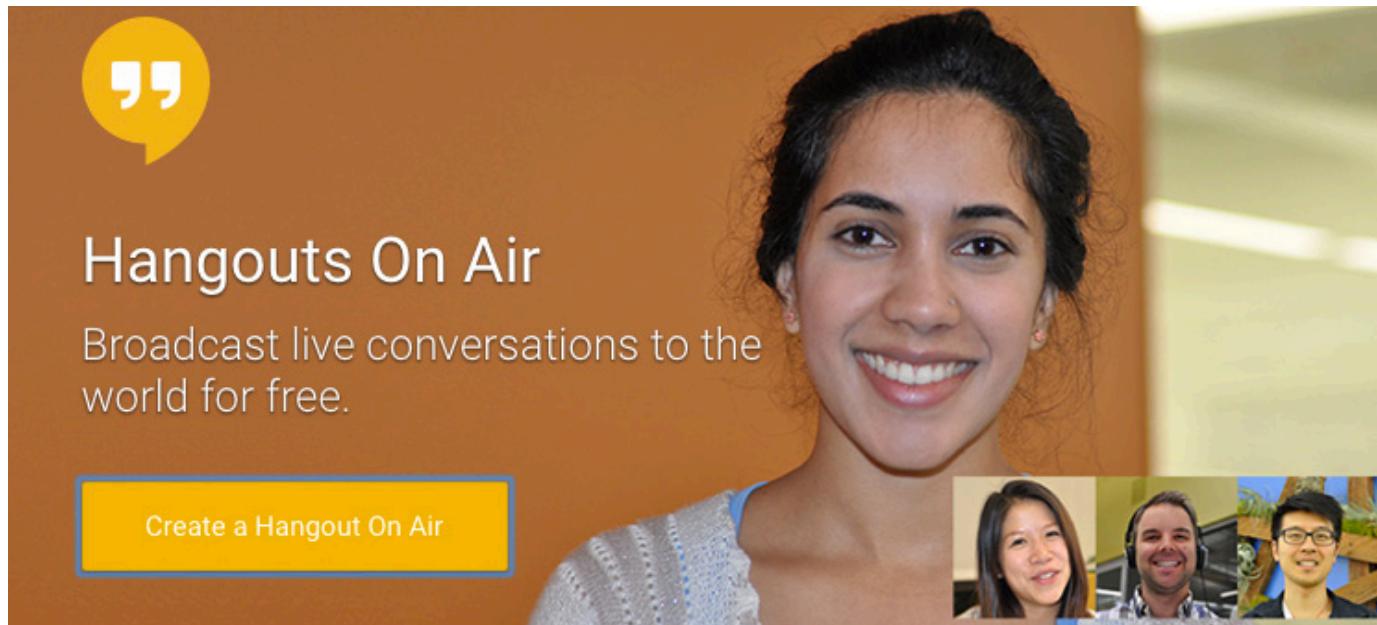
Men for the sake of getting a living forget to live.

—Margaret Fuller

There are no secrets to success. It is the result of preparation, hard work, and learning from failure.

—Colin Powell

Look Mom! I'm on the Internet!



“

Hangouts On Air

Broadcast live conversations to the world for free.

Create a Hangout On Air

(Image from Google)

Streaming video to multiple people always has been a challenge. Back when Kyle Rankin and I did “Linux Journal Live”, we’d use services like ustream or justin.tv in order to accommodate the bandwidth requirements. The problem with those services is that unless you pay significant money, the features are extremely limited. Fairly recently, Google has changed that with its “Hangouts on Air”.

The process works like any other Google Hangout video chat, except that while you’re chatting, a YouTube live stream is taking place. It’s possible to embed the live stream like any other

YouTube video, and because it uses the Google networks, there’s no need to worry about using too much bandwidth.

The process for starting, stopping and embedding Hangouts on Air still can be a bit cumbersome. Thankfully, as the product matures, its integration into YouTube gets better and better. I personally used Hangouts on Air to broadcast my kid’s drama presentation to family members who couldn’t make it. You even can keep the live stream as an archive video for folks who can’t watch the stream live. Check it out today at <http://plus.google.com/hangouts/onair>.

—SHAWN POWERS

General Relativity in Python

I have covered several different software packages for doing scientific computation in *Linux Journal*, but I haven't spent as much time describing available libraries and the kind of work that can be done with those libraries. So in this article, I take a look at SymPy, the Python module that allows you to do symbolic mathematics, and specifically at a utility named GraviPy that is built upon SymPy. With this utility, you can do all sorts of computations that you'll want to do if you're researching general relativity.

You can install both modules using pip with the following command:

```
sudo pip install sympy
sudo pip install gravipy
```

Then, you can import everything from both modules with the Python statements:

```
from sympy import *
from gravipy import *
```

This way, you will have access to all of the functionality from these modules within the namespace of your

own Python code.

When you do work in general relativity, you often need to look at very complicated equations. The GraviPy module includes a function, called `init_printing()`, that sets up everything you need for pretty-printed equations. You probably will want to call this near the top of your code so that your worksheet is more human-readable.

Let's start with the SymPy Python module. SymPy is designed to give you the ability to do symbolic mathematical computations. With it, you can do things like solve algebraic expressions, rearrange and simplify equations, and even perform symbolic derivatives and integrals. I've looked at SymPy in a previous issue of *LJ*, so here, I just focus on some of the core parts as a reminder.

The first necessary step is to define the symbolic variables that you are going to use in your calculations. Using the `symbols` command, you can define them with:

```
x,y,z = symbols('x y z')
```

This way, you define the variables that define the three space dimensions

in Cartesian coordinates.

If you want to have all four space-time coordinates defined in spherical coordinates, you can use this:

```
t, r, theta, phi = symbols('t, r, \theta, \phi')
```

Remember that you are feeding a string into the `symbols` function. This means you need to escape any special characters to get the results you are expecting.

These commands will give you the symbolic variables that you can use in expressions. But, general relativity has a special class of variables, called coordinates, that are used to define the space-time itself. There is a class, called `Coordinates`, that helps define this. Using the spherical coordinates above, you can create the coordinates with the statement:

```
x = Coordinates('\chi', [t, r, theta, phi])
```

The four space-time coordinates are stored in the object `x`. You can access the individual coordinates by using an index. In general relativity, there are two different ways of indexing variables: covariant and contravariant indexes. To look at the element values for the covariant version, you need to use positive index values. Negative index values will give you the

contravariant versions. As an example, if you wanted to get the time variable, you would use the following:

```
x(-1)
```

Right away, you should notice that this implies that GraviPy uses 1-based indexing rather than 0-based indexing.

Now that you have a set of variables to define the space-time coordinates to be used, let's move on to actual general relativity.

The core part of general relativity is the metric. The metric defines how space-time is shaped. This space-time shape is what defines the gravitational force. The usual phrase that explains what is happening is that "matter tells space-time how to bend, and space-time tells matter how to move". To define the metric within GraviPy, you need to start by creating a metric tensor object. You can do so with this statement:

```
Metric = diag(-(1-2*M/r), 1/(1-2*M/r), r**2, r**2*sin(theta)**2)
```

In this example, you'll notice a new variable, named `M`, which represents the mass of the matter that is creating this space-time distortion. If it is an item that will remain static, you don't need to do anything extra. But, there is no reason that it should remain static. If it is something that

can change, you need to use the `symbols` command to define it. In its most general form, the metric tensor is a 4-by-4 matrix of elements. The above example is of a diagonal metric where only the non-zero elements are along the diagonal. Once you have this tensor object, you can define the metric based on it with the statement:

```
g = MetricTensor('g', x, Metric)
```

This example gives the function the metric definition and the coordinate system to be used.

To access the various elements, you can use indices that follow the same format as above. For example, you could use the following:

```
g(1, 1) -> 2M/r-1
```

For both vectors and tensors, you can use a special index called `All` that will give you every possible value for the index in question. For example, you can get the entire list of coordinates with this:

```
x(-All) -> [t r theta phi]
```

You can get all of the elements of the metric tensor with the statement:

```
g(All, All)
```

Now that you have a set of coordinates and a metric tensor, there are a number of standard tensors that need to be calculated to help work out what the geometry of space-time actually looks like and how things like light beams travel through this geometry. The GraviPy module includes the tensor classes `Christoffel`, `Riemann`, `Ricci`, `Einstein` and `Geodesic`. The `Christoffel`, `Riemann` and `Ricci` tensors depend only on the metric. Therefore, they all have very similar forms to create new instances and get results out of them. For example, you can define the `Christoffel` tensor values with this statement:

```
Ga = Christoffel('Ga', g)
```

You can get individual elements with indices, just like with the metric tensor. But, some of these tensors can have a number of uninteresting zero entries. So, you can get the non-zero values with the statement:

```
Ga.components
```

This returns a dictionary where the keys are the coordinate sets for where this particular value is located, as well as the actual non-zero value at the point.

The `Einstein` tensor is the one used in the actual Einstein equations

for space-time, and they are a little different. In order to calculate them, you first need to calculate the Ricci tensor with this statement:

```
Ri = Ricci('Ri', g)
```

Once you have this tensor, you can calculate the Einstein tensor with this:

```
G = Einstein('G', Ri)
```

Before I leave off, I should look at two techniques that are absolutely necessary to doing general relativity. The first is index contraction. This is where you end up summing values over two of the indices in a tensor. In Python, you can do this by explicitly summing with:

```
Rm = Riemann('Rm', g)
ricci = sum([Rm(i, All, k, All)*g(-i, -k) for i, k in
    ↪list(variations(range(1, 5), 2, True))], zeros(4))
```

These two lines are equivalent to the above single creation of the Ricci tensor. In many cases, complicated calculations are not simplified automatically. This means that you need to do this simplification explicitly with the command:

```
ricci.simplify()
```

The other important technique is to

calculate geodesics, which are equations that define how light beams travel in this space-time. You need to create a new variable to handle the world line parameter for these equations:

```
tau = symbols('\\\tau')
```

Now, you can calculate the geodesics with this:

```
w = Geodesic('w', g, tau)
```

Again, you can use 1-based indices to access the various non-trivial equations for your space-time of interest.

With SymPy, you can do all sorts of symbolic calculations normally reserved for programs like Maple, Maxima or Mathematica. Building on these capabilities, GraviPy lets you play with space-time and do calculations to determine curvature and gravitational effects. There is not a great deal of information available on-line explaining what GraviPy can do. Your best option is to download the source files, which include a tutorial iPython notebook, even if you install GraviPy through pip. Now you can do your gravitational research all from the comfort of Python. If you are a Python fan, this module will let you do interesting research work in your favourite language.—**JOEY BERNARD**



One Port to Rule Them All!

I was chatting with Fred Richards on IRC the other day (flrichar on freenode) about sneaking around hotel firewalls. Occasionally, hotels will block things like the SSH port, hoping people don't abuse their network. Although I can respect their rationale, blocking an SSH port for a Linux user is like taking a mouse away from a Windows user! I mentioned that I used to have a remote server running SSH on port 443 so I still could get to my servers. (Port 443 is the HTTPS port, which is rarely blocked.)

I also mentioned that it was inconvenient to use port 443 for SSH, because it meant I couldn't host secure Web sites on that server. Fred graciously pointed me to `sslh`, which is an awesome little program that multiplexes (or maybe de-multiplexes?) network traffic based on the type of traffic it sees. In simple terms, it means that `sslh` will listen for incoming connections on a port like 443, and if it's a request for a Web page, it will send the request to Apache. If it's an SSH request, it sends it to the SSH daemon. It also has support for OpenVPN traffic, XMPP traffic and `tinc`.

Conceptually the program is simple, but I never considered it would be something a simple open-source application could manage! I assumed it would require a hardware appliance and lots of horsepower. I'm happy to say I was very, very wrong. In fact, it's such an impressive piece of software, it gets this month's Editors' Choice award! If you'd like to reach your SSH server over port 443 while still hosting secure Web pages, check out `sslh` at <http://www.rutschle.net/tech/sslh.shtml>. —**SHAWN POWERS**

containercon

Sheraton Seattle

August 17-19, 2015

events.linuxfoundation.org/events/containercon



Model Relationships in Django

Django provides excellent support for relationships among database tables. Here's how you can define those relationships in your projects.

In my last few articles, I've looked at Django, a powerful Python-based Web framework. I discussed how Django projects are built out of individual, reusable applications. I also covered how Django's templates provide flexibility, while avoiding the use of explicit code within the template files themselves. Then I showed how you can use Django's ORM (object-relational manager) to define your models, which then were translated into table-creation queries in SQL. Finally, I explained how you can, through your models, access the database—creating, modifying and even deleting database records using Python objects.

So in this article, I'm going to finish this exploration of Django, looking at one final part of any application that uses a relational database: how you

can define, and then use, associations among your Django models.

Relational databases have been around for several decades, and despite the hype surrounding NoSQL databases of various sorts, one of their great strengths is the ease with which you can combine information from different sources, mix and match them together, and then produce a result. Perhaps this isn't always true; in particular, relational databases often work poorly with hierarchical data. But the idea that you can read from several tables, combine them in a variety of ways, and get precisely the answer you were looking for, is not only powerful, but also addictive once you get used to it.

Thus, if you're using Django along with a relational database, you're likely

going to want to use those relations with your Django models. However, it's not always obvious how you can do this. Here, I take a look at the three different types of relationships you can create among models (and thus SQL tables) and see how Django supports those relationships.

One-to-One Relationships

The first and easiest to understand relationship between models is the one-to-one relationship. This means each model has one and only one corresponding model in another class. (At the database level, it means every row has one and only one row in another table.)

For example, let's say I'm in charge of a bus company, in which each bus is assigned to a specific driver. For every driver, there is a bus. And, for every bus, there is a driver. The relationship in this case is one-to-one.

I created a small Django project ("buscompany") to demonstrate this and an application ("schedule") inside that project:

```
$ django-admin.py startproject buscompany
$ cd buscompany
$ python manage.py startapp schedule
```

In order for my Django project to recognize and work with my new

application, I need to add it to the application's settings. This means going into the application's `settings.py` and adding the following line to the definition of `INSTALLED_APPS`:

```
'schedule'
```

Once an application is installed into a Django project, the project will recognize the application's model and other definitions.

Inside the "schedule" application, I then create two models: one for buses, and the other for drivers. Both of these are defined in `schedule/models.py`:

```
class Driver(models.Model):
    first_name = models.TextField()
    last_name = models.TextField()
    birthdate = models.DateField()

    def __str__(self):
        return "{} {}".format(self.first_name,
                             self.last_name)

class Bus(models.Model):
    registration_number = models.TextField()
    license_plate = models.TextField()
    purchased_on = models.DateField()
    driver = models.OneToOneField(Driver)

    def __str__(self):
        return "Vehicle license {}".format(self.license_plate)
```

Once the migrations have been run, the database will contain the models I want. But more important, an instance of Bus now can be linked to an instance of Driver.

As you can see, I have defined a Driver class, and a Bus class. In particular, note the “driver” attribute defined in Bus, and that it is defined to be an instance of `models.OneToOneField`. I indicate that this attribute should be linked to the Driver class by passing it as a parameter to `models.OneToOneField`.

Once I have created these basic model definitions, I can put them into practice by creating and then running the migrations:

```
$ python manage.py makemigrations  
$ python manage.py migrate
```

Once the migrations have been run, the database will contain the models I want. But more important, an instance of Bus now can be linked to an instance of Driver. For example, next I create a driver using the Django management shell:

```
$ python manage.py shell  
>>> from schedule.models import Bus, Driver
```

```
>>> d = Driver(birthdate='1980-01-01', first_name='Mr.',  
>>> last_name='Driver')  
>>> d.save()  
>>> d.id      # Returns 1
```

Now I can use this new driver’s ID when creating a new Bus object:

```
>>> b = Bus(driver_id=1, license_plate='abc123',  
           purchased_on='2015-01-01', registration_number=123)  
>>> b.save()
```

With the above in place, I can get the driver via the bus, and vice versa:

```
>>> str(b.driver)  
'Mr. Driver'  
  
>>> str(d.bus)  
'Vehicle license abc123'
```

And of course, if I want to retrieve specific fields from an associated object, I can do that:

```
>>> d.bus.registration_number  
u'123'
```

With this relationship in place, I now can produce a table of drivers and buses that the bus company's management can use to see the current allocation of drivers and buses.

I want to add two quick points to the above design. First and foremost, a one-to-one relationship is inherently double-sided. That is, both models effectively see a one-to-one relationship. You can, in Django, define the relationship on whichever object you wish. The results will be the same; the appropriate attributes will be created in both directions.

