LINUX JOURNAL™
Since 1994: The Original Magazine of the Linux Community
SEPTEMBER 2013 | ISSUE 233 | www.linuxjournal.com

A LOOK AT **20 YEARS OF WEB DEVELOPMENT**

HOW-TOs

COMMAND-LINE TRICKS AND TIPS

TECHNIQUES FOR HARD DRIVE CACHING

USE VCSH TO MANAGE YOUR CONFIGS

REVIEWED:
ACER C7 CHROMEBOOK

gcalcli
THE CLOUD ON THE COMMAND LINE

SciDAVis
FOR YOUR DATA ANALYSIS TASKS

Attend the Largest Dedicated Android Conference in the Universe!

AnDevCon SAN FRANCISCO

November 12-15, 2013

Registration Now Open!

Get the best real-world Android developer training anywhere!

- Choose from more than 75 classes and tutorials
- Network with speakers and other Android developers
- Check out more than 40 exhibiting companies

"AnDevCon is a great opportunity to take your Android skills to the next level, get exposed to technologies you haven't touched yet, and network with some of the best Android developers in the world."
—Joe Mitchell, Software Engineer, Quicken Loans

"It's a blast learning and exchanging ideas with phenomenal speakers and cutting-edge experts who have the experience."
—Brad Holmes, Software Developer, uShip

Register Early and Save at www.AnDevCon.com

A BZ Media Event Follow us: twitter.com/AnDevCon

Accelerate Your App

Are you tired of watching a sad workstation chug through your data?

Sign up for a test drive on one of our GPU server solutions today. See how you can accelerate your code or applications with parallel processing on NVIDIA® Tesla® K20 GPUs.

http://www.siliconmechanics.com/testdrive

NVIDIA

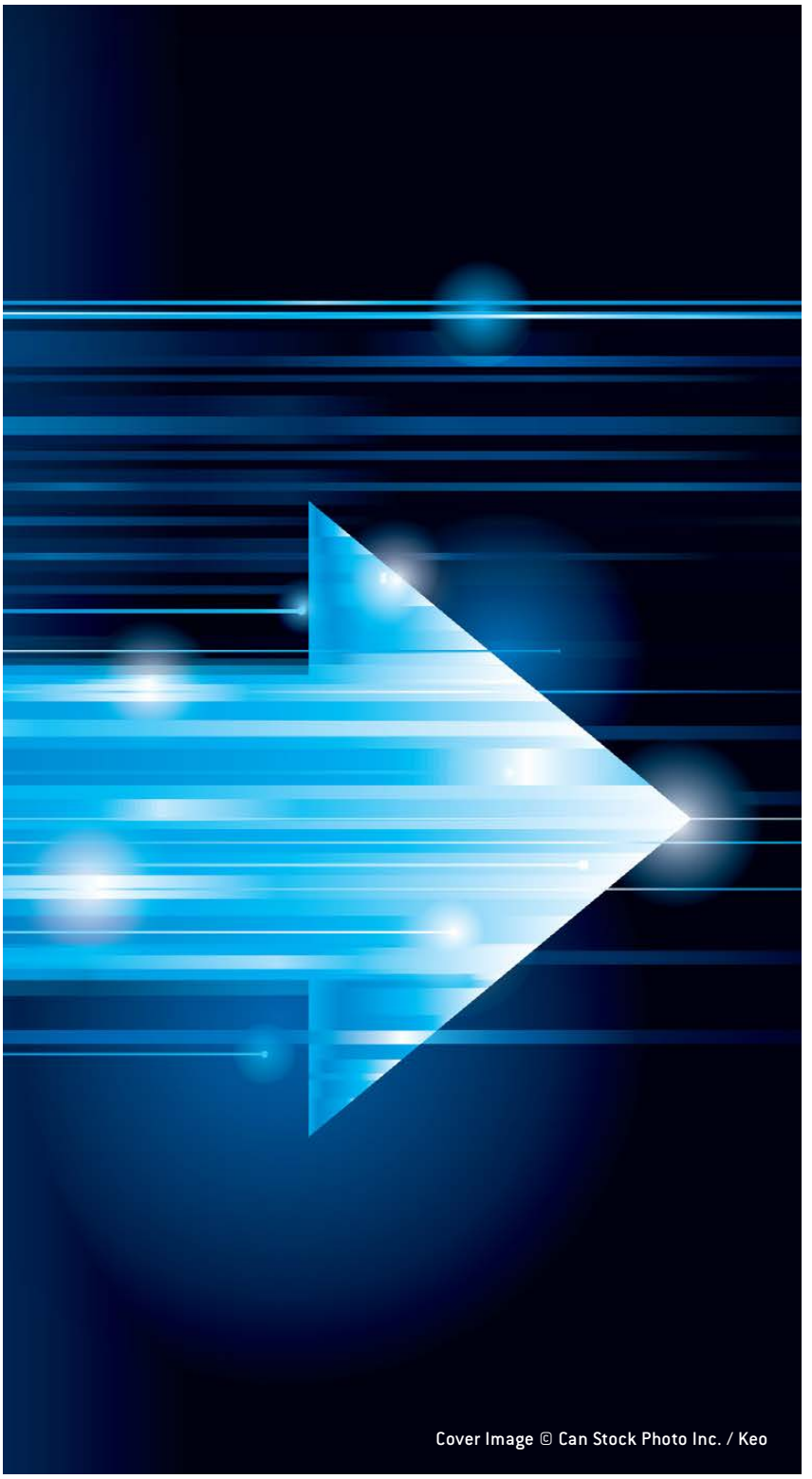
SILICON MECHANICS
Expert included.

CONTENTS

SEPTEMBER 2013
ISSUE 233

HOW-TOs

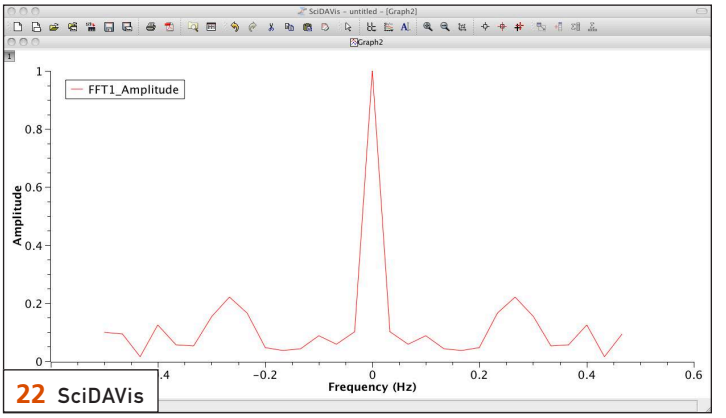
FEATURES



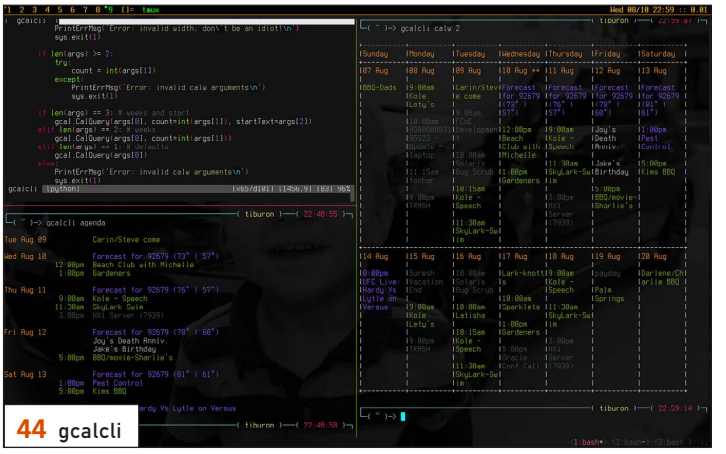
Cover Image © Can Stock Photo Inc. / Keo

ON THE COVER

COLUMNS



22 SciDAVis


44 gcalcli

REVIEW


66 ACER C7 CHROMEBOOK

IN EVERY ISSUE

LINUX JOURNAL (ISSN 1075-3583) is published monthly by Belltown Media, Inc., 2121 Sage Road, Ste. 395, Houston, TX 77056 USA. Subscription rate is $29.50/year. Subscriptions start with the next issue.