The second thing to notice, and remember, is that Django assumes, unless you explicitly say otherwise, that database columns are NOT NULL. This means if I try to create a bus without an associated driver:

```
b2 = Bus(license_plate='abc456',
          purchased_on='2015-02-02', registration_number=456)
b2.save()
```

Django will throw an `IntegrityError` exception, because I have not specified anything for the `driver_id` column. When creating your models, you will have to decide whether you want to have the added flexibility of NOT NULL columns—which, in this case, might indicate that a bus has not (yet) been allocated to a driver—or the added

integrity of NOT NULL columns.

One-to-Many Relationships

The above works well when you have a simple, one-to-one relationship. However, life is a bit more complex than that. You often encounter situations where you need a one-to-many relationship. For example, let's assume my bus company wants to make the most of its buses. Thus, there isn't just one driver associated with a bus, but rather a number of possible drivers associated with a bus.

As things currently stand, that's not possible; a given bus can have only a single driver at a time. In order to change this, and to allow multiple drivers to drive the same bus (although not simultaneously, one would hope), I need to change the association. Instead of a one-to-one relationship between buses and drivers, there now will be a one-to-many relationship.

In SQL, you would accomplish this by having two separate tables: one for drivers and one for buses. Because one bus could have multiple drivers, I would put a "bus" foreign key in the drivers table. In this way, each driver would point to a single bus, but a bus could be affiliated with multiple drivers.

When one table references another in the database world, it's called using

a “foreign key”. And so it shouldn’t come as a surprise that in Django, I need to modify my models, removing the `OneToOneField` from `Bus` and adding a `ForeignKey` field in `Driver`. The models then will look like this:

```
from django.db import models

class Bus(models.Model):
    registration_number = models.TextField()
    license_plate = models.TextField()
    purchased_on = models.DateField()

    def __str__(self):
        return "Vehicle license {}".format(self.license_plate)

class Driver(models.Model):
    first_name = models.TextField()
    last_name = models.TextField()
    birthdate = models.DateField()
    bus = models.ForeignKey(Bus)

    def __str__(self):
        return "{} {}".format(self.first_name,
                              self.last_name)
```

Notice that I flipped the order of the models definitions, so that `Bus` would be recognized as a known class name to Python when I define the `Driver` class.

As always in Django, I then tell the system to create a migration:

```
$ python manage.py makemigrations
```

But in this particular case, because I already have added some records to my database, Django refuses to continue. It warns me that I’m asking to create a new field (“`bus`”) on the `Driver` model, but that because there already are records, and because the field is defined as “not null” (the default in Django), the migration cannot work. Django thus gives me the choice between providing a default for existing records or for quitting from the migrations and trying again later. I decided to choose a default and then entered 1, the ID of the bus in my system.

Once that was done, Django finished making the migrations. I ran them with:

```
$ python manage.py migrate
```

```
$ python manage.py shell
```

```
>>> from schedule.models import Bus, Driver
>>> d = Driver.objects.first()
>>> d.bus
<Bus: Vehicle license abc123>
```

So far, so good—my driver has a bus. Let’s add another driver for the same bus:

```
>>> d2 = Driver(birthdate='1980-02-02', first_name='Ms.',
```

```

last_name='AlsoDriver', bus_id=1)

>>> d2.save()

>>> d2.bus

<Bus: Vehicle license abc123>

```

Now I have two drivers who work with the same bus. Does the bus know this? Let's find out:

```

>>> b = Bus.objects.first()
>>> b.driver_set.all()
[<Driver: Mr. Driver>, <Driver: Ms. AlsoDriver>]

```

Sure enough, I find that when I retrieve the `driver_set` from the bus and then ask Django to provide me with the objects themselves, I get the two drivers.

Many-to-Many Relationships

The above is fine if each driver uses only a single bus. But it's more likely that each driver will use a number of different buses, and that each bus will have a number of different drivers. In such a case, this existing model will not allow me to describe the data; instead of a one-to-many relationship, I need a many-to-many relationship. This is a common thing in the data-modeling world; consider the relationship between authors and books, or ingredients and recipes, or classes in school, and you'll see that a many-to-many

relationship is often the best (and only) way to model things correctly.

In a relational database, there's no way for two tables to have a many-to-many relationship directly. Rather, you have to create a "join table", a third table in which the relations are listed. Django understands this and allows you to define a many-to-many relationship by creating a `ManyToManyField` on one of the models. Django then creates the join table for you. Let's modify my `Driver` model, so that they'll have this relationship:

```

class Driver(models.Model):
    first_name = models.TextField()
    last_name = models.TextField()
    birthdate = models.DateField()
    bus = models.ManyToManyField(Bus)

    def __str__(self):
        return "{} {}".format(self.first_name,
                             self.last_name)

```

Notice that I put the `ManyToManyField` only on one of the classes and not both of them. I then create and run my migrations:

```
$ python manage.py makemigrations
$ python manage.py migrate
```

Next, I re-enter the Django shell and

re-configure the associations:

```
>>> d = Driver.objects.get(id=2)
>>> d.bus.add(1)
>>> d.save()
>>> d.bus.add(2)
>>> d.save()

>>> d.bus.all()
[<Bus: Vehicle license abc123>,
 →<Bus: Vehicle license xyz987>]
```

After executing the above, I similarly added a second driver to each of the buses. Do I see the results of this on the other side? Let's find out:

```
>>> b = Bus.objects.get(id=1)
>>> b.driver_set.all()
[<Driver: Mr. Driver>, <Driver: Ms. AlsoDriver>]
```

Indeed, you can see that associating a bus and driver can be done from either side. The fact that a join table exists is transparent; Django creates and maintains that table for you, part of its general approach of helping you

concentrate on the important aspects of your projects, rather than the nuts and bolts that happen behind the scenes.

This concludes my tour of many of Django's major features, which started a few months ago. If you're a Python developer, and if you are interested in creating complex Web applications, there's no doubt that Django is the way to go. If you're looking to do something simpler, I'd suggest Flask—but only you know how complex your system will be. There's no doubt that if you have an ambitious Web application that will use a database, Django is up to the task and is worth serious consideration.■

Reuven M. Lerner is a Web developer, consultant and trainer. He recently completed his PhD in Learning Sciences from Northwestern University. You can read his blog, Twitter feed and newsletter at <http://lerner.co.il>. Reuven lives with his wife and three children in Modi'in, Israel.

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

Resources

The main site for Django is <http://DjangoProject.com>, and it includes a great deal of excellent documentation, including a tutorial.

Information about Python, in which Django is implemented, is at <http://python.org>.

WEBINAR

Comprehensive Identity Management and Audit for Red Hat Enterprise Linux

Thursday, July 16 at 10am PDT

Centrally managing Windows users, group policy and entitlements through Active Directory is a blessing for Windows IT, but leaves RHEL IT out in the cold. Native tools only go so far. Join Centrify and a guest speaker from the National Weather Service to explore the benefits of using Centrify Server Suite to centralize RHEL identities within Active Directory and granularly control privileges, while auditing all privileged activity.

Speakers:

Tony Goulding

Centrify



Jeff Williams

National Weather Service



Wade Tongen

Centrify



REGISTER NOW: <http://linuxjournal.com/identity-webinar>



DAVE TAYLOR

Working with Functions: Towers of Hanoi

Dave takes a look at how to work with functions in shell scripts, offering both an iterative and recursive solution to the classic Towers of Hanoi problem.

For this article, I thought it would be beneficial to go back to some basics of shell scripting and look at how functions work. Most script writers probably eschew using functions because it's a bit antithetical to how scripts tend to evolve, as a sequence of commands on the command line that are captured in a file.

As with general programming, however, if you have a block of even a few lines that are invoked multiple times, in multiple locations in a script, turning that into a function makes a lot more sense.

The general syntax is thus:

```
MyFunction() {  
    commands to execute  
}
```

Within the main script—or any other functions, or the function itself—a function is invoked by simply referencing its name:

```
echo "before call to MyFunction"  
MyFunction  
echo "after call to MyFunction"
```

Easy enough. Positional parameters are given to the function in a manner exactly analogous to how command flags are handed to the script overall, as \$1, \$2, \$3 and so on.

This means that within a function, I could write:

```
if [ -n "$1" ] ; then  
    echo "I was given $1 as my first parameter"  
fi
```

The biggest limitation with shell functions is that they can return an integer value only of 0–127, where a typical script actually utilizes the 0 = false or failure, 1 = true or success.

Well, that's a bit lazy of me, testing \$# would be a better way to ascertain if positional parameters were handed to the function, so let's rewrite that as:

```
if [ $# -gt 0 ] ; then
    echo "I was given $# parameters"
fi
```

The biggest limitation with shell functions is that they can return an integer value only of 0–127, where a typical script actually utilizes the 0 = false or failure, 1 = true or success. Or, a lot of scripters just ignore return values entirely and use a global variable to pass back values, particularly if they're string values, which otherwise are impossible to deal with in a function.

Functions manipulating global variables is a bit sloppy compared to best practices in Ruby, Java or C++, but you've got to work with what you've got, right?

Towers of Hanoi

To make this column more interesting, I'm going to brush off a classic recursion program and see how to make it work

as a shell script. The program is called Towers of Hanoi, and you've probably seen a kid's toy for this. It's a set of different sized disks and three pegs, with the goal being to move all the disks from one peg to another while never violating the rule that bigger disks should never be atop smaller ones. If the pegs are labeled 0, 1 and 2, the simplest case of one disk is to simply move it from peg 0 to peg 2. For two, the first (smaller) disk moves from 0 to 1; the second disk moves from 0 to 2, and the first then moves atop it from 1 to 2.

There's an iterative solution that's succinct:

```
#!/bin/sh
/bin/echo -n "Towers of Hanoi. How many disks? "
read disk
for (( x=1; x < (1 << $disk ); x++ )) ; do
    i=$((($x & $x - 1 ) % 3))
    j=$(((($x | $x - 1 ) + 1 ) % 3))
    echo "Move from tower $i to tower $j"
done
```

When run, this delightfully short script produces the result I desire, a

step-by-step solution to the Towers of Hanoi problem:

```
Towers of Hanoi. How many disks? 4
Move from tower 0 to tower 2
Move from tower 0 to tower 1
Move from tower 2 to tower 1
Move from tower 0 to tower 2
Move from tower 1 to tower 0
Move from tower 1 to tower 2
Move from tower 0 to tower 2
Move from tower 0 to tower 1
Move from tower 2 to tower 1
Move from tower 2 to tower 0
Move from tower 1 to tower 0
Move from tower 2 to tower 1
Move from tower 0 to tower 2
Move from tower 0 to tower 1
Move from tower 2 to tower 1
```

It turns out that the solution to n disks is $2^{**n} + 1$ steps mathematically. Put succinctly, 20 disks would take a staggering 1,048,577 moves. That's a lot more patience than I would have with a puzzle game; I don't know about you.

Recursion in Shell Script Functions

The point of introducing the Towers of Hanoi puzzle, however, was to demonstrate a neat recursion trick within a shell script, so let's have a look at how that might work too.

The basic algorithm has three steps:

1. Move the topmost disks from Source to Temp.
2. Move the next largest disk from Source to Destination.
3. Move the topmost disks from Temp to Destination.

There are various Web sites with illustrations of how this works, but it just might be easier to present my script so you can see what I'm talking about:

```
moves=0      # start with no moves
hanoi()
{
    if [ $1 -gt 0 ] ; then
        hanoi "$((\$1-1))" $2 $4 $3
        echo move $2 "-->" $3
        moves=$(( $moves + 1 ))
        hanoi "$((\$1-1))" $4 $3 $2
    fi
}
/bin/echo -n "Towers of Hanoi. How many disks? "
read disks
hanoi $disks 1 2 3
echo "It took $moves moves to solve Towers for $disks disks."
```

Notice that there are four parameters that you must give to the recursive Towers of Hanoi function and that you have to "prime the pump" with the initial invocation of:

```
hanoi $disks 1 3 2
```

The parameters, in order, are the number of disks and the identity of each of the three pegs you'll be using. For four disks and three pegs, the solution:

Towers of Hanoi. How many disks? 4

```
move 1 --> 3
move 1 --> 2
move 3 --> 2
move 1 --> 3
move 2 --> 1
move 2 --> 3
move 1 --> 3
move 1 --> 2
move 3 --> 2
move 3 --> 1
move 2 --> 1
move 3 --> 2
move 1 --> 3
move 1 --> 2
move 3 --> 2
```

It took 15 moves to solve Towers of Hanoi for four disks. Look closely at the code, and you'll realize you actually can use mnemonic names for the pegs by changing the initial invocation near the bottom of the script, which makes the output considerably more understandable as a solution, this time for just three disks:

Towers of Hanoi. How many disks? 3

```
move source --> temp
move source --> destination
move temp --> destination
move source --> temp
move destination --> source
move destination --> temp
move source --> temp
It took 7 moves to solve Towers of Hanoi for 3 disks.
```

Although you might not encounter situations where you need to create recursive functions in a shell script, the more general function creation and usage definitely can make your scripts more efficient and easier to understand. And, as for Towers of Hanoi? Well, do you have a better algorithm? It's a staple of computer science education, so I bet a lot of you have tackled this one in the past.

Credit to Kamaraj Subramanian for his iterative Hanoi script and phoxis for his recursive Hanoi script—they proved to be good starting points for my own creations this month.■

Dave Taylor has been hacking shell scripts for more than 30 years—really. He's the author of the popular *Wicked Cool Shell Scripts* (10th anniversary update coming very soon from O'Reilly and NoStarch Press) and can be found on Twitter as @DaveTaylor and more generally at his tech site <http://www.AskDaveTaylor.com>.

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



KYLE RANKIN

What's New in 3D Printing, Part II: the Hardware

Find out what's changed in 3D printers during the past three years.

This is the second article in what will be a four-part series on the current state of 3D printing compared to how things were three years ago when I wrote my first series on 3D printing. Of course, this is *Linux Journal*, so the focus will be on Linux and open-source-specific aspects in 3D printing. I won't dwell much on proprietary products. In my last article, I gave a general overview on the state of 3D printing; in this one, I focus on the hardware side.

If you were to compare 3D printers three years ago to today, probably the first thing you would notice is just how polished and consumer-focused the overall look of the machines are now. Three years ago, most printers were based off the RepRap line of 3D

printers. They had a hobbyist look, with 3D-printed gears and other parts combined with nuts and bolts you could get from the hardware store. Those printers that didn't consist of a series of threaded and smooth rods for their structure were made from laser-cut wood. The focus was much more on community and sharing designs freely to improve the quality of the printers as rapidly as possibly while still using parts easily purchased from a hardware store or on-line. Many of the commercial 3D printer offerings at the time also were some form of a RepRap printer sold pre-assembled and calibrated with some refinements and improvements, plus support from the company if anything went wrong.

Fast-forward to today, and the



Figure 1. The Original Ultimaker

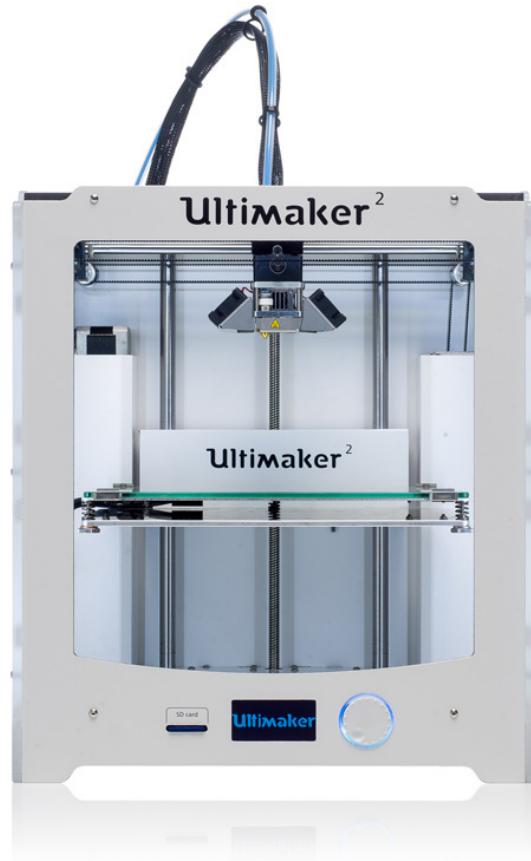


Figure 2. The Current Ultimaker 2

selection of 3D printers is widely different. As the focus has shifted from the hobbyist to the consumer, you see modern designs that hide away the electronics and wiring, forgo laser-cut wood for painted metal and acrylic cases, and look more like something you'd put on a desk in your office than a work bench in the garage. It reminds me a lot of the early attempts to polish the Linux desktop to appeal more to the end user, and both changes have received a similar backlash from the original community.

One of the best examples of the difference between old and new is a comparison between the original Ultimaker and the current Ultimaker 2. The original has the classic boxy laser-cut wood case, while the modern printer has a frosted acrylic case.

So, what caused this shift in focus? A number of factors are involved, but I'd argue it was the success of the original Printrbot Kickstarter campaign, itself a RepRap design modified to be more affordable and aimed at schools and the end consumer, that attracted

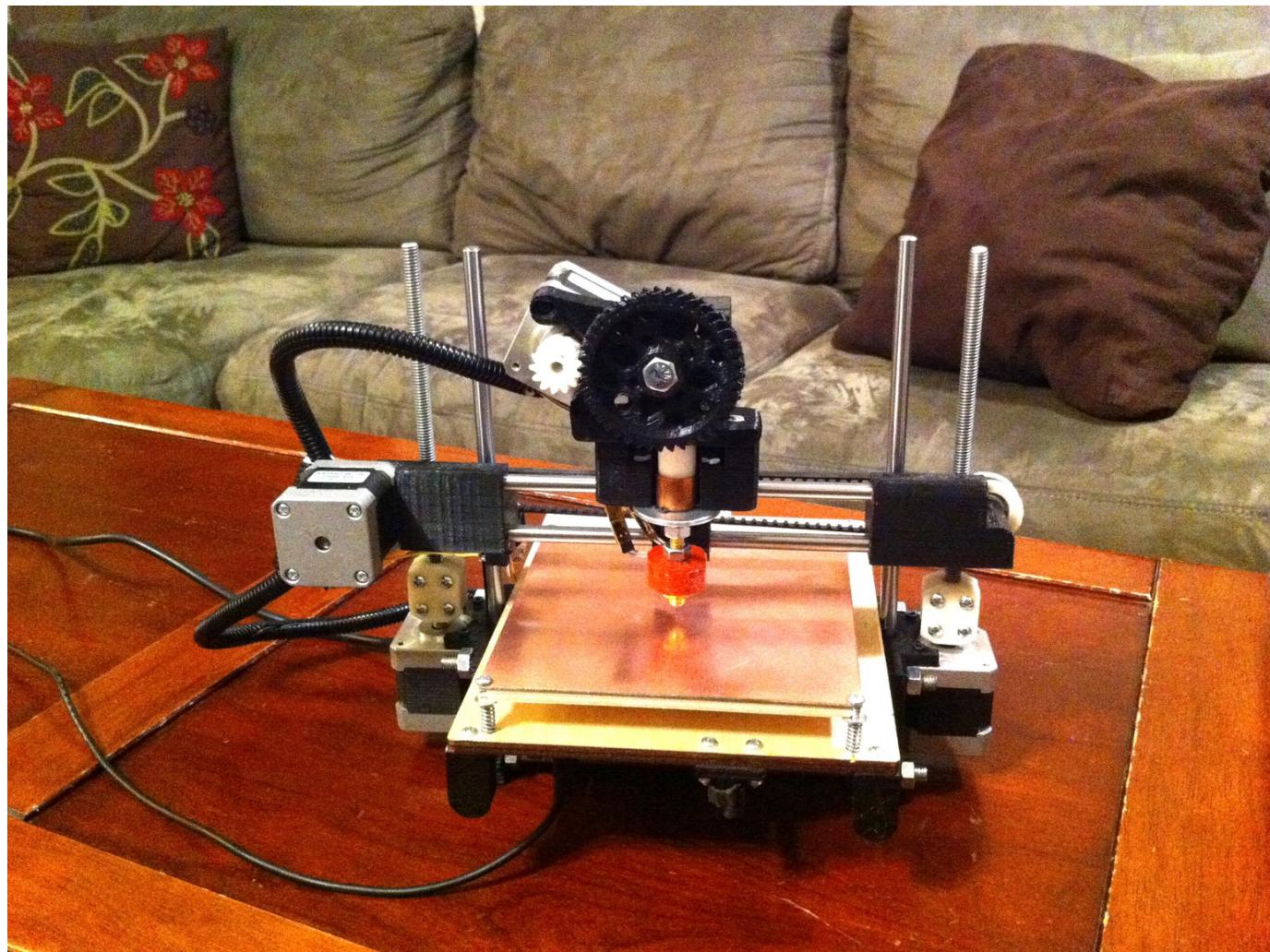


Figure 3. The Original Printrbot

many entrepreneurs to say “I can do that” and start their own 3D printer companies. In a world of \$2,000 Makerbots or \$800 RepRap kits, a \$500 3D printer grabbed the attention of a whole new group of users (including yours truly) who couldn’t quite justify \$800 for a kit that wasn’t guaranteed to work after many hours of assembly.

You can see the changes in the Printrbot line itself. The original printer

was much like a RepRap with many 3D-printed parts and a wooden print bed with an optional heated bed. The immediate follow-up to that printer traded some of those parts for laser-cut wood to speed up manufacturing. Compare that to the modern Printrbot line, which are all shipped assembled, all made with painted metal frames and machined metal print beds, and still are around the same price as the original assembled Printrbots.



Figure 4. The Current Printrbot Simple Metal

One unfortunate trend in modern 3D printers is the absence of open design files. Years ago, most of the major 3D printers fully shared all of the hardware design files; thus, if you were so inclined, you could build your very own 3D printer with your own self-sourced parts. Makerbot's announcement that it no longer would share the design files for its Replicator line created quite a bit of outrage in the community, but despite that, a number of other 3D printing companies followed suit. That said, there still are some notable 3D printing companies that continue the open-source ethos and share their design files, such as Ultimaker,

Lulzbot and Printrbot, all offering their hardware design files either along with the introduction of a new product or within a few months.

Extruders

Extruder design is another area that has seen a lot of innovation during the past three years. The extruder is the part of the 3D printer that feeds plastic filament into a hot end where it is melted and printed out. Three years ago, many printers adopted some variation of the Wade's extruder design, which used a stepper motor connected to a small gear that powered a larger gear connected to a bolt with teeth that would grip onto the filament. The concept of multiple extruders on a single printer was still in the early prototype phase, and most extruders used 3mm filament.

These days, just about every printer has its own custom extruder design and hot end. Many extruders have shifted to a direct drive system where the stepper motor has a gear directly attached to it that feeds filament into the hot end, and 1.75mm filament is starting to become the norm. Many of the most popular printers now offer dual extruders as an upgrade option. With dual extruders, a printer can feed two different colors of filament or even two different filament types

into the same print. This not only allows you to print multiple colors at the same time, but it also allows you to print support material out of a water-soluble filament instead of having to snap or cut it off of the print when it's finished. In the past, many hot ends were ceramic and used a teflon tube inside to feed the filament from the top of the hot end into the heated area before it was extruded. With the new interest in more exotic filaments, some of which require higher temperatures than PLA or ABS require, all-metal hot ends are starting to become popular as they can be heated past the limits imposed by a teflon tube.

Calibration Improvements

Since most 3D printers three years ago were geared toward hobbyists who likely built the printer as a kit, a large amount of calibration and tinkering were assumed to be part of the fun. Everything from calibrating potentiometers for your stepper motors to the levelness of your print bed to the height your Z axis above the print bed required a screwdriver, feeler gauges, calipers, locking bolts, and trial and error. Often moving the printer from one location to another would throw off your calibration, so you had to budget a whole new

round of adjustments.

With the focus shifting from hobbyists to consumers, a lot of attention has shifted to making calibration simpler. First, the majority of printers sold today are assembled and calibrated before they show up at your door. Second, some printer models, such as the Printrbot line, include a magnetic sensor that can sense when it is close to a metal print bed. This sensor then can be used with an automatic calibration routine to measure the distance from the print bed along three corners and then level the print bed in software. Instead of adjusting a screw above a Z-axis endstop to change the Z-axis height above the bed, the Printrbot line takes advantage of the metallic sensor to get close to the print bed and allows you to enter a number in the printer's settings to move the Z axis higher or lower by a fraction of a millimeter. Once this value is set, you no longer have to calibrate your Z axis unless you tinker with or remove the hot end. Although these days the stepper motor currents already were calibrated when the printer was being assembled, if you find you do need to adjust them, some printer boards even allow you to adjust that in software instead of with a screwdriver.

New Materials

One of the most exciting areas of innovation in 3D printing today is in the materials you can use. Three years ago, your material choices basically were limited to PLA and ABS plastic. Today, in addition to a much greater variety of colors for PLA and ABS (including glow-in-the-dark), there are a number of more exotic plastics, such as PVA (water-soluble), and a number of different nylon filaments that provide different strength and flexibility profiles. There also are new flexible filaments like Ninjaflex, Filaflex and Ninjaflex Semiflex that let you print with a material that acts much more like rubber than plastic and can be used for sandals, toy car tires or cell-phone cases.

Most recently, a number of exotic PLA material blends have appeared where traditional PLA filament is blended with tiny fragments of other materials including wood, bamboo, carbon fiber, iron, copper, brass and bronze. Each of these materials more or less extrudes like PLA, but the finished product has properties of the blended material. For instance, wood and bamboo filaments look quite a bit like wood grain with each layer and can be stained with wood stains. The metallic filaments each can be polished and overall given post-processing treatments like metal,

and they can be polished with steel wool or in a rock tumbler. For instance, the copperfill filament I have can be polished, and the finished product even smells like a penny. Iron filament even can react with magnets. Unfortunately, since PLA still is used to bind the metallic dust together, products made with these materials don't have the strength of the pure metal, so the applications are more appropriate for jewelry or busts and less for metal bars or gears. If you need strength, it's still better to look into one of the other plastics like ABS or Nylon.

As you can see, 3D printing is a rapidly moving technology, and a lot has changed in the past three years. I know this article has veered a bit away from the traditional Linux-based content you normally see in this space, so next month, I'll make up for it with a look into the innovations in open-source 3D printing software that interfaces with this hardware. ■

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



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



SHAWN POWERS

MySQL – Some Handy Know-How

Learn some simple MySQL commands that will make your life as a Linux admin much easier.

I recently was talking to someone over IRC who was helping me with a PHP app that was giving me trouble. The extremely helpful individual asked me to let him know the value of a certain field in a record on my MySQL server. I embarrassingly admitted that I'd have to install something like PHPMyAdmin or Adminer in order to find that information. He was very gracious and sent me a simple one-liner I could run on the command line to get the information he needed. I was very thankful, but admittedly embarrassed. I figured if I don't know how to get simple information from a MySQL server, there probably are others in the same boat. So, let's learn a little SQL together.

Get a Database

It turns out there are quite a few

sample databases to download from the Internet. Unfortunately, they're all far more complicated than I'd like to use for demonstration purposes. So, I created a database. Although you don't have to have my database in order to follow along, it certainly will help if you do. So first, let's create a database and import my data.

The first thing you need to do is install MySQL. Depending on your distribution, this either will be an `apt-get` command, a `yum` command, or a search in the GUI software center. I'll leave the installation to you—feel free to use Google if you're struggling. The main thing is to remember the root password you set during the installation process. This isn't the same as the root password for your system; rather it's the root user in your MySQL server. If you're

using a live server, just create a new user/password with access to create databases. I'm going to assume you've just installed MySQL, and you know the root user's password.

When you work with MySQL on the command line, you use the "mysql" application. So in order to create the database for this example, type:

```
mysql -u root -p -e "CREATE DATABASE food"
```

You should be prompted for a password, which is the password you set during installation for the MySQL root user account. If you get an error about the database already existing, you can choose a new name for your database. Just realize that the name you pick will be what you'll use later when I refer to the "food" database.

Next, you need to get my data into your database. I have an SQL file stored at <http://snar.co/foodsqli>. You can download that file, or use wget on the command line to get it. If you use wget, the resulting filename might be "foodsqli" or "food.sql", depending on how your version of wget works. Either filename will work, just make note of what you have so you can change the command you're going to use below. To download and

import the data, type:

```
wget http://snar.co/foodsqli  
mysql -u root -p food < ./food.sql
```

Remember, if your downloaded file is "foodsqli" instead of "food.sql", you can just change the command to ./foodsqli instead of ./food.sql. Both will work.

What Did You Just Do?

The mysql program can work either interactively or as a one-liner like above. The first command created a database on your MySQL server named "food", which you'll be using to follow along with this article. The -u flag allows you to connect as a specific user—root in this case. Typing -p tells mysql to ask you for a password. You also could have typed the password on the command line like this:

```
mysql -u root -pmypassword -e "CREATE DATABASE food"
```

However, that bothers me for two reasons. One, the password is displayed clearly on the screen, which just creeps me out. But also, you probably noticed there's no space between the -p and the actual password. That wasn't a typo; that's how you actually must do it—weird. I usually just have it prompt me for

the password. The last part of the command tells mysql to execute a command. I'll cover using commands interactively in a bit, but here, you told it to create a database called "food", and then exit. The CREATE and DATABASE don't have to be all caps, but it's standard practice in the SQL world. If the word is a command or a special word, it's all uppercase. If it's a piece of data or name, it's lowercase. Again, it's just a convention, but I'll try to stick to it. You should too, as it makes reading SQL stuff much easier.

The second section of code did two things. It downloaded my sample database using wget, and then it dumped that SQL data into the "food" database. The downloaded file is just a text file. You can look at it, and you'll see a bunch of SQL statements (with capitalized commands). Those commands were piped in via STDIN and executed much like the -e command you used to create the initial database. The end result is that you have a database called "food" on your local MySQL server, and it contains my tables and data. Let's go check out the data.

Connect to the Database

If you've been following along, you've actually connected to MySQL and the

database already, but you created single commands that executed and ended. To enter interactive mode, you simply type:

```
mysql -u root -p food
```

This will prompt you for your password and then log you in to the interactive mode of mysql, with the "food" database open for you to explore. You should get a prompt that looks something like this:

```
Welcome to the MySQL monitor. Commands end with ; or \g.
```

```
Copyright (c) 2000, 2014, Oracle and/or its affiliates.  
All rights reserved.
```

```
Oracle is a registered trademark of Oracle Corporation and/or  
its affiliates. Other names may be trademarks of their  
respective owners.
```

```
Type 'help;' or '\h' for help. Type '\c' to clear the  
current input statement.
```

```
mysql>
```

To check whether you're logged in to the "food" database, and that the SQL import worked, let's look at the tables in the database. Type the following:

```
SHOW TABLES;
```

You should see this:

```
mysql> SHOW TABLES;
+-----+
| Tables_in_food |
+-----+
| fruit          |
| vegetable      |
+-----+
2 rows in set (0.00 sec)
```

Note that every command you'll enter must end with a semicolon. If you forget the semicolon, it just will go to the next line and expect you to type more commands. If you do that (I do it about half the time), just type a semicolon alone on the next line, and it will execute just as if you didn't forget the semicolon.

You should see the fruit and vegetable tables in the food database. If you don't, go back through the first steps, as something must have gone wrong. Read any error messages closely.

The next thing you'll do is look at the data in each table. To do that, you'll use a new command, `SELECT`, which in interactive mode just shows the data you're "selecting" based on whatever criteria you specify. So, type:

```
SELECT * FROM fruit;
```

In mysql, the asterisk is a wild card. So the `SELECT` command is showing everything "FROM" the "fruit" table. You then should see a visual display of the entire table's worth of data:

```
mysql> SELECT * FROM fruit;
+-----+-----+-----+
| name | size | color |
+-----+-----+-----+
| lemon | small | yellow |
| grape | small | purple |
| apple | small | red |
| banana | small | yellow |
| watermelon | big | green |
+-----+-----+-----+
5 rows in set (0.01 sec)
```

Try doing the same thing with the vegetable database. You should see a similar set of data, but with vegetable information instead of fruit.

Filter the Produce!

Usually when you're manipulating a database full of data, you want to work only on a subset of the data. Officially that's called a "query", but don't let the database jargon scare you off. You're just going to use almost-English commands to filter results to meet your needs.

Let's say you want to see a list of

only small vegetables. If you type:

```
SELECT * FROM vegetable WHERE size = "small";
```

you should see:

```
mysql> SELECT * FROM vegetable WHERE size = "small";
+-----+-----+-----+
| name | size | color |
+-----+-----+-----+
| pea  | small | green |
| radish | small | red   |
| bean  | small | green |
| corn  | small | yellow |
+-----+-----+-----+
4 rows in set (0.01 sec)
```

You'll notice "pumpkin" isn't listed, because its size is "big" instead of "small".

You also can just show the name of the vegetable that matches your query instead of showing all the fields. So if you type:

```
SELECT name FROM vegetable WHERE size = "big";
```

you simply should see:

```
mysql> SELECT name FROM vegetable WHERE size = "big";
+-----+
| name |
+-----+
| pumpkin |
+-----+
1 row in set (0.00 sec)
```

This has the interesting result of showing the name of all the large vegetables without actually showing the size data. It obviously requires you to know a little about how the database is structured (so you knew the "size" information was there), but you can set filters based on data you don't actually display.