LINUX JOURNAL

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

Contributing Editors

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

Publisher	Carlie Fairchild publisher@linuxjournal.com
Director of Sales	John Grogan john@linuxjournal.com
Associate Publisher	Mark Irgang mark@linuxjournal.com
Webmistress	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

High Performance, High Density Servers for Data Center, Virtualization, & HPC

E5-2600

HIGH & Throughput INCREDIBLE Performance Density

On-board 10 Gigabit Ethernet and Infiniband for greater throughput in less rack space

The Intel® Xeon® Processor E5-2600 family powers the highest-density servers iXsystems has to offer. The iXR-1204+10G features dual onboard 10GigE + dual onboard 1GigE network controllers, up to 768GB of RAM and dual Intel® Xeon® E5-2600 family processors, freeing up critical expansion card space for application-specific hardware. The uncompromised performance and flexibility of the iXR-1204+10G makes it suitable for clustering, high-traffic webservers, virtualization, and cloud computing applications - anywhere you need the most resources available.

IXR-1204+10G: **10GbE On-Board**

For even greater performance density, the iXR-22X4IB squeezes four server nodes into two units of rack space, each with dual Intel® Xeon® E5-2600 Family Processors, up to 256GB of RAM, and an on-board Mellanox® ConnectX QDR 40Gbp/s Infiniband w/QSFP Connector. The iXR-22X4IB is perfect for high-powered computing, virtualization, or business intelligence applications that require the computing power of the Intel® Xeon® Processor E5-2600 family and the high throughput of Infiniband.

4 Server Nodes in 2U

IXR-22X4IB

iXR-1204+10G

- Dual Intel® Xeon® Processors E5-2600 Family
- Intel® X540 Dual-Port 10 Gigabit Ethernet Controllers
- Up to 16 Cores and 32 process threads
- Up to 768GB Main Memory
- 700W Redundant high-efficiency power supply

iXR-22X4IB

- Dual Intel® Xeon® Processors E5-2600 Family per node
- Mellanox® ConnectX QDR 40Gbp/s Infiniband w/QSFP Connector per node
- Four server nodes in 2U of rack space
- Up to 256GB Main Memory per server node
- Shared 1620W Redundant high-efficiency Platinum level (91%+) power supply

ixsystems®

intel® inside™ Xeon®

Call iXsystems toll free or visit our website today! **1-855-GREP-4-IX | www.iXsystems.com**

SHAWN POWERS

How'd Ya Do That?

I tend to read science fiction or fantasy for entertainment and/or escape from reality. In fact, I suspect my childhood would have been far less tolerable had I not been able to escape to Mordor for a time or hide from the bugs with Johnnie Rico. And as much as I loved watching the exploits of Captain Picard and his crew, there was never an episode on how to build a replicator. Or a holodeck. Or a phaser. And that's what I've loved about *Linux Journal* for far longer than I've been on staff: it shows how to do stuff!

This issue not only continues that legacy, but it also even focuses on it! I apologize in advance for any lost productivity at work while you live out the How-To issue this month.

Reuven M. Lerner starts out with a look back at 20 years of Web development. It's hard to believe it's been 20 years since the first www was put in a browser's address bar, but what a 20 years it's been! No more blink tags and far fewer animated GIFs make the Web a lot more fun, even if it is still "Under Construction".

Necessity truly is the mother of invention, and Dave Taylor knows that better than anyone. As he continues to battle with a DDOS on his server, he shares his process with us all. While scripting is the Swiss Army knife of system administration, it's not terribly helpful if your scripts have a "dull blade". Dave shows some best practices on creating scripts that can provide invaluable information.

Kyle Rankin reveals that he's really been lying all these years, and that he uses Windows Vista on a 17" sports utility laptop. Okay, that's a complete lie—I couldn't resist, sorry Kyle. In true Kyle Rankin form, he describes how to use and manipulate a command-line calendaring app, gcalcli. If you want to use Google Calendar, but don't want to load up that pesky browser, Kyle's article is for you.

Rather than teach how to do

something this month, I took the time in my column to show you how I do things. I get lots of e-mail asking what sort of hardware and software I use, so I figured the How-To issue was a good time to spill the beans. Some of my setup is probably not surprising, but part of it might be. I'm looking forward to feedback and to seeing what everyone else uses.

Next up, Peter Cook provides a review and tweak guide for the Acer C7 Chromebook. While targeting a similar market to the ARM-based Samsung Chromebook Bill Childers reviewed recently, the C7 uses the Intel architecture, and sports a full hard drive. It's upgradeable, and after reading Peter's article, you'll see it's quite customizable.

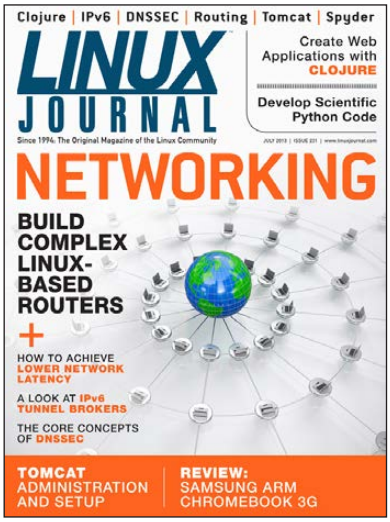
Janos Gyerik follows with an excellent article on command-line tricks. I've been using Linux for almost two decades, and I learned a lot from Janos' cornucopia of CLI tricks. Somewhere, Kyle Rankin has to be proud. I also learned some cool things from Petros Koutoupis this month with his article on hard drive caching. Hard drives are slow, and even the fastest spinning media is the bottleneck in any system. By caching to RAM or SSD, Petros explains how to take advantage of Linux's powerful caching abilities.

Bill Childers finishes up our How-To articles with his tutorial on setting up vcsh for managing configuration files. If you've ever tar'd up your /etc directory, and called that good enough, you won't want to miss Bill's article. In fact, most articles in this issue start with the premise of, "you used to do it this way, but you should try this!" That's part of the reason I love the Linux community so much. The old saying "if it ain't broke, don't fix it" is all well and good, but if you're a Linux user, a better saying might be, "if it ain't broke, good, that means we can work on making it better!"

This issue showcases what made me a *Linux Journal* reader years before I was on staff. I love to read about stuff, but nothing is quite as exciting as getting to do that stuff yourself. I hope you enjoy this issue as much as we've enjoyed putting it together. Now if you'll excuse me, I'm going to use my replicator and make some Earl Grey, hot. Beam on over and I'll make you a cup too.■

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

letters



Re: SIGALRM Timers and Stdin Analysis

In Dave Taylor's "SIGALRM Timers and Stdin Analysis" article in the November 2012 issue, he was using `ps` to check to see whether a process was running before sending a signal for termination, as in the snippet below:

```
ps $$ > /dev/null
if [ ! $? ] ; then
  kill -ALRM $$
fi
```

As an alternative to `ps ...`, he might want to use `kill -0 $$`. The `-0` option is designed to give an exit code that shows whether a signal can be sent.

I learned this from some good folks I used to work with. Thanks for the article.
—Paul

Dave Taylor replies: *Nice solution, Paul. Thanks for sharing it!*

My Facebook

I have been doing some research on Facebook and found that it can be run on CentOS with PHP.