Changing Data

Up until this point, you've looked only at existing data. The interactive mysql program also allows you to modify and add data in the database. To make a change, you use the UPDATE command. So if you want to change corn from a small vegetable to a big vegetable, you'd type:

```
UPDATE vegetable SET size = "big" WHERE name = "corn";
```

You should end up with corn that is now big instead of small, and so running the same command you ran earlier:

```
SELECT name FROM vegetable WHERE size = "big";
```

should result in something like this:

```
mysql> SELECT name FROM vegetable WHERE size = "big";
+-----+
| name   |
+-----+
| pumpkin |
| corn   |
+-----+
2 rows in set (0.00 sec)
```

You can do more than modify existing data, however; you also can add new data. It's a little more complicated than updating an existing value, but it's still fairly clear. Let's say you want to add honeydew to your database. To add a row to a table, type:

```
INSERT INTO fruit (name, color, size)
VALUES ('honeydew', 'green', 'big');
```

And then if you `SELECT` everything from the fruit table, you should see this:

```
mysql> SELECT * FROM fruit;
+-----+-----+-----+
| name      | size   | color  |
+-----+-----+-----+
| lemon     | small  | yellow |
| grape     | small  | purple |
| apple     | small  | red    |
| banana    | small  | yellow |
| watermelon| big    | green  |
| honeydew  | big    | green  |
+-----+-----+-----+
6 rows in set (0.00 sec)
```

Just the Tip of the Iceberg!

The mysql command-line interactive program is a very powerful way to access, display and even manipulate data without the need for any GUI at all. The basic commands are well worth learning, so if you're in a situation like I was, you can pull some MySQL data without installing a GUI tool to do it. That said, the GUI tools are great, and they can be used as a way to learn the command-line stuff. For example, Figure 1 shows Adminer (<http://www.adminer.org>) looking at the database. Searching, filtering, sorting and countless other SQL functions are easily accessible via drop-down menus. The really cool part is that Adminer shows you the exact query it used to get the results. So if you want to use a GUI tool to learn command-line options, Adminer is a great way to do so.

This silly little database of fruits and vegetables is obviously far more simple than the sorts of databases you'll be troubleshooting for Web applications. Thankfully, the concepts are exactly the same whether you're searching through thousands of financial transactions or a handful of produce items. It's also important to realize that

Select: vegetable

Select data Show structure Alter table New item

Select Search Sort

color != green
(anywhere) =

Sort color descending
descending

Limit Text length Action

50 100 Select !

SELECT * FROM `vegetable` WHERE `color` != 'green' ORDER BY `color` LIMIT 50 (0.003 s) Edit

<input type="checkbox"/> Modify	name	size	color
<input type="checkbox"/> edit	pumpkin	big	orange
<input type="checkbox"/> edit	radish	small	red
<input type="checkbox"/> edit	corn	big	yellow

(3 rows) whole result

Modify Selected (0) Export (3) Import

Save Edit Clone Delete

Figure 1. Adminer is an incredibly powerful tool, plus it teaches you what it's doing!

just SELECT-ing records in a live database won't alter any data, so you don't have to worry about ruining things just by looking. In fact, it's really good practice to try building complex queries on existing databases just to see if you can do it correctly. If you get stumped, just fire up Adminer and see what you did wrong. Good luck, and happy databasing! ■

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

|||||||
Send comments or feedback via
[**http://www.linuxjournal.com/contact**](http://www.linuxjournal.com/contact)
or to ljeditor@linuxjournal.com.

LINUX JOURNAL

on your
Android device

Download the
app now on the
**Google Play
Store**



www.linuxjournal.com/android

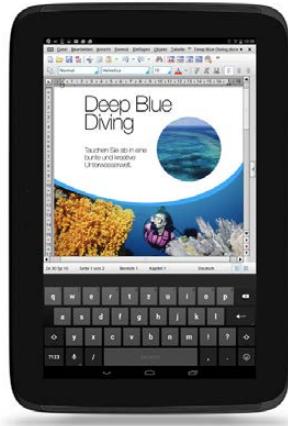
For more information about advertising opportunities within *Linux Journal* iPhone, iPad and Android apps, contact John Grogan at +1-713-344-1956 x2 or ads@linuxjournal.com.

Canonical Ltd.'s Ubuntu for Cloud & Servers

The feature-rich 22nd release of the Ubuntu Debian Linux-based operating system, version 15.04, recently became available for download from Canonical Ltd., the OS's commercial sponsor.

In the Cloud & Server versions of this "Vivid Vervet" release, three new features stand out: the LXD hypervisor, Snappy Ubuntu Core and the OpenStack Kilo release. First, the next-generation LXD hypervisor for containers, asserts Canonical, eliminates the virtualization penalty of traditional hypervisors, making Linux-on-Linux workloads much faster and much more dense. Meanwhile, Snappy Ubuntu Core, the smallest Ubuntu available, is designed for lightweight cloud container hosts running Docker and for smart devices. Snappy Ubuntu Core already is running on the next generation of network switches, home routers, smart drones and robots, adds Canonical. Finally, the company notes that this Ubuntu release is "the world's first OpenStack distribution to make the newest 'Kilo' release available to users, a significant step forward in scalability for virtual networks on OpenStack".

<http://www.ubuntu.com>

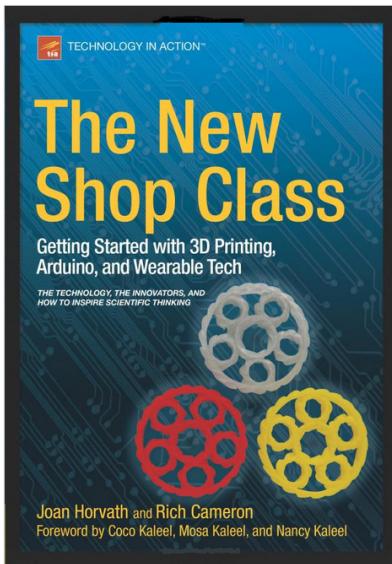


SoftMaker Office HD Basic

Although it lacks the advanced functionality of its sibling version, the newer, free and slightly more spartan SoftMaker Office HD Basic "still considerably overtrumps all the other free Android Office packages", claims its developer. SoftMaker adds that the original SoftMaker Office HD was the first office suite for Android tablets to offer the full functional scope of a Windows Office package; the "Basic" variant meanwhile forgoes some advanced functionality. Basic consists of three apps, namely TextMaker HD

Basic (word processing), PlanMaker HD Basic (spreadsheets) and Presentations HD Basic (presentation software), each optimized for touch operation and seamlessly compatible with their MS Office counterparts. The programs offer functionality beyond simple correspondence to include advanced formatting and graphical design. Connectivity with Dropbox, Google Drive, Evernote and OneDrive is built in.

<http://softmaker.com>



Rich Cameron and Joan Horvath's *The New Shop Class* (Apress)

Publisher Apress proposes an ideal use for its new book *The New Shop Class* as a hands-on guide for mentoring budding makers and hackers to a potential science career. Subtitled *Getting Started with 3D Printing, Arduino, and Wearable Tech* and authored by real “rocket scientist” Joan Horvath and 3D printing expert Rich Cameron, the work demonstrates what open-source “maker” technologies and wearable tech really can do in the right hands. Not only will this book keep readers a step ahead

of the young makers in their lives, but it also will provide essential background on mentoring future scientists and engineers. Insights into how engineers and scientists got their start, what these professionals do and how their mindsets mirror that of the maker also are included.

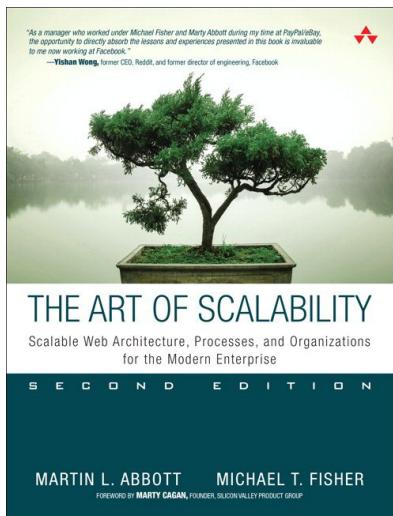
<http://www.apress.com>

AdaCore's CodePeer

AdaCore's CodePeer 3.0 is the recently upgraded version of the company's advanced static analysis tool for the automated review and validation of Ada source code. Version 3.0 sports a variety of enhancements that help developers detect potential runtime and logic errors early in the software life cycle, and its deep analysis can directly support formal certification against industry-specific safety standards. These standards include DO-178B for avionics and EN50128, the highest standard for safety integrity concerning software for railway control and protection. AdaCore claims a proven track record in the most demanding systems and can help customers in any application domain. The tool, AdaCore says, simplifies the verification effort by detecting subtle bugs in both new code that is being developed and in existing code bases that need to be analyzed for vulnerabilities.

<http://www.adacore.com>





Martin L. Abbott and Michael T. Fisher's *The Art of Scalability, 2nd Ed.* (Addison-Wesley Professional)

In their new book *The Art of Scalability*, 2nd ed., leading scalability consultants and authors Martin L. Abbott and Michael T. Fisher comprehensively explain how to scale products and services smoothly for any requirement. The contents of this extensively revised edition include new technologies, strategies and lessons, as well as new case studies from the authors' consulting work for companies like eBay, Visa, Salesforce.com and Apple. Written for both technical and nontechnical decision-makers, Abbott and Fisher explore the core elements affecting scalability, including architecture, process, people, organization and technology.

<http://informit.com>



VMware's Project Lightwave and Project Photon

As part of its push to enable enterprise adoption of cloud-native applications, VMware recently announced two new open source-projects: Project Lightwave and Project Photon. Project Lightwave is an identity- and access-management project that VMware says will extend enterprise scale and security to cloud-native applications. Meanwhile, Project Photon is a lightweight Linux operating system optimized for cloud-native applications. These projects are designed to help enterprise developers securely build, deploy and manage cloud-native applications.

By integrating into VMware's unified platform for the hybrid cloud, these projects will create a consistent environment across the private and public cloud to support cloud-native and traditional applications. The company hopes that the open-sourcing of these initiatives will foment a dynamic ecosystem of partners and developer community, leading to common standards, security and interoperability within the cloud-native application market. The desired outcome is improved technology and greater customer choice.

<http://www.vmware.com>

Imagination Technologies' Creator CI20 Microcomputer

Looking for a high-performance, fully featured Linux and Android development platform? "Step right up" says Imagination Technologies, developer of the new MIPS-based Creator CI20 microcomputer. The company recently upgraded the Creator CI20, which now features an improved board layout that optimizes Wi-Fi performance and is easier to mount in cases. Imagination Technologies also has designed a new 3D-printable enclosing, and makers can use source files from the company's Web site to build their own versions. On the software side, the new version adds out-of-the-box support for FlowCloud, an IoT API designed to connect devices to the cloud.

<http://imgtec.com>



Sentrix's Cloud-DMZ



The traditional response to the growing threat of attack on Web applications has been to deploy Web Application Firewalls (WAFs). WAFs, says Sentrix, are no longer suitable as they are extremely complex to deploy, configure and manage within today's dynamic Web development models. The Sentrix alternative is Cloud-DMZ version 3.0, "a disruptive approach" that increases security effectiveness by replicating the customer's original Web site, thus locking out attackers by removing more than 99% of the attack surface. Unlike WAFs, Cloud-DMZ does not inspect all incoming traffic but rather continuously analyzes the protected Web property's functionality and virtualizes the customer user interface within a cloud-based grid. As a result, Cloud-DMZ prevents data breaches, defacement and DDoS attacks while not generating false positives and forcing the blocking of legitimate traffic. Among other features, version 3.0 unveils a new, real-time Application Layer Visualizer, "the first tool that empowers both security and Web development teams to visually identify the critical business logic components of their Web application, identify security gaps, fix them using a visual policy editor, and test them before publishing", notes Sentrix.

<http://www.sentrix.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.

BLUETOOTH HACKS

CONNECT TO THE INTERNET,
WORK WITH YOUR FILES,
LOCK YOUR WORKSPACE,
LISTEN TO MUSIC
AND DO SO MUCH MORE
WITH THE HELP OF
BLUETOOTH TECHNOLOGY.

ALEXANDER TOLSTOY

In an effort to bring more comfort and security to listening to music via headphones while cycling, Swedish telecom vendor Ericsson invented Bluetooth—a wireless connectivity technology, which dates back to 1994. Since then, the standards behind Bluetooth have greatly evolved, but it still remains a niche technology, not widely used outside traditional business applications.

How can you benefit from using Linux on a Bluetooth-equipped PC? There are many use cases, some of which are unconventional and even mind-blowing.

Friendly Recon

Bluetooth in Linux is powered by the BlueZ software stack. It includes basic tools for remote device discovery and setup. Using it, you can collect some helpful information about the devices and people around you.

To start, let's see what devices are reachable for local discovery:

`hcitool scan`

You'll get a list of devices with their BD_ADDR values (similar to MAC) and possibly text descriptions. These descriptions are optional, and some users advisedly leave them blank. But still there's a way to find some

information on such devices. You can find the class code of a device if you run the `hcitool inq` command. After that, you can analyze it with the Bclassify utility (<http://github.com/mikeryan/btclassify>), which accepts class code as an input parameter and gives back class disentanglement. Here is the example 0x5a020c code:

```
atolstoy@linux:~/Downloads/btclassify-master> ./btclassify.py  
→0x5a020c  
0x5a020c: Phone (Smartphone): Telephony, Object Transfer,  
→Capturing, Networking
```

Then you can find what services are supported by the device. Here is the sample output:

```
atolstoy@linux:~> sdptool browse 11:22:33:44:55:66 |  
→grep Service\ Name  
Service Name: Headset Gateway  
Service Name: Handsfree Gateway  
Service Name: Sim Access Server  
Service Name: AV Remote Control Target  
Service Name: Advanced Audio  
Service Name: Android Network Access Point  
Service Name: Android Network User  
Service Name: OBEX Phonebook Access Server  
Service Name: SMS Message Access  
Service Name: OBEX Object Push
```

Note that `sdptool` works even when the device is not discoverable, but is somewhere nearby. Using

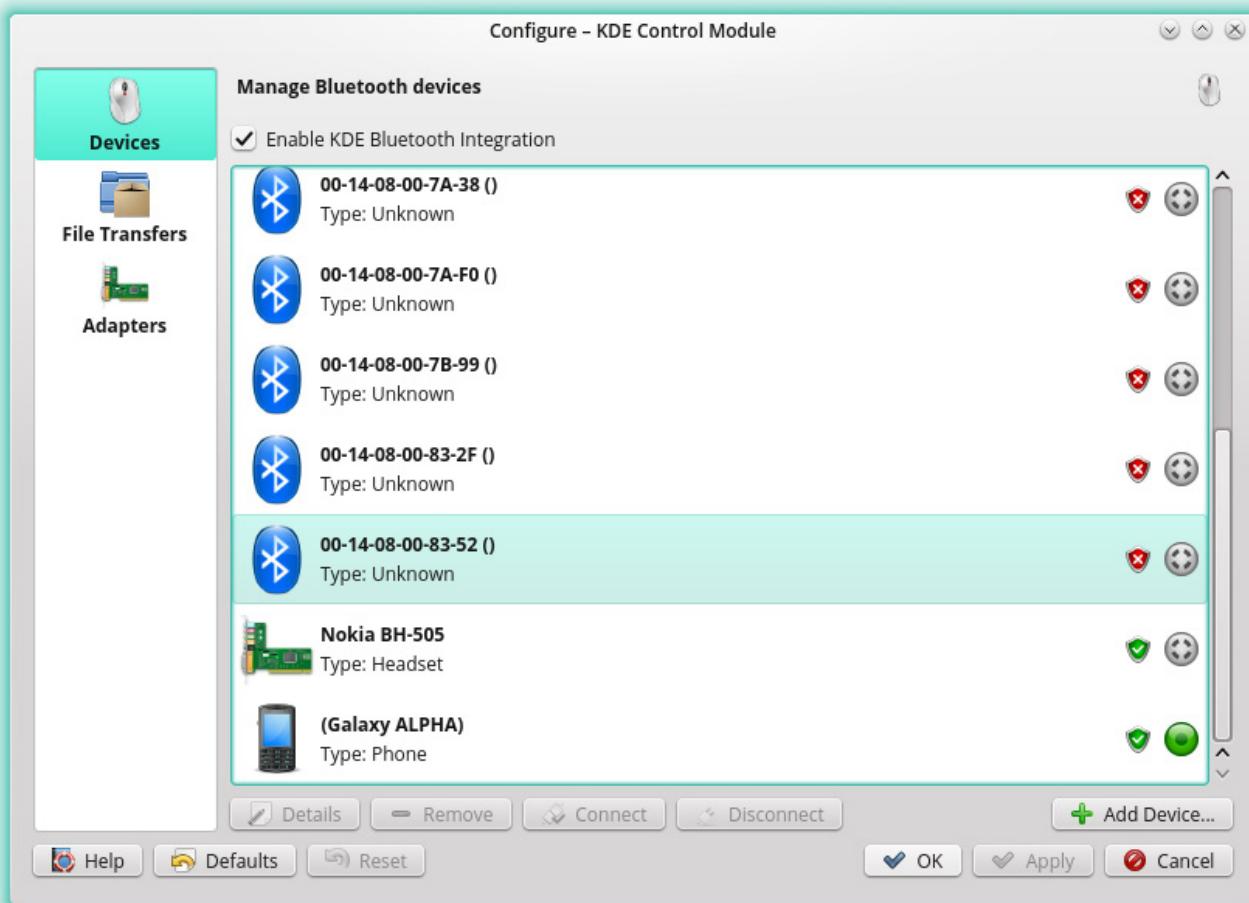


Figure 1. If your neighborhood is densely populated, you'll see an abundance of ready-to-connect Bluetooth devices.

that technique, you can discover smartphones, headsets, printers, wearable gadgets and, of course, desktops and laptops. On many Mac OS X systems, Bluetooth is set in discoverable mode silently and forever, plus it carries the owner's name in the device's description, which is perfect for amateur social engineering.

Handle and Browse Your Files

Although a Bluetooth connection is not very speedy when it comes to

transferring files, it still is a viable option. In Linux, there are several ways to access your device's memory via Bluetooth, and all of them play with the OBEX protocol.

To make use of it, you'll need a phone or a smartphone that supports ObexFTP service. Most legacy devices do, but some modern Android smartphones don't include ObexFS support by default. Luckily, you easily can fix it by installing a third-party Obex server from Google Play (like this: <http://bit.ly/1BjUkEw>) and

re-connecting your device with the Linux PC. After that, both GNOME Bluetooth and Bluedevil will show you an option for browsing the device, and if you click it, a Nautilus/Nemo/Dolphin window will appear with your device's contents.

However, let's go a little further this time and mount a Bluetooth device as a regular filesystem.

Assuming that you know its BD address and created a directory for mounting, do the following:

```
obexfs -b 11:22:33:44:55:66 /path/to/directory
```

To unmount it, use the `fusermount` command:

```
fusermount -u /path/to/directory
```

You even can do an auto-mounting of your Bluetooth filesystem in the traditional UNIX-style way. Just add the following line to your `/etc/fstab`:

```
obexfs#/b11-22-33-44-55-66 /path/to/directory  
  fuse allow_other 0 0
```

By default, only the current user will see the mounted filesystem. That's why the `allow_other` option is needed. Keep in mind that it might be a security risk to allow everybody to see remote filesystems.

The `obexftp` command is used for sending and receiving files without mounting a filesystem. You can send any file to your device by issuing a command like this:

```
obexftp -b 11:22:33:44:55:66 -p /some/file/to.put
```

The file will land in the default "Received files" directory on your device. To retrieve a file from the phone, just change the command a little:

```
obexftp -b 11:22:33:44:55:66 -g to.get
```

where the "to.get" file must reside in the phone's shared folder.

Play with Proximity

The Bluetooth receiver in Linux is managed via the BlueZ software stack, which, in turn, can scan your surroundings for other Bluetooth devices. A common case is a desktop or laptop PC, which you can pair with your smartphone. On the PC side, the Bluetooth adapter not only scans the range (it's the `hcitool scan` command), but it also estimates the strength of the signal. The stronger the signal, the closer the device is. You can make use of it if you associate a change in proximity with some action. For example, you can lock your PC when you're away and unlock it when

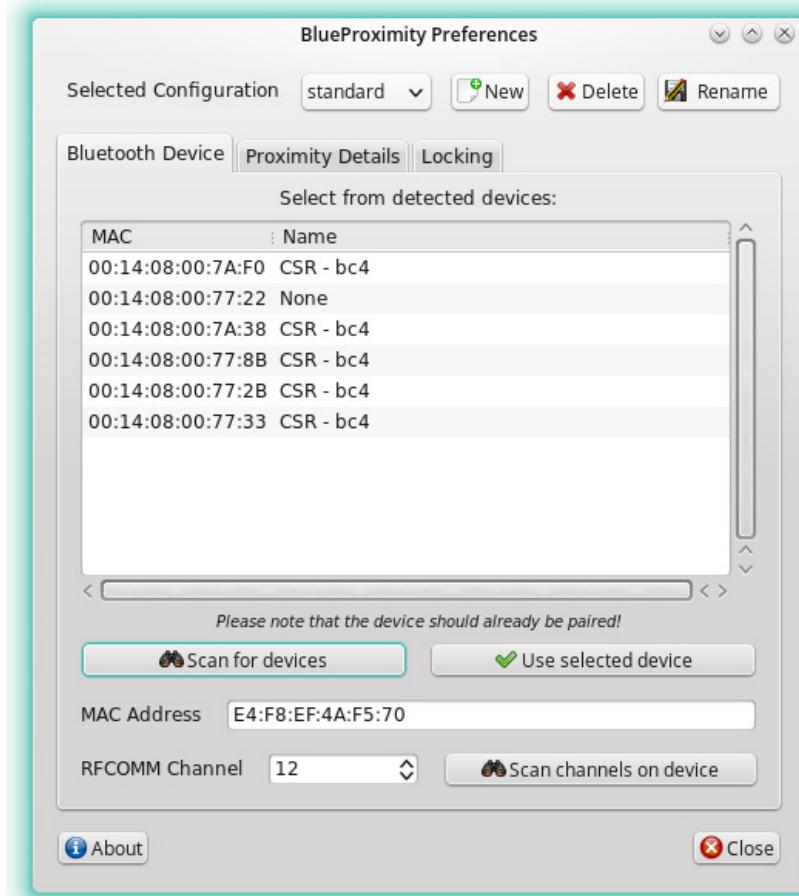


Figure 2. After spending a couple minutes to fine-tune proximity settings, you can surprise your friends with a PC that responds to your approach.

you approach it again. From a third-party's view, this will look like magic.

Linux has a ready-to-use solution for this purpose, and it's called Blueproximity. The utility is a Python wrapper over the `rfcomm` and `hcitool` commands, with a nice GTK2 user interface. Blueproximity is widely available across most Linux distributions, so just install it using your standard package manager.

After you launch the application, a tray icon will appear on your

panel. Click it to open the Blueproximity settings window. Make sure you've enabled the Bluetooth discovery on your phone, and then press the "Scan for devices" button. Select your phone (a caption will help you recognize it) and then go to the Proximity details tab. Here you can adjust the locking and unlocking distance and duration. Drag the sliders to make the values work best. The values depend on your room configuration, Bluetooth adapter active range and numerous other details. But to make things work, go to the Locking tab and make sure the commands there work for you. The Blueproximity defaults

are designed to work in GNOME, Mate, XFCE, Cinnamon and Unity environments, so if you happen to use one of those, no changes are needed. In the case of KDE, use the following replacements:

```
qdbus org.kde.screensaver /ScreenSaver Lock
↳ // for locking the screen
qdbus | grep kscreenlocker_greet | xargs -I {} qdbus
↳ {} /MainApplication quit // for unlocking
qdbus org.freedesktop.ScreenSaver /ScreenSaver
↳ SimulateUserActivity // for simulating user activity.
```

After you close the Settings window, you're ready for a test run. Walk back and forth to make sure that the values in Blueproximity are comfortable, and try the auto-locking and unlocking trick. If for some reason you need to monitor the proximity without bringing the Blueproximity setting window to the front, use the following command:

```
watch -n 0.5 hcitool rssi 11:22:33:44:55:66
```

where 0.5 is the refresh period and 11:22:33:44:55:66 is your phone's BD ADDR address. To find your address, simply run `hcitool scan` on the command line.

Back in Dial-up Times

It's always good to have a backup Internet connection. Sometimes your main provider fails, or there is some service downtime,

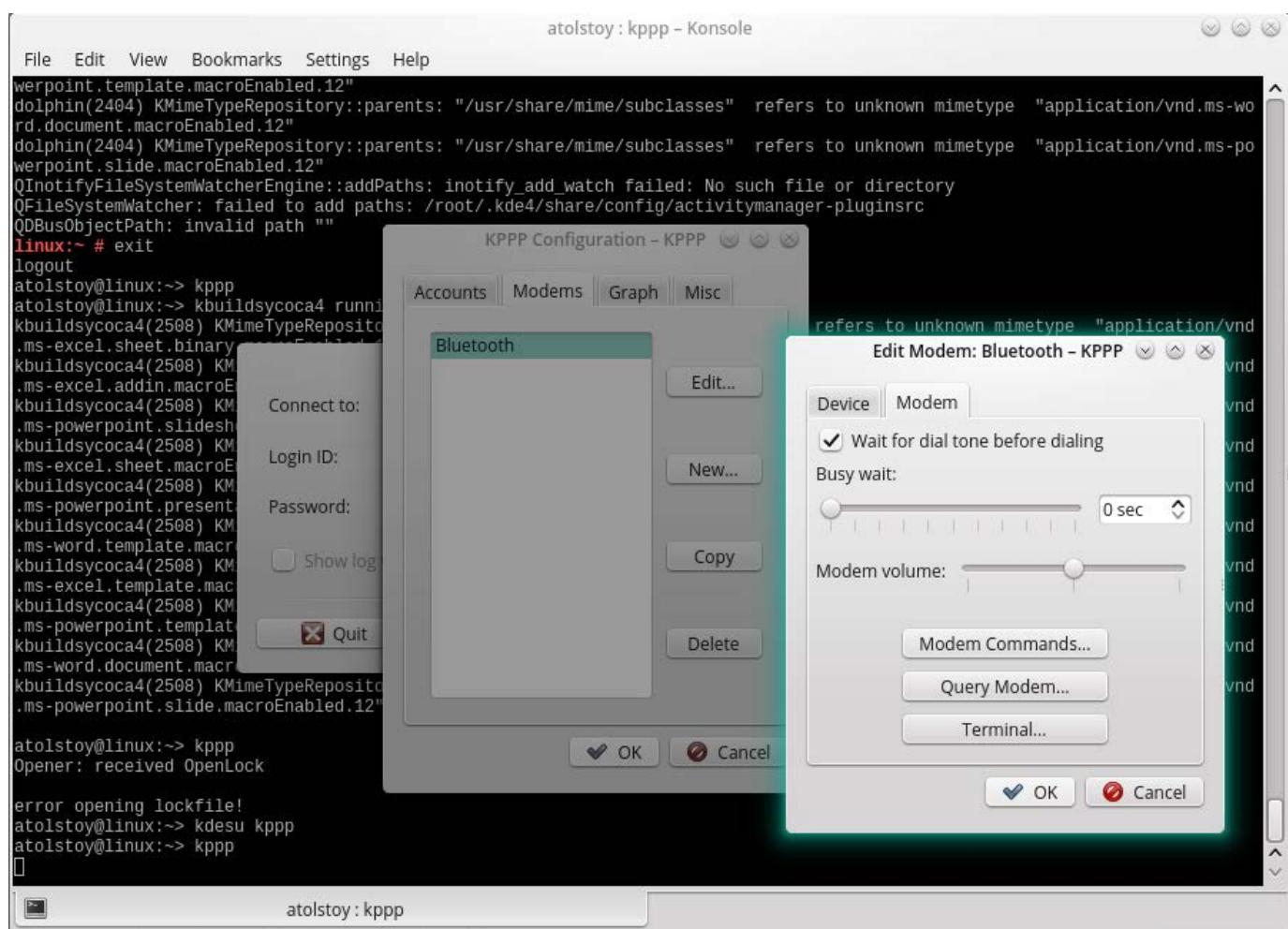


Figure 3. Although it looks like a retro type of connectivity, PPP perfectly supports all the latest tech novelties, including broadband LTE.

or whatever. Having another connection via your mobile carrier is great, but you may want to share this connection on your desktop PC. Basically, there are two methods for accomplishing it: launching a Wi-Fi ad hoc spot on a phone and connecting with Bluetooth. Let's look at the latter case.

Pairing your phone with a PC via Bluetooth is commonly done for browsing files, while binding makes the phone accessible as a /dev/rfcommX device (usually rfcomm0). This is a network device, and it can be used in the same way as a dial-up modem. After that, you can set a PPP (point-to-point) connection either with NetworkManager, KPPP, wvdial or the standard network-managing commands of your system (like the legacy ifup/ifdown commands). Here, I cover the KPPP method. Once you install the KPPP package, you're ready to go.

First, make sure your phone can access the Internet—this can narrow your search for a bottleneck if something goes wrong. Mobile phones are capable of doing this since the early 2000s, although 3G and LTE speeds came later. Then enable Bluetooth on the phone and make it discoverable (for security reasons, it is disabled by default on

most devices).

On the desktop side, you'll need to pair your phone with a PC. In the past, this was done manually by changing the BlueZ config files, but on modern desktops, all you have to do is use either the GNOME Bluetooth applet or Bluedevil to create a simple pairing.

Check whether your device has emerged on the PC side by issuing the `sudo ls /dev/rfcomm*` command. If it's not there, you'll need to do things manually:

```
hcitool scan // find your phone's address
sdptool search DUN // find the channel for dial-up networking (DUN)
rfcomm bind 11:22:33:44:55:66 1 (where 1 is a channel number)
sudo chmod 777 /dev/rfcomm0 // that will let us run KPPP
➥without root privileges
```

After that, it's time to run KPPP and set the connection details. Click Configure→Modems→New, and select /dev/rfcomm0 as your device. Press the "Query Modem..." button, and make sure that the modem device responds.

To establish a connection, you will need to alter the "Initialization string 1", which is in the "Modem Commands" window. The string differs between carriers, so you'll need to get the proper one. For example, Verizon in the US accepts

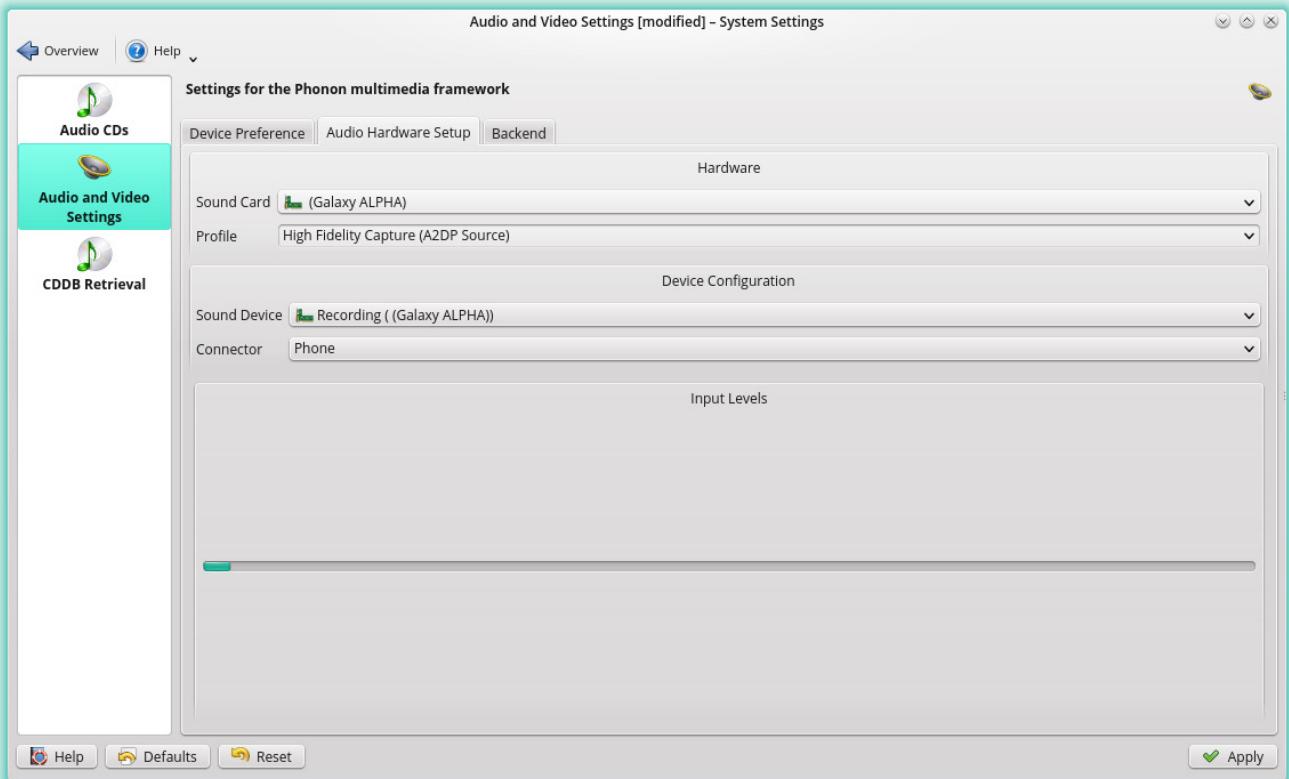


Figure 4. Previous versions of BlueZ in Linux required extra skill to stream audio to remote output. Now all the setup steps are just a few clicks away.

the following:

```
AT+CGDCONT=3,"IPV4V6","vzwinternet","0.0.0.0",0,0
```

When your modem is set up, create a standard connection in KPPP. The only thing you need to know is the dial-in number, which can look like *99***3# (for example, for Verizon). The user name and password fields can be left blank or filled with random characters, as they're not required. Finally, press the Connect button in the main KPPP window and wait for a successful link.