That said, I thought I would start my own Facebook, and I saw this and thought you might like to do an article on it: **http://elgg.org**.
—Jeffrey Meyer

Fascinating, Jeffrey. Perhaps individual social-networking platforms are the next big thing. With a common back end for users (like gravatar even), I could see it working.—Shawn Powers

Why I Let My Subscription Lapse and Won't Ever Re-subscribe

It took my desktop computer longer than 0 seconds to render a page with an ad—closer to a minute, I think.

The opportunity cost to read your magazine in PDF is too high, even if it is free.
—Andrew Snyder

Sorry you're having a difficult time with ads, Andrew. Although advertising "keeps the lights on", we try to select only vendors that will be of interest to our readers.—Ed.

I Will Renew

Thanks to Shawn Powers for the humor and info. I've been a reader since March 1994, and I will continue with the digital subscription in no small part due to his tongue-in-cheek approach.
—William Weiss

Thanks William! I have no idea what you're talking about, however. I totally want a cybernetic implant in my brain.—Shawn Powers

I Like the Way You Write

Shawn, I'm writing regular tips-and-tricks articles for Linux, etc., for a blog (**http://tipstricks.itmatrix.eu**). I do it the very dry way, and after reading Shawn Powers' "Taming Tomcat" article in the July 2013 issue, I was delighted by the simple and nice way he wrote it.

Thanks for the contribution. If you ever want to write on any of the subjects I wrote about in the blog, be my guest. I have put no restrictions on the use of this material.
—Michel Bisson

Aw, shucks. It's the month for making me feel shiny about my writing, I guess. It was my birthday recently, so I'll take your letter as an awesome gift. Thank you very much.—Shawn Powers

Sleep Patterns

The graph of Shawn Powers' sleep patterns looks a lot like mine did, until I realized that I probably entrain to artificial light [see Shawn's "Sleep as Android" article in the Upfront section of the July 2013 issue]. In other words, my brain misinterprets artificial light

as sunlight, so it doesn't think it's nighttime even after the sun has set, and so it doesn't prepare my body's physiology for sleep. Those changes take a while (an hour or two), so when I turned out the lights, I tossed and turned for an hour or two until my body adjusted to the darkness.

As it turns out, this is all due to the presence of retinal neurons that function to detect a narrow range of blue light (sky blue, in fact). Their job is to inform the brain when they no longer detect that light (that is, when night has fallen). However, in some individuals, these cells see enough blue light in the artificial light spectrum to fool them into thinking the sun is still up (computer screens and TVs emit *a lot* of light in this blue range). I'm not sure why this happens in some individuals and not others, but I wonder if it might have to do with eye color (I have blue eyes).

In any event, it's easy to use glasses to filter out this narrow range of blue light, thereby plunging the relevant retinal neurons into "darkness". Individuals with this problem would don the glasses for 1–2 hours before they wish to retire; after a couple hours, sleep then (in my experience) comes easily and naturally, and much less fitfully.

I bought a pair of these glasses from **http://www.lowbluelights.com**, and my sleep patterns have improved enormously. The effects have lasted long enough (about a year) to rule out any significant placebo effect. You might want to give them a try (I have no association whatsoever with this company, except as a satisfied customer).

—Chris Brandon

Whoa, that's fascinating. I have bluish-grey eyes as well. (Again, this might be anecdotal, but it's still interesting.) I'll have to try that out. Thanks for the info!—Shawn Powers

DevOps

Tracy Ragan's opinion piece "21st-Century DevOps—an End to the 20th-Century Practice of Writing Static Build and Deploy Scripts" in the June 2013 issue told us repeatedly that she doesn't hold with scripts. She hints once or twice that a model-driven approach would be better; it would have been great if she'd told us how that would work!

—S.

Tracy Ragan replies: *Defining how a DevOps Model-Driven Framework is implemented is a complete article in itself that cannot be answered easily*

in a few short sentences. However, there are solid examples in both the commercial and open-source markets that have model-driven solutions that are worth looking into. Take a look at how Chef from Opscode uses "recipes" for defining standard server configurations. On the build management side, take a look at Meister from OpenMake Software and how it uses "Build Services" for creating standard models for compiling more than 200 different languages. In the deployment world, CA Release Automation (previously Nolio) uses standard models for performing application-level deployments, similar to IBM's Rational Application Framework for managing Websphere deploys.

In essence, to deliver a Model-Driven Framework, you establish solutions and processes that can separate the build, test or deploy logic from the "hard-coded" references to filenames, directories and so on. That information is delivered through manifest files, target files or other containers, which are then passed into the logic. In other words, the "how" and "what" are kept apart until an action needs to occur. You may end up with many files containing the "what", but only a few containers that include the "how". The logic thereby becomes reusable and easily managed. By having only a few containers that manage the "how", you end up with a highly repeatable and transparent process for moving code from development to production with the ability of reporting on details that scripts cannot produce. Thanks for the feedback.

Electronic Vs. Paper

So, it has been some time since you guys have done a paper copy of *LJ*.

Up until the time you stopped printing *LJ*, I had read every issue since issue 1, usually within a month or two of it coming out.

Now that you have gone digital, I am now well over a year behind. So, clearly reading stuff on a tablet or otherwise for me doesn't work. I do it for smaller articles, but not for something the size of *LJ*.

You seriously need to consider a paper copy again—there are those of us who would happily pay more for a paper copy. You could do it in a limited run at a break-even or slightly higher cost production.

Nuts and Volts magazine has figured out how to do this, so why can't you?
—Jeff Regan

It hasn't come up in a while, Jeff, but thanks for letting us know your struggle. I'm not sure how Nuts and Volts *does it, but it's something we can look into for sure. I do worry the price might be painful, but some folks might be willing to pay the premium I suppose. At the very least, it might be worth looking into print-on-demand options. I will do some research and see what I can come up with.—Ed.*

Using a Chromebook for Development

I just got an Acer C7 Chromebook as a replacement for a laptop that recently decided to work no more.

It has great hardware specs for its price with no Microsoft tax, but unfortunately, the software is not suitable for development. The first thing I did was install Crouton to install Ubuntu. Great! But several packages are needed to be installed to make it a development machine.

Later I found Nitrous. It provides you a virtual 8-core Linux box running Ubuntu 12.04.2 LTS and your choice of development languages and frameworks (RoR, Python, DJango, Ruby and node.js at the moment). You also can connect your box to GitHub. Best of all, besides being free, you get a decent Web IDE. Shell access also is provided either by SSH or directly with the IDE.

You can join Nitrous for free at **https://www.nitrous.io**.

Keep up the good work!
—jschiavon

Cool! Moving the development side of things to the Web is a step that I haven't considered, nor have I seen it done before. Thanks for the great tip. Perhaps my wife will let me get the Chromebook Pixel now—you know, for research purposes!—Shawn Powers

Variant of Shell Timeout

Something I had to work out was having a way to timeout a command executed within a shell script (not the whole shell script itself). I had the case where an `ssh` to a box would connect, but the command never ran because of a problem on the box. It just hung my whole script. So, I wrapped the `ssh` in this function that allows me to kill the `ssh` if it goes too long:

```
function run_with_command () {

   CMD=$1

   TIMEOUT=$2

   COUNTER=0
```

```
    ${CMD} &
    CMD_PID=$!

    while ps $CMD_PID > /dev/null && [ $COUNTER -lt $TIMEOUT ]; do
        sleep 1
        COUNTER=$((COUNTER+1))
    done

    if [ $COUNTER -eq $TIMEOUT ]; then
        kill $CMD_PID 2>/dev/null
    fi

    wait $CMD_PID 2>/dev/null
}

# TEST
run_with_command "sleep 20"  10     # this will timeout
run_with_command "sleep 10"  20     # this will not timeout

# If I want the result from the command, then I do this
result=$(run_with_command "ssh box1 hostname" 10 )
```

The wait makes sure the return code of the function is provided to know if it ran successfully. If it was killed, it will return 143. If it ran, I'll get 0. **—Mark**

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!

diff -u
WHAT'S NEW IN KERNEL DEVELOPMENT

Linus Torvalds doesn't just make whatever changes he wants to the kernel. Sometimes he goes through the normal procedure of sending code to the relevant maintainers. Recently he sent a patch to **Al Viro**, trying to speed up the **VFS** by migrating some data out of **SELinux** and into the **inode data structure**.