Audio Playback

A useful and impressive trick is streaming your audio playback from your Bluetooth-enabled phone to your desktop PC (perhaps with powerful speakers attached). Or, it can be applied to your car system, so you can listen to your favorite tunes while driving.

The setup procedure is straightforward, because your phone services are discovered automatically when it is paired with a PC. After that, open your Pulseaudio control window (Pavucontrol or Audio and

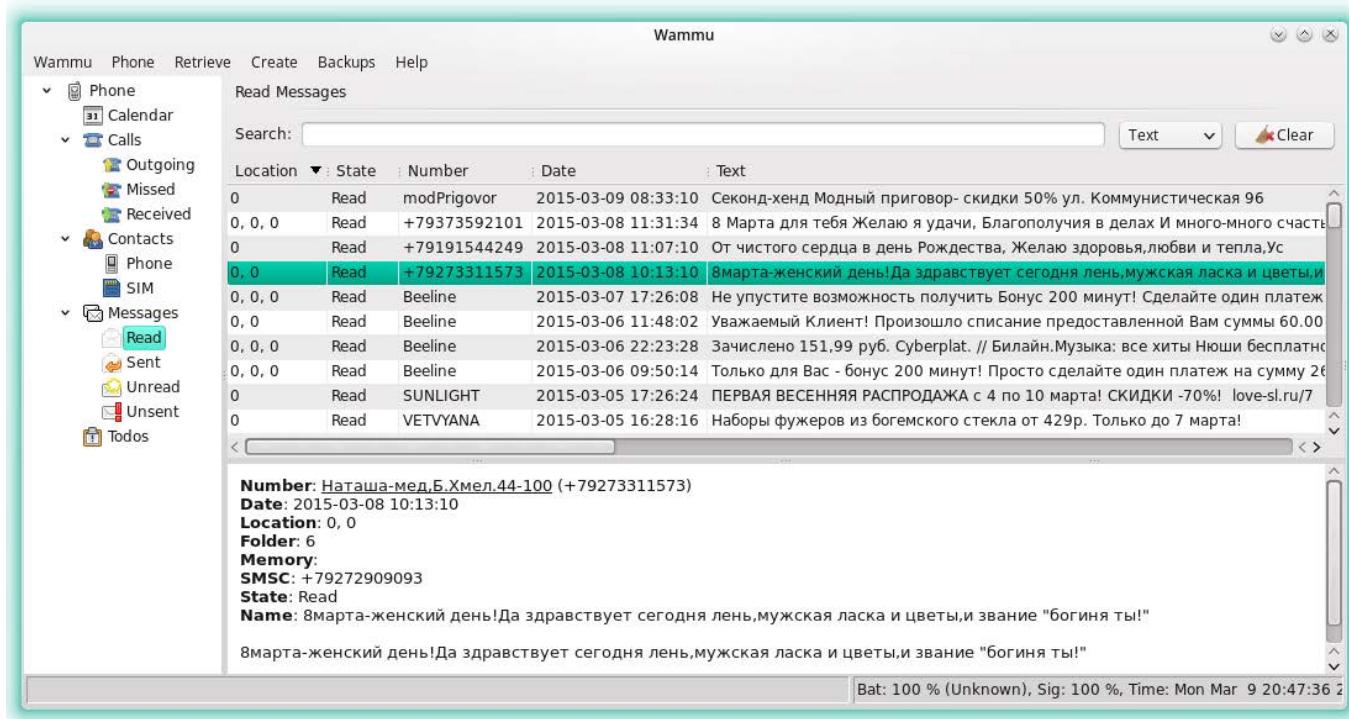


Figure 5. A phone itself might not be worth a penny, but if it stores precious content, it can be carefully retrieved and saved by Wammu.

Video Settings in KDE's system settings) and find your phone as an Input device. In the case of Android phones, enable multimedia streaming in the Bluetooth settings, and you're done.

Thanks to Pulseaudio, you can use your Bluetooth headset in a similar way—set it as a priority recording and playback device. To do this, you'll have to switch the profile from A2DP (high-quality stereo) to HSF/HFP (headset mode). It plays and records only in mono, but you can get an extra benefit from Pulseaudio by utilizing it as a built-in noise and echo filter. For example, let's launch

Skype with that filter:

```
PULSE_PROP="filter.want=echo-cancel" skype
```

You can switch between Pulseaudio output (sinks) and input (sources) devices using the `pactl` and `pacmd` commands, which work the same way on any Linux system.

The opposite way is limited though. You can stream audio from your Linux PC to a headset or Bluetooth speakers, but in the case of Android phones, it's possible only via Wi-Fi, not Bluetooth.

Extract Contacts and SMS

This mostly applies to older phones,

like the Nokia S40 and S60 phones, Sony Ericsson and so on. These phones still are widely used as backup phones, because they're rugged, friendly and easy to use. Unlike modern smartphones running Android or iOS, these mobile phones don't have advanced sync options with on-line services. Instead, you can retrieve various data stored on such a phone via Bluetooth. This is particularly useful if you own an old mobile phone, and there's no way to get into its memory other than wirelessly.

On Linux, there's a mature and feature-rich application for that purpose: Wammu. Go ahead and install it from your favorite Linux distribution (it's available almost everywhere). Once it is installed, launch it and follow the first-time connection wizard. When prompted about connection type, choose Bluetooth, enable it on the phone and click Next. The wizard may ask you a few more details about the connection, but you should be finished in a minute or two.



Powerful: Rhino

- Rhino M4800/M6800**
- Dell Precision M6800 w/ Core i7 Quad (8 core)
 - 15.6"-17.3" QHD+ LED w/ [X@3200x1800](#)
 - NVidia Quadro K5100M
 - 750 GB - **1 TB hard drive**
 - Up to 32 GB RAM (1866 MHz)
 - DVD±RW or Blu-ray
 - 802.11a/b/g/n
 - Starts at \$1375
 - E6230, E6330, E6440, E6540 also available

- High performance NVidia 3-D on an QHD+ RGB/LED
- High performance Core i7 Quad CPUs, 32 GB RAM
- Ultimate configurability — choose your laptop's features
- One year Linux tech support — phone and email
- Three year manufacturer's on-site warranty
- Choice of pre-installed Linux distribution:



Tablet: Raven

- Raven X240**
- ThinkPad X240 by Lenovo
 - 12.5" FHD LED w/ X@1920x1080
 - 2.6-2.9 GHz Core i7
 - Up to 16 GB RAM
 - 180-256 GB SSD
 - Starts at \$1910
 - W540, T440, T540 also available



Rugged: Tarantula

- Tarantula CF-31**
- Panasonic Toughbook CF-31
 - Fully rugged [MIL-SPEC-810G](#) tested: drops, dust, moisture & more
 - 13.1" XGA TouchScreen
 - 2.4-2.8 GHz Core i5
 - Up to 16 GB RAM
 - 320-750 GB hard drive / 512 GB SSD
 - CF-19, CF-52, CF-H2, FZ-G1 available

EmperorLinux
...where Linux & laptops converge

www.EmperorLinux.com
1-888-651-6686

Model specifications and availability may vary.



Bluetooth and Wi-Fi

Many actions can be done via both Bluetooth and Wi-Fi, so you may be curious about why one would use Bluetooth. The technologies are totally different. They have different purposes and even different power requirements. Bluetooth devices emit a signal that travels for about 30 feet, while Wi-Fi signals travel about ten times as far, which results in much a higher power consumption in the case of Wi-Fi.

Bluetooth also can be a better solution for older PCs that you might want to breathe new life into. Instead of obtaining a Wi-Fi module, try a Bluetooth USB dongle. This will result in a much more versatile usage, including Internet access.

Thousands of phones are supported, so your model most likely will be on the whitelist.

In the main Wammu window, choose Phone→Connect, and make sure the connection is established. Now you can retrieve data from the phone, including Contacts (either on SIM or in phone memory, or both), Calls History, Messages, To-Dos and Calendar. All of those options are available from the Retrieve menu. Whatever you choose, no changes will be made on the phone side, so it's absolutely safe.

The left part of the Wammu window hosts a category tree, where you can switch from one data section to another. You can add, edit or remove items and contents, which is far

more comfortable than struggling with phone controls and reading a tiny screen. One of Wammu's best features is the ability to back up and restore retrieved data. Look under the Backups menu for importing/exporting and saving options. If you wish to transfer data from one phone to another, there's no better choice than Wammu.■

Alexander Tolstoy is a desktop Linux power user and experienced sysadmin, and has been devoted to the Open Source community for years. He installed his first distro from a floppy disk and hasn't seen the Start button ever since.

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



Where every interaction matters.

break down your innovation barriers

power your business to its full potential

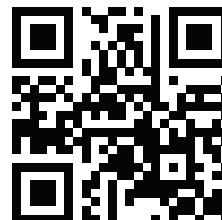
When you're presented with new opportunities, you want to focus on turning them into successes, not whether your IT solution can support them.

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

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

Want more on cloud?

Call: 844.855.6655 | go.peer1.com/linux | View Cloud Webinar:



Linux and the Internet of Things

"When wireless is perfectly applied the whole earth will be converted into a huge brain, which in fact it is, all things being particles of a real and rhythmic whole. We shall be able to communicate with one another instantly, irrespective of distance. Not only this, but through television and telephony we shall see and hear one another as perfectly as though we were face to face, despite intervening distances of thousands of miles; and the instruments through which we shall be able to do this will be amazingly simple compared with our present telephone. A man will be able to carry one in his vest pocket."—Nikola Tesla

ANDREY KATSMAN

I wake up in the middle of the night, mouth parched and vision blurry, and fumble around to find my iPhone. I press my thumb to the fingerprint scanner, and in the dim blue light, just out of instinct, I squint at the screen, find the right app, open it, and check the ambient temperature and air quality indoors. It turns out my goose bumps are lying; the temperature is quite comfortable and not arctic-level freezing, which is how it feels to me. Now, I touch a virtual switch, and warm yellow light illuminates my way to the kitchen. I slowly waddle towards the water bottle without stumbling over one of my cats, have a drink, and safely waddle back to bed. Another tap on my app screen, and the light fades away; I sleep.

This is my reality, and an increasingly common one in many people's homes. The only difference? My tech is mostly self-made—with few exceptions, of course.

Our homes slowly but surely fill up with small, smart gadgets that make our control over our surroundings stronger and more precise. These are the tools that help us create customizable, responsive home environments that anticipate our needs and intelligently adapt to our rhythms.

Perhaps these gadgets even make us a little spoiled, but mostly, they help us feel comfortable in and educated about our environments. Never before have we known so much in precisely quantified units about our living spaces, working spaces and ourselves. Smart homes tell us about the air we breathe, the water we drink, and the temperature and humidity in which we live, and vigilantly guard us from strangers or threats. They have become an extension of ourselves in the best possible way, automating important but mundane tasks so we can focus on more important things, like maintaining the late-night cat/human detente.

The Four Factors That Led to IoT

What are the circumstances that allowed this wonderful reality to materialize? From my perspective, I think of four factors as being largely responsible:

1. Tiny, inexpensive, power-efficient-yet-powerful processors replacing older, simpler microcontrollers.
2. Cloud processing becoming cheap enough to be accessible and affordable for large and small companies alike.

3. Smartphones becoming powerful multicore computers that are practically ubiquitous.
4. Linux making it remarkably easy to spin up a smart application that can run on anything from toaster to a spaceship with minimal effort, and that same Linux OS powering the back-end cloud side.

Let's flesh this out a little more.

Microcontrollers: Silicone vendors have improved manufacturing technologies so much that each transistor in a processor is now imperceptibly tiny (14 nanometer technology!). They bring magic to life with their new lines of low-powered systems on chips (SOCs) and have brought the market to the point where a chip the size of a small postage stamp costs \$2–\$7 and still has:

- A full hardware network stack (that is, a built-in capability to connect to the Internet).
- A dual-core ARM family processor.
- Enough computation power to run an operating system.
- Plenty of other useful features that

all run off a single coin cell battery.

All you have to do is stick it in a package—a watch, a small gadget on the wall, a light bulb, or whatever your company desires—and you instantly “smartify” your product. This kind of luxury (which wasn’t available even a decade ago) has driven companies to opt for the use of true operating systems on their devices (namely Linux) and to forgo the older, more difficult and less efficient path of direct microcontroller programming with a single “forever” loop and every software aspect done in-house.

The Cloud: At the same time, organizations like Amazon are offering their cloud infrastructures to anyone for reasonable costs. This enables companies to wield unimaginable computational power—still running Linux, mind you—without investing anything in the purchase, physical installation or maintenance of the hundreds of powerful processing machines at their disposal. A company of two people can bring distributed services to its customers that require intensive computational power; this would have been a complete deal-breaker (even for large companies) in the pre-cloud era.

Smartphones: There is not much to be added about the role of

smartphones in the “smartification” of our environment. As a society, we quickly adapted to the beautiful, intuitive control interfaces and quick response times, and they’ve become so much a part of our daily lives that parting with one, even just for a few hours, causes most people to feel naked. What many people do not realize, however, is the vast

Galaxy, has a Qualcomm Snapdragon with a quad-core 2.5GHz processor, 2GB of RAM, and 32GB of fast Flash storage and a 32 pipeline 3D hardware graphic accelerator.

To review, just looking at CPU speed and quantity of cores, my phone has roughly 250 times more

As a society, we quickly adapted to the beautiful, intuitive control interfaces and quick response times, and they’ve become so much a part of our daily lives that parting with one, even just for a few hours, causes most people to feel naked.



computational power in each and every modern smartphone. Just to bring a few personal reference points into the picture:

- 1993: my first computer was an Intel 386 40MHz with a whopping 8MB of RAM, 170MB of hard drive storage and a 1MB graphics accelerator.
- 2015: my smartphone, a Samsung

computation power than my desktop did. I’m not even taking into account the so-very-useful additional features that every phone packs nowadays—portability, elegant UI, constant connectivity and precise sensors like GPS, accelerometers, gyroscopes, magnetic compasses, ambient light sensors and megapixel/HD cameras. All of those factors are pure gold for making our environment smart and creating IoT concepts.

Linux: Linux is the final component that makes the Internet of Things a reality—the glue that holds everything together. How, you ask? Well, let's look at a typical product and try to understand how Linux contributes to and affects each step of development.

Let's start with the end point—the wearable or home-based gadget. It usually hides either a high- or low-end ARM-based SOC inside it. (Considering that even low-end microcontrollers are capable of running a small operating system, it's safe to assume this one could as well.) Now, as a company, what sounds easier: a) building an original software environment from scratch that includes task scheduling, memory management, peripherals access that supports multiple technologies (for example, I2C, SPI, SDIO and so on) and creating your own implementation of a network stack, including various cryptology solutions to support secure socket layer (SSL, the almost omnipresent secure communication standard over the Internet); or b) taking a free, constantly evolving and improving operating system tested by billions that provides all of those things and more? Obviously, b is the clear winner.

The Rise of Linux

Through the years, Linux has become such a complete solution that you would need to find an incredibly convincing argument to choose an alternate approach to a hardware OS. Not only is the Linux kernel versatile and easy to tweak and adjust to fit the needs of a project exactly, but it's also cross-compilation-friendly; Linux can be brought to almost any given platform, and as operating systems go, it can be very low maintenance in terms of hardware resources. This is an operating system internally built to support almost any imaginable hardware layout or peripheral, and while it's not completely plug-and-play and certainly requires an investment of labor, it comes ready for such work and aims to make it as easy as possible.