Linus also went on a brief tirade about security features, and how some of them ended up slowing down performance by 50%, before he and Al fixed it. He and **Casey Schaufler** (the **Smack** maintainer) had a bit of a dispute over that. Casey said security folks cared a lot about performance and had to deal with hostile tirades like Linus' all the time, and so were motivated to do better.

Once upon a time, **32-bit kernels** were so cool. Nowadays, they are scorned with derisive sneers. Linus Torvalds recently remarked, "anybody who runs a 32-bit kernel with 16GB of RAM doesn't even understand *how* flawed and stupid that is."

However, some popular distributions still ship with 32-bit kernels by default, and **Pierre-Loup A. Griffais** recently pointed out that this was causing huge slowdowns for unsuspecting users.

The problem is bad enough that **Rik van Riel** suggested giving 32-bit kernels a hard limit of 8 or 12GB of RAM, regardless of how much physical RAM was present on a given system. He added that the kernel also should print a friendly warning "to inform users they should upgrade to a 64-bit kernel to enjoy the use of all of their memory."

Regardless of this patch, the problem with distributions may persist. As **H. Peter Anvin** put it, "We kernel guys have been asking the distros to ship 64-bit kernels even in their 32-bit distros for many years."

Linus Torvalds went on vacation and temporarily lost the ability to make **tarball** releases of new kernel versions. He updated his git tree, but he apologized for the lack

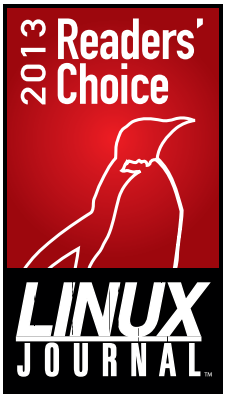
of tarballs. He also speculated that maybe nobody even used the tarballs anymore, now that git existed.

However, **Randy Dunlap** and **Mikael Pettersson** immediately replied that they did indeed rely on the tarball method. And, Linus reassured them that he'd keep putting out tarballs for the moment.

Nathan Zimmer posted a patch to parallelize **RAM initialization**. Typically during bootup, RAM is initialized by a single CPU, before any other CPUs are started up. On Nathan's system, this caused boot times of more than an hour. With his patch, the CPUs would all be brought up first, and then RAM could be initialized by all of them together, over much less time.

Originally, he made it a configuration option, but **Greg Kroah-Hartman** said there was no reason for that—just make it an intrinsic feature. Clearly no one would choose to configure a slower boot time.—**ZACK BROWN**

2013 Readers' Choice Awards



Don't forget to participate in the 2013 Readers' Choice Awards!

Cast your votes at http://www.linuxjournal.com/rc13.

Voting ends September 22, 2013. The results will be published in the December 2013 issue of *LJ*.

They Said It

I'd rather work with someone who's good at their job but doesn't like me, than someone who likes me but is a ninny.
—Sam Donaldson

Better be despised for too anxious apprehensions, than ruined by too confident security.
—Edmund Burke

Age is no guarantee of maturity.
—Lawana Blackwell

Eighty percent of success is showing up.
—Woody Allen

When you can't have what you want, it's time to start wanting what you have.
—Kathleen A. Sutton

Pitch Perfect Penguins

My daughters love the movie *Pitch Perfect*. I suspect our XBMC has played it more than 100 times, and I'm not exaggerating. Whether or not you enjoy young-adult movies about singing competitions and cartoon-like projectile vomiting, I'll admit it's a pretty fun movie. The question my girls ask me most often is about the audio-mixing software the protagonist uses to make her "sick beats".

Although Anna Kendrick uses a Macintosh in the movie, and the software's name is obscured or changed in close-ups, it's easy to tell the software she uses is Mixxx. The cool part for me is that while indeed Mixxx works on OS X, it's an open-source application that works natively in Linux as well. That means, in the eyes of my daughters, Linux is now super-cool.

Mixxx is an editing package with incredible visual tools (**http://www.mixxx.org**). It's designed to "mix" several songs or rhythm tracks and export a completed song. The very nice GUI makes mixing audio easy to understand, and the interface itself is apparently cool enough for Hollywood. So whether you want to make your own "wicked tracks" or join an a cappella group, Mixxx is just the tool for you. Also, I recommend watching *Pitch Perfect*, one or two of those XBMC viewings might have been my own!

—SHAWN POWERS

Beyond Google Reader: CommaFeed

Now that Google Reader is officially gone, most folks have settled on a replacement of some sort. In fact, a few months ago I even went through the process of installing Tiny Tiny RSS as a viable and powerful replacement. At the time, there was only one feature I sorely missed, the "next unread blog" link. Approximately three days before Google Reader shut down for good, I found the holy grail of RSS readers: CommaFeed.

CommaFeed is an open-source project written in Java. It's offered as a free Web-based solution at **https://www.commafeed.com**. Although the interface is similar to Google Reader, it feels slightly stripped down. Thankfully, it provides a bookmarkable link that will take you to your next unread RSS entry. That feature is the single most important, and difficult to find, RSS reader feature I need.

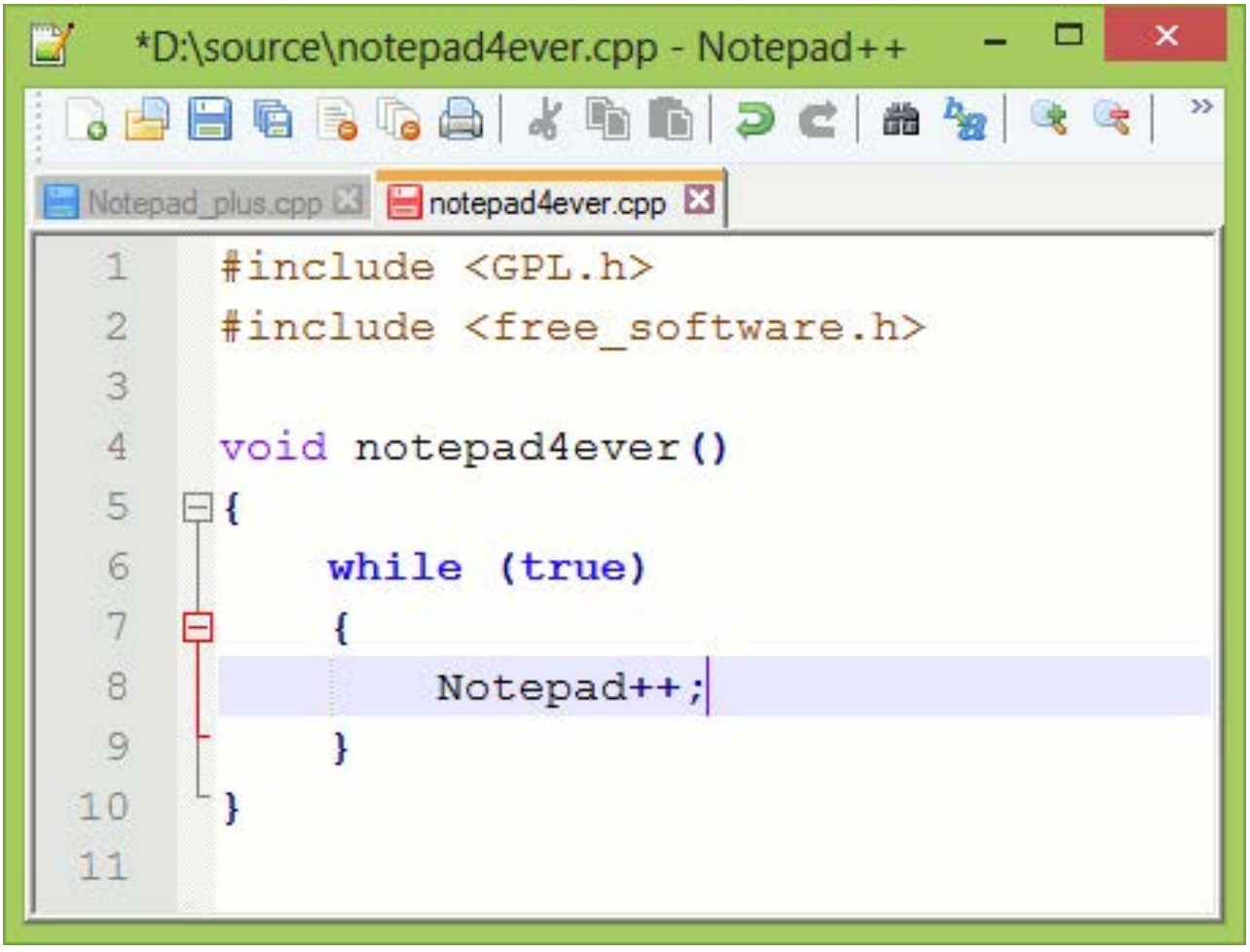
CommaFeed unfortunately doesn't have a really good mobile interface, and it's lacking features in its Android app, but improvements are being made on both fronts. I must admit, however, that even more important than the "next unread blog" link feature, is the ability to download the source code from GitHub and compile CommaFeed for self-hosting (**https://github.com/Athou/commafeed**). I don't ever want to get "Google Reader'ed" again. Hosting is complicated, because it's a Java application, but the instructions on the GitHub site make it fairly painless. I recommend trying out **https://www.commafeed.com** before compiling and self-hosting, because CommaFeed's interface might not be for everyone.—**SHAWN POWERS**

Non-Linux FOSS: Notepad++ Is Better Better

If anyone understands the importance of a good text editor, it's a Linux user stuck on Windows. Sure, Microsoft supplies Notepad and Wordpad, but neither really feels like the powerful sort of text editor a Linux user expects. Enter Notepad++.

Notepad++ provides features like line numbering, syntax highlighting and tabbed file editing. If those seem like ordinary features that should be included in any text editor worth its salt, well, you're right. Notepad++ is fully open source, and it is the preferred simple text editor on Windows. It's certainly not a full IDE, but all the developers I know have it installed if they use Windows. Give it a try at http://notepad-plus-plus.org.

—SHAWN POWERS



OVH.COM

Dedicated Server

You are in Control!

Dedicated Server KS 1

5 TB traffic

24/7 North American Support

CPU: **Intel i3 (2 Cores / 4 Threads)**
Frequency: **3.4GHz+**
RAM: **8 GB DDR3**
Hard disk: **2x 1TB SATA2**

$39 /month

Find the Entire Range:
www.ovh.com/dedicated-servers

Dedicated Servers

Dedicated Cloud

For more details:

OVH.COM
Dedicated Infrastructure for your Business

or contact us: **1-855-684-5463** (toll free)

Graphics and Stats in One Easy Package

As always in experimental science, you need to analyze the data you collect to see what it's telling you about the world. In the beginning stages, you usually need to be able to do some graphical analysis to get an overall view of any trends being revealed by your experiment. Once you get to the point of having a model you want to test, you then need to run some statistical tests to see how well your model fits the data. It always is more convenient to learn one tool rather than two. To this end, there is the package called SciDAVis (Scientific Data Analysis and Visualization, **http://scidavis.sourceforge.net**).

SciDAVis started life as a fork of QtiPlot. It has moved quite a bit from the original codebase with the addition of several new features and changes to the underlying data structures. The functionality it provides is similar to commercial programs like Origin and SigmaPlot. It also is similar to another open-source program called LabPlot. In fact, beginning in 2008, these two projects started working together on a common back end while continuing with their own front ends and their own feature sets. In this article, I take a quick look at some of the things you can do with SciDAVis for your own data analysis tasks.

First, you need to install SciDAVis. Most distributions should have a package available. In Debian-based distributions, you can install it with:

```
sudo apt-get install scidavis
```

Binaries also are available for Mac OS X. If binaries aren't available, you can download the source tarball and build it specifically for your system. You need to have the Qt libraries installed, because the interface is built using them.

When you first start SciDAVis, you get an empty project and an empty data table with two columns of numeric values (Figure 1). Selecting a column displays the details of that column in a window on the right-hand side. Selecting the description tab lets you change the name of the column, as well as

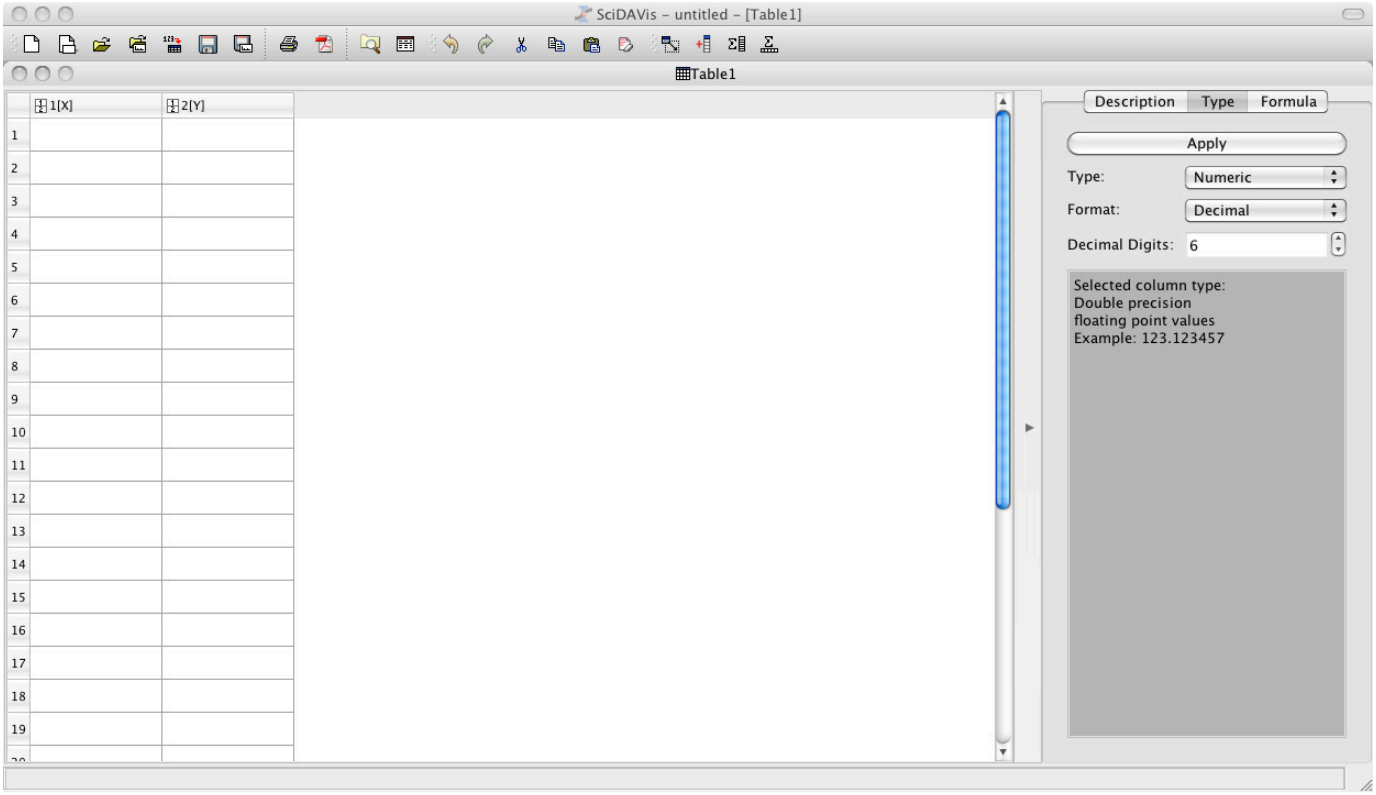


Figure 1. Opening SciDAVis gives you an empty two-column data table.

add a comment describing what the column represents. The type tab lets you change what kind of data you can enter for this column from numeric to text, month names, day names or full dates. You also can set the format of the data type for each column.