Let's move on to the next part of the chain—the cloud, where the power of Linux manifests itself from a different angle. Now, ideally, we want an operating system that is capable of managing vast computation power, many processors on one machine, high-throughput network stack, and if you don't mind, could you please throw in off-the-shelf powerful solutions for different aspects of the cloud infrastructure? Oh, and could you

also make it all free? Yes, it's all that—your operating system is that same Linux, but this time, it's tweaked to answer your server-side demands, most of the additional components any company needs (such as message queues, caching, Web servers, databases and so on) and comes at no cost thanks to the Open Source community. This is

developed a wide, Open Source community that creates and maintains the various infrastructure solutions (like the ones mentioned above), and these same solutions get adjusted and ported to work in the mobile environment as well. To that end, even if Linux itself does not necessarily run on the mobile phone, its derivatives and side products

To that end, even if Linux itself does not necessarily run on the mobile phone, its derivatives and side products often are present, simplifying the process of integrating the smartphone as part of the Internet of Things ecosystem.

mind-bogglingly wonderful.

The last part is the smartphone. Here, it gets a little trickier—how are an Android or iOS phone, an Internet of Things solution, and Linux all connected? Indeed, the link is a less obvious one. (And as a tangent, let's quickly call out that Android is, in fact, Linux! It's the same old Linux with a little polish on top known as the Android services layer.)

The main thing is that Linux has

often are present, simplifying the process of integrating the smartphone as part of the Internet of Things ecosystem.

The Inherent Benefits of Linux

So far, I've discussed the direct effect of the evolution of processors on the adoption of Linux and the versatility of Linux as a factor for its selection as operating system of choice on both front end and

back end/cloud. However, there are other aspects to the use of Linux that benefit companies and further proliferation in the Internet of Things ecosystem.

First, off-the-shelf solutions that help companies develop and deploy solutions without re-inventing the wheel are available on the cloud side, and are applicable at all the points where Linux is used. This means that your hundreds of cloud servers and tiny smart light bulbs potentially could be using the same exact code base for the same purposes—for example, message parsing or queuing—thus saving money, time and manpower.

Second, manpower and skill set also are transferable across different components, provided that both run Linux. A startup script for a toaster and a startup script for a cloud database server require the exact same Linux system administration skills, meaning your personnel could be more productive and contribute to more parts of the system. This allows for easier transfers between different teams, and as a result, happier engineers. Although this may sound like a soft benefit, this is a very serious advantage for companies that build on Linux.

Finally, it is good to note the

benefit of community development. Linux and its components are being developed, used and maintained by a huge community. This is a prime example of the “law of large numbers”, meaning that most bugs eventually will be detected and (mostly sooner than later) fixed by the community that acts as an innate immune system. Thus, the products developed gain the benefit of being more robust and error-free.

A Practical Application

One of the things that has been keeping me busy at work lately is the integration of different sensors into our product. Sensors are tricky little guys; some are simple and straightforward, not much more than a resistor that changes according to environmental factors. (Think of a thermistor that's affected by temperature or a photoresistor that changes based on the amount of light it absorbs.) Some are trickier and need a groundwork of special mechanisms before they're used, like I2C or SPI bus-connected sensors.

Although the first kind of sensor can be easily integrated anywhere, usually reading values off a built-in, analog-to-digital converter—a feature that exists on most modern SOCs—the second kind would drive you bananas

before you're able to force it to talk to you—or it would, unless your operating system already supports it.

Case in point: let's look at the integration of one of those trickier sensors—an I2C-based temperature and humidity sensor by Sensirion.

The sensor has a slave address of 0x70, and we'll use a single "give us data" command word 0x5C24 to mean "give us a humidity sample followed by a temperature sample".

First, let's assume we're not using Linux and working "bare-bones", so to speak. The SOC in question has a built-in I2C controller—in fact, it has two—and the controller is represented by a collection of registers mapped to the physical memory. Communicating through it would mean changing those values in a timely fashion to make the communication possible. However, there are a few twists; there are three I2C buses that are multiplexed between the two controllers because the second controller handles two buses, and there are multiple devices on each of the buses.

Here is "simple code" to write the bytes to request temperature from the sensor:

```
mov    r0, #CONTROLLER_BASE  
orr    r0, r0, #DEVICE_OFFSET
```

```
mov    r1, #0x01  
str    r1, [r0, #PRESCALE_LOW]  
mov    r1, #0x01  
str    r1,[r0, #I2C_ENABLE]  
mov    r1, #0x04  
str    r1, [r0, #FMCTRL]  
mov    r1, #0xE0 // <-- slave address  
str    r1, [r0, #FMDATA]  
mov    r1, #0x5C //<-- first part of measurement command  
str    r1, [r0, #FMDATA]  
mov    r1, #0x24 //<-- second part of measurement command  
str    r1, [r0, #FMDATA]  
mov    r1, #0x0A  
str    r1, [r0, #FMCTRL]
```

(Keep in mind, this is before waiting for acknowledgement and later reading the actual output values.)

To put it simply, you have to do everything yourself and control the communication process end to end. And as a bonus, it usually completely messes up the readability of your source code.

Now, let's look at the same exact example with Linux i2c support in place (assume there's an open handle for the I2C device):

```
static char sample_cmd[] = { 0x5c, 0x24 };  
ioctl(i2c_fd, I2C_SLAVE, 0xE0);  
write(i2c_fd, sample_cmd, 2);
```

Needless to say, reading the output is just as simple, making

Linux the obvious solution for this application.

Looking Ahead

I believe that the Internet of Things is not a passing trend, and together with continued improvement in the low-end processors market and decreasing prices of cloud computing, we'll see more and more solutions showing up all around us.

Our homes are the first front of IoT adoption, and with the explosion of smart wearables, our bodies are the second wave. What will the next canvas be?

The main trend for the next decade will be an increased number of interconnected devices filling up the space around us, perhaps with some harvesting green energy as the power demands for such devices also become smaller and smaller. Perhaps the streets will become smart and interconnected? I easily can imagine streetlights responding to human presence and adjusting according to personal preferences ("I feel like walking the streets in purple today."), thus conserving energy by not illuminating unattended streets. Or we could see more interconnected engagement with our cars, which already is happening, but on a very low scale as compared to the potential. Eventually,

we will all live in a world that responds very precisely to our needs and wants, a more pleasant space where it is safer for our children and ourselves to live. As for Linux, it will continue to play a crucial part in these advances either as the main platform of choice or the platform on top of which all new solutions will be developed. It is possible that soon enough low-end processors will be so powerful that bare-bones Linux will be replaced with Android as the chosen solution for gadgets, but worry not; it will still be the same old Linux, just with benefits.

On this note, I have to finish. My cats are getting hungry and my last attempt at making a smart cat feeder did not end well—for the feeder, that is. Some things remain to be done by hand, at least for now, but I know I'll have better success with the larger/non-feline projects in my life as long as I have Linux. ■

Andrey Katsman runs the embedded software team for Canary. With more than a decade of software design and development experience, Andrey has participated in a range of military and civil projects. In his spare time, he loves to travel the world and entertain his cats.

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

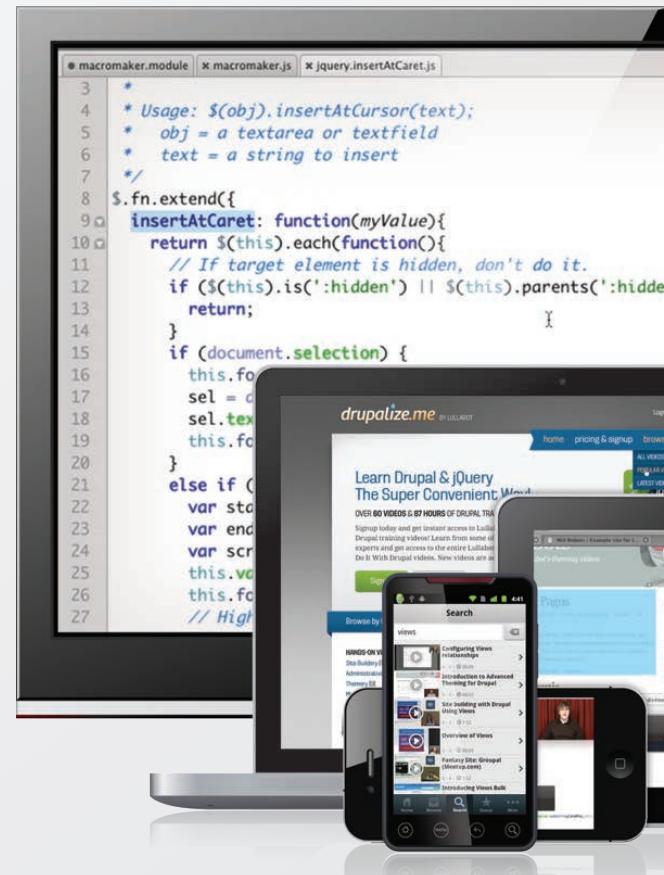


Instant Access to Premium Online Drupal Training

- ✓ 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!



WEBCASTS



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

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

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



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

Sponsor: SAP | **Topic:** Big Data

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

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

WHITE PAPERS



White Paper: JBoss Enterprise Application Platform for OpenShift Enterprise

Sponsor: DLT Solutions

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

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

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

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

WHITE PAPERS



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

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

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

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



Standardized Operating Environments for IT Efficiency

Sponsor: Red Hat

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

Benefits of an SOE:

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

SOE leads to:

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

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

Introducing Luwrain

Can GNU/Linux help us rethink accessibility solutions for the blind?

MICHAEL POZHIDAEV

When blind people work on personal computers, their mental perception of images replaces visual data on the screen. Users construct these images by getting portions of data transmitted from the machine in one of two alternative ways: either through the help of a speech synthesizer or through a braille display. The most popular solution to this kind of accessibility technology allows users to work in a typical graphical user interface (GUI) with the assistance of screen-reading software. The utilities, usually called screen readers, announce every step users perform by describing each change occurring on the screen as words.

This approach was a true breakthrough in the world of accessibility for blind people in information technologies. There have been a tremendous number

of success stories of people with disabilities who are able to work, study and do a lot of other things completely on their own. Despite these wonderful benefits, there still are unresolved problems, mainly caused by the GUI itself. When using existing screen readers, the mental image should correctly conform to the rather complicated window environment.

Children who are born blind are never able to see the sky and clouds. How can they understand a screen reader's feedback? The feedback always comes in terms of GUI elements, which always have a visual nature. Would blind people feel comfortable enough using this? Audible announcements of GUI changes on a computer might be a suitable solution if users had some prior experience in a windows system before losing their eyesight. Unfortunately, this is not always the case.

On the contrary, Luwrain offers a solution for everyday work for a fixed number of tasks with maximum comfort and speed.

The second reason to look for an alternative solution is the fact that GUIs are convenient and effective only with a mouse. Unfortunately, a mouse typically is not used for work with screen readers. Blind users have to work with keyboards only, sometimes emulating mouse actions with keyboard commands.

The Luwrain Project

All of these factors make work a major struggle with a GUI, and research has to continue to discover new ways of interacting to improve the situation. Research efforts have yielded some new technologies, such as Adriane (<http://www.knopper.net/knoppix-adriane/index-en.html>), but let's take a closer look at one more concept.

Luwrain is a GNU/Linux distribution in which the main user environment has an implementation on Java. Although its main part is suitable for running on any other operating system that has proper Java support, Luwrain launched in this way will be less functional than as a standalone OS.

Luwrain attempts to accomplish the following tasks:

- Making personal computers accessible to blind people who were unable to use them before due to various reasons.
- Making the environment of popular everyday applications more comfortable and effective for people who spend a lot of time working.
- Suggesting a solution for the problem of distributing accessible applications for blind people (it is exactly for this reason that Luwrain supports launch on any system with Java support).

Luwrain isn't a competitor to screen readers, because screen readers attempt to ensure that all things accessible for sighted users will be accessible for blind users as well, but without any care for practicality or convenience. On the contrary, Luwrain offers a solution for everyday

work for a fixed number of tasks with maximum comfort and speed. If Luwrain's functionality is insufficient for a particular situation, you can install it just as an application and leave other tools untouched.

Text-Based Interface

Luwrain's main characteristic that distinguishes it from a screen reader's approach is its new type of interaction. If you want to make an environment comfortable and effective, you should simplify the image in the mind that reflects the working space on the screen. While working, a user's thoughts should be focused on the work itself, and needless interface elements should not distract the user. Regarding Luwrain's user environment, it would be fair to acknowledge that Luwrain adopts a number of ideas previously proposed in the Emacspeak add-on for the GNU Emacs text editor (<http://emacspeak.sourceforge.net/>).

Let's take a closer look at the Luwrain interface. Luwrain rethinks the behavior of popular widgets in such a way that no visual data may be involved in their representation. A monitor still is used, but now it plays a supplementary role. Pictures on it help users with low vision, but there is nothing difficult if graphical

information is totally unknown. Next, Luwrain also requires a redesign of general application structure, because all new controls must be organized in a completely different way from what people are used to seeing with a GUI.

Historically, when the GNU Emacs authors tried to implement various features that definitely were much larger than the bounds of the text editor (for example, mail reading, calendars, file management and so on), they didn't realize what a brilliant idea they actually had. They clearly proved that nearly all objects can be represented in text form only. Indeed, roughly speaking, if you'd like to write somebody a message, you can ask one script to prepare a message template in a text file having a "To:" string on the first line, a "Subject:" on the second and the message body on the rest. After filling out all the necessary values in the offered template, you call another script, which looks at what you wrote, constructs a valid e-mail, and sends it through your favorite relay.

The text itself (as well as everything that can be described as text) is the most suitable data structure for blind people because they can explore it easily through speech or braille. When users jump from one line to another, they hear the line's text.

When moving from letter to letter, users get the character under the new cursor position.

Several extra commands enhance navigation. For example, when jumping between words, users would “hear” the next word. This behavior always is the same, regardless of what screen-reading technology the users choose. The substantial difference is that in a GUI, this kind of navigation is used for text edits only (which are, apparently, just a subset of a large variety of existing controls). In Emacspeak and in Luwrain, it is used everywhere.

Besides generally improved accessibility with a ubiquitous text interface, users have some additional advantages. If the text is everywhere, users can do a text search in any place they want. Users can press the corresponding hotkey and type the corresponding substring. After that, the search either finds the substring or ensures that this string is absent. The second useful aspect is that in any arbitrary place, regardless of whether it is a file manager panel, a calendar page, an address book entry or a media player control, it is possible to copy the entire content as text to the clipboard—for example, for e-mailing to a friend.

It is important to be able to know

what items are absent on the screen, because screen readers mostly take care of describing only the elements they recognize. Usually there is no way to ensure that something isn’t shown on the screen, because users can’t trust that screen readers understand the GUI completely. If everything is offered in text form, this is easy to do, because no problems arise when going through the text from top to bottom. In a GUI, on the other hand, it is much harder, because the focus doesn’t need to stop at every element on the screen. Some of them can be skipped.

Constructing an Application for Luwrain

It is impractical to try to access an entire application’s functionality in one solid text space, so usually it is recommended to split an application’s objects into several parts called areas. Each area implements the text-interface behavior. Their exact setting is dependent on the purpose of the application and should be designed very carefully. If you consider a double-sided file manager, you can see that it requires three independent areas: two panels for directory browsing (left and right) and one more for listing continuous user actions (copying, moving or deleting).

All of them behave independently, which means they have their own cursor, and changes in one of them do not influence the others.

All of them are shown on the screen in such a way that allows filling the entire screen space (as tiles). Two panels are positioned as usual, and the action list goes to the bottom. Once again, appearance on the screen no longer plays a substantial role; it is just a complementary source of information for people with low vision. But as the screen still exists, let me mention a few additional details about it. For better adjustment for special needs, it requires some new features. Each area on the screen is filled only with text data drawn with a monospaced font. Users can choose the color scheme and the font size freely during work without any support by the application.

Each of the three areas, which the file manager consists of, behaves as a list. However, no matter what they are, they should be accessible as text. Each particular item is on a separate line; the movements are announced with some supplementary information, such as whether the current item is a directory or a regular file. All extra comments can be switched off if the user holds the Ctrl button on the keyboard while jumping from line

to line. This little trick significantly increases efficiency, but it is not the only example in the file manager. Luwrain has such tricks in various places everywhere, and that makes the environment truly more suitable for blind people compared with screen readers, which sometimes are not able to distinguish the main content from the decoration.

Luwrain Widgets

Luwrain has its own library of classes that are useful for constructing an interface with the controls. Generally it is similar to the usual GUI widgets, but redesigned in the ways described previously. This library includes lists, trees, text inputs, forms, menus, some special items like calendars and so on. Of these, the forms are probably the most difficult. The forms in Luwrain are text areas containing one particular control on each line. Lines begin with a control name and can represent a text input, a yes/no check box, an embedded list and so on. Lists always take a single line while a selection is always performed in a separate area.

In contrast to GNU Emacs, Luwrain doesn't have real buffers with real text content for all objects. Text representation is there just as some sort of intermediate level, obscuring internal structures. The internal

But, one thing plays a crucial role—the substantial number of existing libraries created for Java developers, the most important being JavaMail, Rome (the RSS parser), Apache POI (office document filters) and many others.

structures can be very complicated without any restrictions, and their logical level makes them easily observable by a user. You can design your own approach on how to handle any new nontrivial data model in a text way, creating a new custom control.