When you are first learning to use SciDAVis, you probably will just want some junk data to play with. You can do this easily by right-clicking the columns and selecting "Fill Selection with". Then, you can fill the column with either row numbers or random numbers. Once you have some data available, you can create new columns that are functions of the values stored in the other columns. To access this functionality, you need to select the formula tab in the right-hand side pane. In order to explore some of the other features, let's set the first column to be the row numbers and the second column to be a set of random numbers (Figure 2).

One of the first things you will want to do is plot the data to see what it looks like. To do a basic plot, you simply can right-click on a column and select Plot. If you

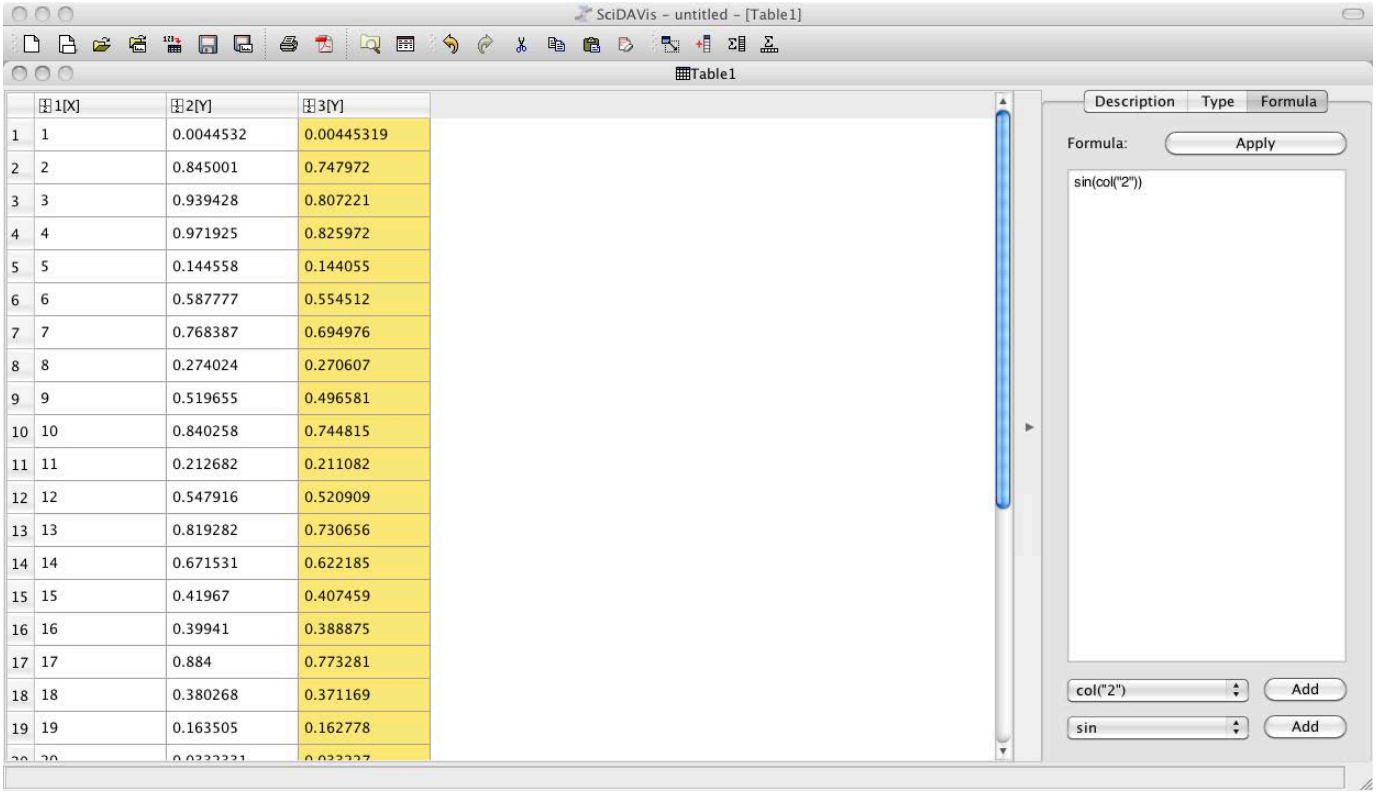


Figure 2. You can use fill functions to create data in order to try things out.

are just doing an initial look at the shape of the data, select Plot→Line or Plot→Scatter (Figure 3). The x axis is simply the index values, and the y values are the data elements from the column.

If you want to do more complicated plots, SciDAVis provides a full plotting wizard. To access the wizard, either press Ctrl-Alt-W or click View→Plot Wizard. This will pop up a new window where you can select which columns will be used for the x, y and z axes. You also have the option of selecting columns to represent the errors for the x and y axes. You need to create a new curve where you can set the relevant columns. You can create multiple curves that will all be plotted on the same graph. You then get a new window with the plots generated.

The point of SciDAVis is to make this type of work easy, so you can double-click on the various elements to edit the details of your graphs. Double-clicking on the title, or the axis labels, will pop up a window where you can change the content and the display. You also can change the details of the

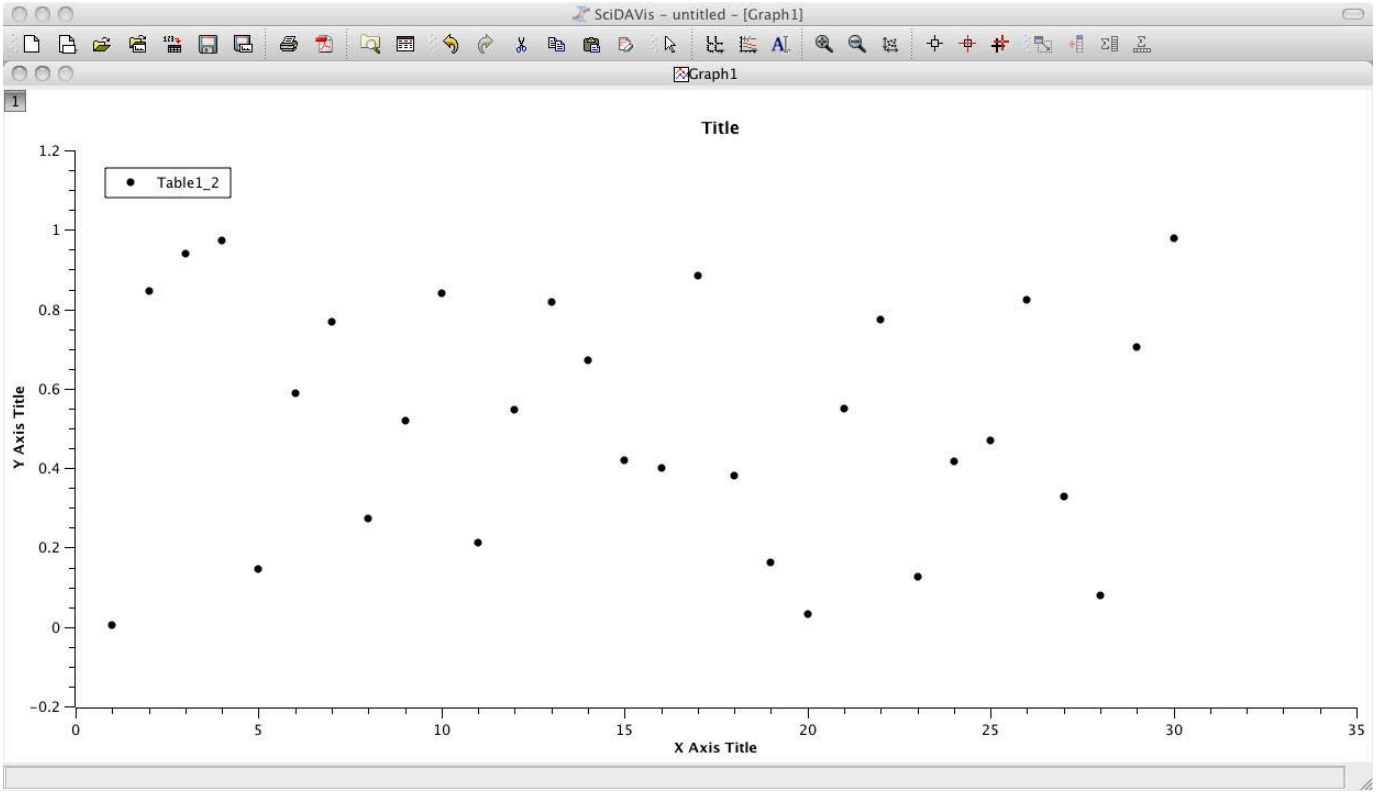


Figure 3. Scatter plots are only a couple mouse clicks away.

axes themselves. Double-clicking on an axis will pop up a new window where you can set the scale, the type of grid, the axis displays and a set of general options (Figure 4). Each of these sets of options is available under its own tab in the option window.

Once the graph looks the way you want it, you will want a copy for your publications. To do this, you either can right-click on the graph and select Export or click on the menu item File→Export Graph. Then you can select your preferred file format for saving the image.

Although graphs can be very useful when trying to get an intuitive grasp of the shape of your data, you do need to back up this intuition with hard numbers. The first thing to do, usually, is simply look at the column statistics. Right-clicking on the column, you would select Column Statistics. This creates a new table where you will get the number of rows in the column, along with other statistics like the standard deviation, variance, sum, minimum and maximum. You can see if there is any correlation between

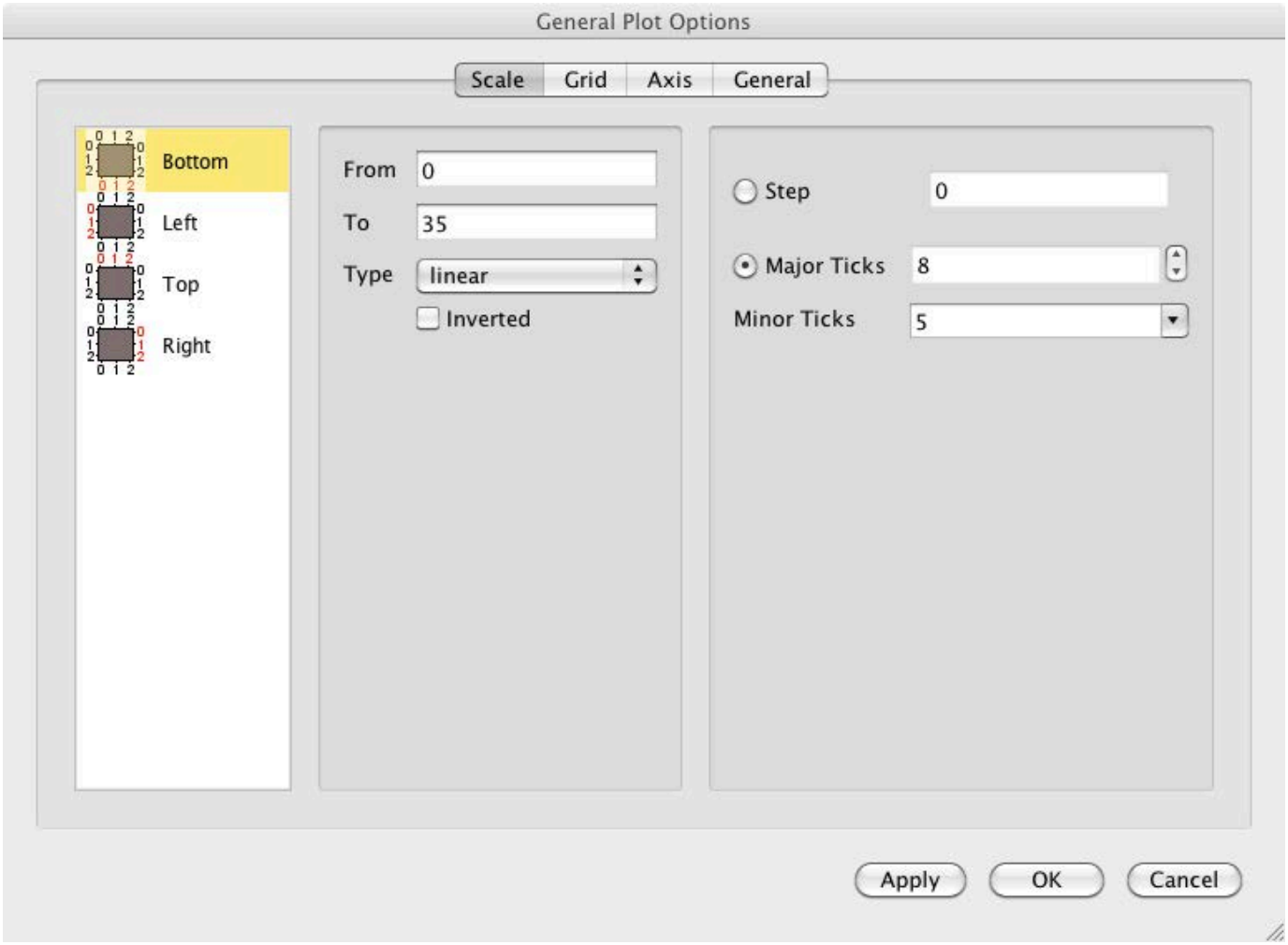


Figure 4. You can change all of the elements of your graphs.

two columns. You need to select two columns from your table, and then click on the menu item Analysis→Correlate. This will pop up a new graph window showing a picture of the correlation. Two new columns will be added to your data table where you can find the lag and correlation values of this particular analysis.

If the data you are looking at has some type of periodicity, you can calculate an FFT of it to see the spread of frequencies within your data. By selecting a column and clicking on the menu item Analysis→FFT, you will get a pop-up window where you can select the details of the FFT you want to calculate. Once these are set, click OK, and a new graph window will be displayed with the FFT plotted (Figure 5).