The Environment Internals

Behind a rather simple front-end conception, there is a complete UI infrastructure based on events dispatching and handling with the support of multithreading and user dialogs (like some type of modal windows). As I already mentioned, the main language chosen for Luwrain is Java. Discussing all reasons for this decision here is almost pointless, as the pros and cons for any particular language are very arguable. But, one thing plays a crucial role—the substantial number of existing libraries created for Java developers, the most important being JavaMail, Rome (the RSS parser), Apache POI (office

document filters) and many others.

The Luwrain core translates various input commands and internal environment actions to events that pass through the main loop queue and, after that, are dispatched to the corresponding object, considered to be the most suitable destination.

The events also are useful for multithreading synchronization. The application may issue as many threads as it wants, although only one of them is allowed to perform operations with the user environment. For this purpose, Luwrain has a corresponding type of events that safely allow data exchange between the background threads and user interface.

Pop-up Areas and Applications

The main loop procedure is capable of the recursive calls needed for the appearance of dialog areas (modal windows). These areas can be shown as a single method call placed inside some other wrapping handler. An

The Luwrain distribution includes several standard applications, such as a double-sided file manager, mail client, feeds reader, music player, command line to launch bash expressions (implemented through Java Native Interface), address book, personal scheduler, office document viewer and editor, notepad and others.

existing unfinished upper handler freezes the top-level event loop, and that potentially could freeze the entire user environment, but the recursive loop instances solve this problem.

A user gets the dialog areas along one side of the screen. Regular applications may open them only along the bottom, but Luwrain opens its main menu on the left (yes, Luwrain has a main menu, which can be opened by the corresponding button on the keyboard where everybody expects it to be), and on the right, there is the context menu with the actions associated with the focused object. Most actions in the main menu are accessible through short commands, which can be issued with a corresponding line opened with Alt-x and appear on the bottom of the screen. (The Luwrain command line is very different from GNU Emacs'. It

takes the system-wide commands only, which should be applicable regardless of what applications are currently opened.) Having this additional input method increases speed, because in a noisy environment—say, in an airport or aircraft—it is easier to type “message” quickly than to select menu items by listening their names.

Applications in Luwrain always fill the entire screen space. Only the amount of system resources limits the number of concurrently launched instances. Users easily can switch between applications by pressing Alt-Tab. (Actually, the term “application” isn’t the most proper one, as Luwrain applications share the same memory space inside the Java virtual machine and don’t go to separate processes, but it’s the easiest term to describe them.) The Luwrain distribution includes several standard applications,

such as a double-sided file manager, mail client, feeds reader, music player, command line to launch bash expressions (implemented through Java Native Interface), address book, personal scheduler, office document viewer and editor, notepad and others.

GNU/Linux Environment

Now you know enough about everything that happens inside the Java virtual machine, but there are a lot of external things that revolve around it. Luwrain should interact with system-level services for managing network interfaces, attaching/detaching removable media and for other similar tasks. The most convenient way to do that is, evidently, D-Bus. D-Bus is accessible from Java, and a number of existing D-Bus services cover a lot of needed features (for example, you can use Network Manager for network configuring). This means Luwrain will be consistent, as more services are introduced to D-Bus. If one day Lennart Poettering and the community around him get systemd in a commonly acceptable state that no longer raises any concerns, that will add the final missing piece to a nice design of Luwrain as a complete operating system.

Luwrain is capable of being installed by blind people freely without any

sighted help. This is achieved by cloning a live CD system to the hard drive and can be done rather quickly.

The Luwrain story would be really nice if all things could be executed inside the Java virtual machine, but unfortunately, that's not the case. Let's step back a bit and think about whether Luwrain is applicable for all commonly required tasks. It is not necessary to consider The GIMP or various types of CAD software, because the nature of such programs requires eyesight, regardless of what interface is offered for them. However, there are a couple cases that are exceptions for Luwrain. These are Web browsing and some closed applications (the most popular example of this is Skype). You can't complete these tasks in Java with a text-based interface, so there needs to be some alternative solutions.

Both Mozilla Firefox and Chromium support accessibility features for blind people but in different ways. Chromium does it through its Google ChromeVox extension (<http://chromevox.com>), which can work independently from any other GNU/Linux assistive infrastructure. I don't discuss it in detail here because it just works. Firefox does it through access to the so-called AT-SPI (Assistive

Technology — Service Provider Interface, <https://wiki.gnome.org/Accessibility>). AT-SPI initially was a part of the GNOME Accessibility Project and allows gathering all information of launched applications and their GUI elements into a single place. After collecting this information, it can be passed to some assistive technology that transforms it into speech and provides it to the user. Usually this job is done by the Orca screen reader, but Orca itself is strongly linked to the GNOME environment and isn't used for any Luwrain tasks. Luwrain includes a special tool to replace Orca that is intended for collecting AT-SPI information and translating it to a human-understandable form. (Great thanks to Mike Gorse, who always is ready to help and clarify details of AT-SPI behavior!)

The nature of basic Skype operations allows their transformation to a text-based interface, but it is closed software and Luwrain is unable to do anything with it. In the meantime, the GNU/Linux version's UI is based on Qt, which also is supported by the AT-SPI functions. All that is needed is to compile and install the special qt-at-spi bridge (<http://projects.kde.org/qtatspi>).

No doubt, it would be better not to have any exceptions for Luwrain's

text-based interface proposal, but we should be happy that it is possible to overcome such special cases in an appropriate way.

Speaking of the Luwrain details that work outside the Java virtual machine, let me point out that all tools launched in the X.org server (Luwrain's main window, a Web browser, Skype) are under control of the special tiny window manager (of course, not written from scratch).

Conclusion

In concluding this Luwrain technical overview, it is interesting to consider what Luwrain can and cannot change in the lives of potential users, although this project is still in a rather early stage. How will users possibly perceive of this system, which has many significant differences compared with most existing solutions? What new areas could it open in the technical world?

First and most important, Luwrain is aimed to be useful for users who previously were unable to interact with PCs due to various reasons. It might be appreciated by seniors for general operations, because Luwrain requires a minimal amount of technical knowledge. Its interface is constructed in a way that makes usage possible for most users after

following the short system guide offered for everybody on booting the live CD. Experienced users likely will be able to work without any prior learning at all (according to a series of our small experiments).

However, some things still may raise concerns. The environment is "in itself", which means that creating a new application will not make it available for blind users by default. Also remember that Luwrain doesn't imply the exclusive selection of a platform to work. Users can have several machines for different tasks. On a desktop, users can continue using a particular required application, but on a laptop, Luwrain could offer a better environment than any other OS. Also, if people want to use Luwrain due to its better organization, but aren't able to abandon Microsoft Windows or Mac OS X (for instance, because other family members use it), they still can run Luwrain on any OS with proper Java SE support. The same is true if users are unable to switch to GNU/Linux due to some specific software.

Let me also note that Luwrain allows using bash expressions. Does anyone refuse to ping just because it is a command-line tool? The command line itself is one of the most accessible conceptions for blind people.

Because Luwrain provides an easily

understandable interface for blind users (hopefully) and potentially could launch on any OS with Java SE support, it could be considered as a platform for distributing accessible speech-enabled applications. Perhaps there are vendors who would like to have an application that will be accessible reliably to blind people. For example, it could be used for payment systems, social applications for health care purposes and so on. With Luwrain, vendors can do this without any prior experience in creating accessible tools, because it just takes already prepared Java classes. Luwrain not only promises that it will be understandable for blind users, but in addition, it offers a platform for launching a newly created application, which is free of charge, eliminating barriers to software costs.■

Michael Pozhidaev has been dreaming of changing the ways of accessible interaction for people with vision problems for more than ten years. Michael is 32 years old, has a PhD in computer sciences and is employed as a software architect at Electronic Moscow. He loves piano music and goes crazy when he gets a chance to take a flight on a big jet or visit a new airport. He welcomes messages sent to michael.pozhidaev@gmail.com.

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



DOC SEARLS

Privacy Is Personal

It's a matter of agency and scale—for individuals, not for organizations.

Try to nail two boards together with your bare hands.

It can't be done. You need a hammer. But, the power is not the hammer's. It's yours, because the hammer is your tool. As a tool, it becomes part of you. That's what tools do: they enlarge your capacity for action and effect.

That capacity is called *agency*. To have agency is to operate with effect in the world. The range of that effect expands with the number and quality of our tools and our expertise in using them.

This range is called *scale*, and it operates at two levels. The first is personal. The best tools work for many purposes in many places. The hammer I use in Wellington, New Zealand (where I am now), works the same everywhere in the world I want to hammer nails through boards. The

second is social. Hammers are familiar tools that lots of people everywhere can use in lots of different ways.

Organizations want scale too. Every new company these days talks about "scaling up". But personal scale is different. It's about expanding our capacities outward, beyond our bodies. We get scale as drivers when we speak about "my engine" and "my wheels", because our senses extend through the whole car. And we get scale socially by being many drivers of many cars on roads everywhere.

Now back to the two boards. Say one is the Net, and the other is a company you want to connect with through the Net. Your main hammer is a browser. What's your nail?

Well, there's the company's Web site, which has a login system, a page where you can manage your account with it, and maybe a phone number

And the problem gets worse with every new company you deal with, because all of them require separate, silo'd "relationships".

or a chat thing so you can talk to a company agent. But none of those are yours. Those are tools that give the company scale, not you.

Worse, every company has its own tools for nailing you to its system. And those are different too, for every company. Even if two companies use the same back-end CRM (customer relationship management) systems (for example, Salesforce's or Oracle's), they use those services in different ways. So as a customer, you need to deal with those companies separately, inside their systems. So while *their* tools scale across many customers, *yours* don't scale across many companies. And the problem gets worse with every new company you deal with, because all of them require separate, silo'd "relationships".

To illustrate the difference between your agency and theirs, imagine telling every company you deal with that you have changed your address, or your phone number, or your last name—*in one move*. You can't, any more than you can push a nail through a board with your bare

hands. Instead, you have to go to every company's Web site, one at a time, log in and go through the gauntlet of requirements.

Now look at the same challenge from a company side. If it wants to tell every one of its customers about its new name, address or phone number, it can do that in one move, because it has tools for that. You don't—not yet, and not as long as you are the client and it is the server. Every server is a castle and every client is a serf.

This kind of scale asymmetry has been with us ever since industry won the Industrial Revolution. That victory brought mass manufacture and mass marketing—both are kinds of scale—to business. Henry Ford said: "Any color you want, as long as it's black." It's likewise with one-sided "agreements" that companies impose on every customer and user.

In 1943, Friedrich Kessler, a law professor at Columbia, observed that *freedom of contract*, a feature of civilization for centuries (if not millennia), was abandoned by big business in the Industrial Age, for the

We're still in Eden, writing the Net's Genesis while we walk around naked.

sake of scale:

The development of large scale enterprise with its mass production and mass distribution made a new type of contract inevitable—the standardized mass contract. A standardized contract, once its contents have been formulated by a business firm, is used in every bargain dealing with the same product or service. The individuality of the parties which so frequently gave color to the old type contract has disappeared. The stereotyped contract of today reflects the impersonality of the market....

He called these *contracts of adhesion*. With contracts of adhesion, the controlling party is held to terms by velcro, while the controlled party is held by cement. Agency for the controlling party is huge. For the controlled party, it is limited to yes or no. The box you click says "accept", not "negotiate".

Kessler despaired that freedom of contract would never again operate in the Industrial world. But that

was before the Internet introduced new conditions to that world—ones hospitable to countless new tools that would give every individual new forms of agency with global reach.

The Internet does many things, but the most profound is giving every end point the same status, and reducing the functional distance between end points to zero—or close enough. It's the same with cost. The protocols that govern the Net cost nothing and were not designed to support billing.

Linux would not be here without the Net. Nor would countless other building materials and methods that support networked life and the institutions that rely on networks, which now include approximately everything.

But the Net is new. Only 20 years have passed since April 1995, when the NSFNET shut down and commercial activity on the Net could begin. In the history of civilization, or even of business, that's nothing. We're still in Eden, writing the Net's Genesis while we walk around naked. Only our wizards have a bit of clothing and shelter, thanks to

Advertiser Index

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

ADVERTISER	URL	PAGE #
AnDevCon	http://www.AnDevCon.com/	19
ContainerCon	http://events.linuxfoundation.org/events/containercon	27
Drupalize.me	http://www.drupalize.me	79
EmperorLinux	http://www.emperорlinux.com	67
O'Reilly OSCON	http://www.oscon.com/open-source-2015	15
Peer 1	http://go.peer1.com/linux	69
SUSE	http://suse.com	3
Usenix Security	www.usenix.org/sec15	7

ATTENTION ADVERTISERS

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

PKI, crypto and such, but as a species we're still bare on all sides.

Meanwhile, machine intelligence in the hands of giants is giving them more scale and us less. In "Be the friction—Our Response to the New Lords of the Ring", Shoshana Zuboff starts with this subhead: "A new social logic is taking shape: It's all about surveillance. The individual is used as a mere provider of data. It's time to break the arrogance of Silicon Valley." Below, she summarizes three laws she issued in the 1980s:

First, that everything that can be automated will be automated. Second, that everything that can be informatized will be informatized. And most important to us now, the third law: In the absence of countervailing restrictions and sanctions, every digital application that can be used for surveillance and control will be used for surveillance and control, irrespective of its originating intention.

Yet surveillance at scale is also delusional. For example, in an interview a few years ago, Google Chairman Eric Schmidt said, "If you have something that you don't want anyone to know, maybe you

shouldn't be doing it in the first place." Yet if that were true, few of us would be here. (Just the ones, I suppose, conceived by acts of public copulation.)

Eric's case is a clear example of the fallacious "nothing to hide" argument in privacy debates. All of us have something to hide, or we wouldn't wear clothes—to cover, among other things, what we call our "privates".

Clothing and shelter are privacy technologies. They involve tools—clothing, doors, windows—that give

us agency and scale. These tools have been well developed and understood for thousands of years. Our civilization is based on those understandings.

So the real privacy challenge is simple one. We need clothing with zippers and buttons, walls with doors and locks, windows with shutters and shades—that work the same for each and all of us, to give us agency and scale.

Giants aren't going to do it for us. Nor are governments. Both can be responsive and supportive, but they can't be in charge, or that will only make us worse victims than we are already. Privacy for each of us is a personal problem on-line, and it has to be solved at the personal level. The only corporate or "social" clothing and shelter on-line are the equivalents of prison garb and barracks.

What would our clothing and shelter be, specifically? A few come to mind.

- Ways to encrypt and selectively share personal data easily with other parties we have reason to trust.
- Ways to know the purposes to which shared data is used.
- Ways to assert terms and policies and obtain agreement with them.

LINUX JOURNAL for iPad and iPhone

Available
on the
App Store



<http://www.linuxjournal.com/ios>

- Ways to assert and maintain sovereign identities for ourselves, and manage our many personal identifiers—and to operate anonymously by default with those who don't yet know us. (Yes, administrative identifiers are requirements of civilization, but they are not who we really are, and we all know that.)
- Ways to know and protect ourselves from unwelcome intrusion in our personal spaces.

All these things need to be as casual and easily understood as clothing and

shelter are in the physical world today. They can't work only for wizards. Privacy is for muggles too. Without agency and scale for muggles, the Net will remain the Land of Giants, who regard us all as serfs by default.■

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

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

Resources

Friedrich Kessler: http://en.wikipedia.org/wiki/Friedrich_Kessler

“Contracts of Adhesion—Some Thoughts About Freedom of Contract” by Friedrich Kessler: http://digitalcommons.law.yale.edu/cgi/viewcontent.cgi?article=3728&context=fss_papers

National Science Foundation Network (NSFNET):
http://en.wikipedia.org/wiki/National_Science_Foundation_Network

Switch to Linux: <https://www.youtube.com/watch?v=8LsxmQV8AXk>

“Be the friction—Our Response to the New Lords of the Ring” by Shoshana Zuboff: <http://www.faz.net/aktuell/feuilleton/the-surveillance-paradigm-be-the-friction-our-response-to-the-new-lords-of-the-ring-12241996.html>

“Google CEO Eric Schmidt Dismisses the Importance of Privacy” by Richard Esguerra: <https://www.eff.org/deeplinks/2009/12/google-ceo-eric-schmidt-dismisses-privacy>

Nothing to hide argument: http://en.wikipedia.org/wiki/Nothing_to_hide_argument

“Thoughts on privacy” by Doc Searls:
<http://blogs.law.harvard.edu/doc/2013/08/31/thoughts-on-privacy>