Once you have had a chance to

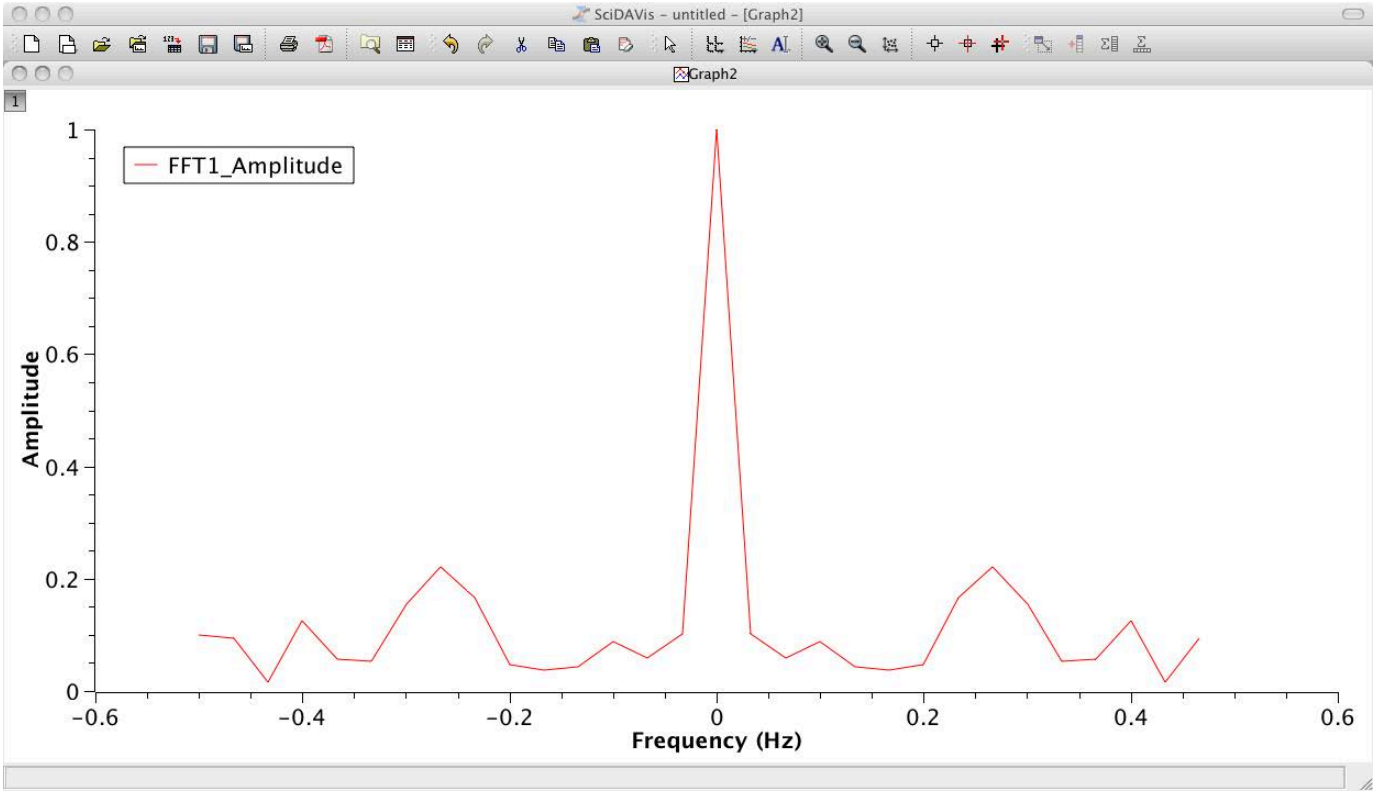


Figure 5. FFT analysis is useful in signal processing.

look at your data, you may have started to form an idea of a model that represents the system you were measuring. An idea is not enough, however. You actually need to do some calculations and see whether your model fits the data you collected. Two options are available. The first is to use the Fit Wizard. You can access it by clicking on the menu item Analysis→Fit Wizard. This pops up a window where you can build a function describing your model. Once you have built up your model, click the button called Fit. This pops up a new window where you can select the details of doing the actual fitting of the generic function to your data. Here you can set the initial guesses and select the algorithm used to do the actual fitting (Figure 6).

You also can set how many iterations to try and the tolerance of when you can stop. When everything is set to your satisfaction, click the Fit button. This pops up a new graph window, plus an output window detailing the results of trying to do the fit. The other option of trying to fit your model is to start with your

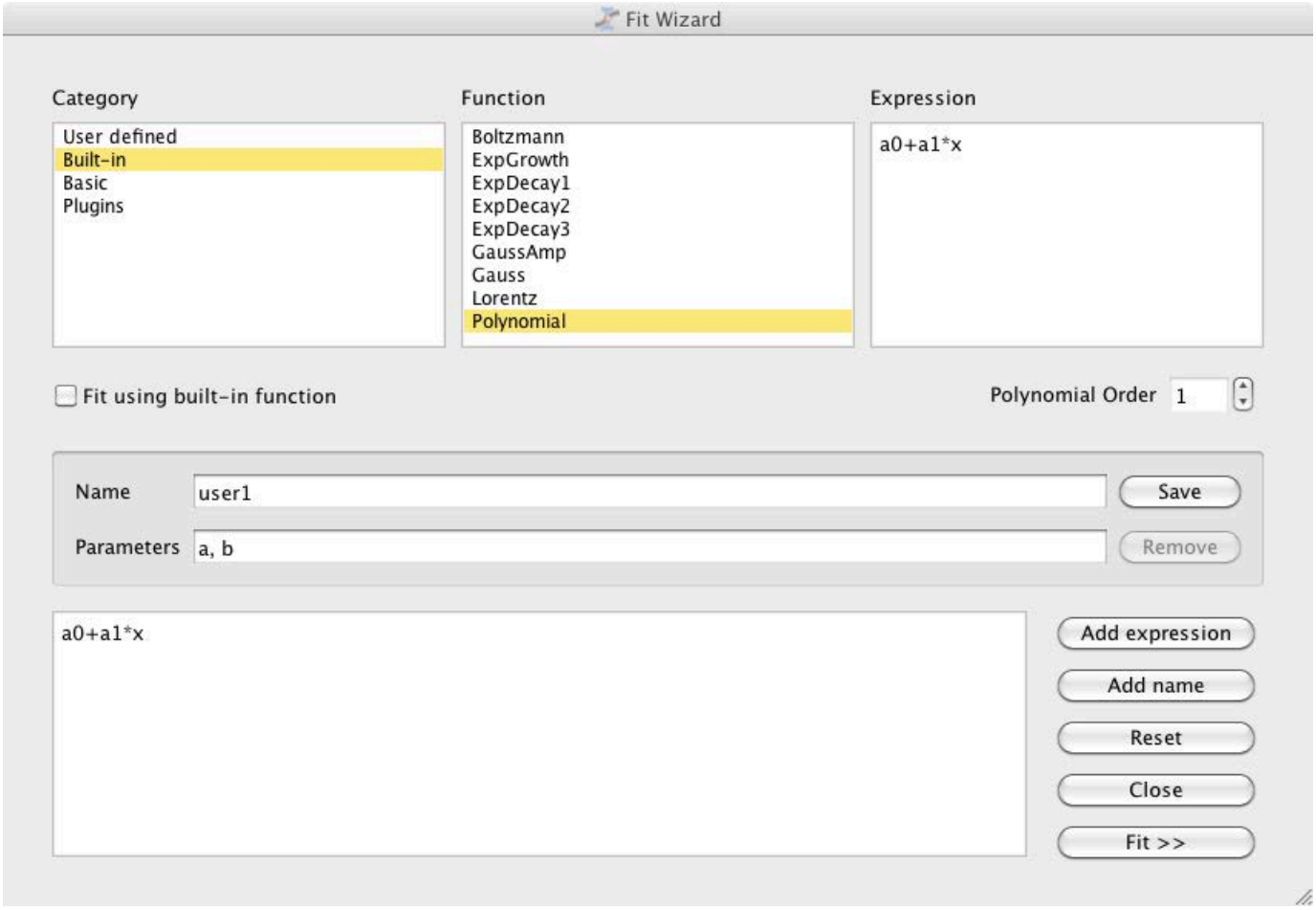


Figure 6. The Fit Wizard lets you define your model and see how well it fits your data.

initial graph. In this case, you start by right-clicking the main graph window and selecting the Analyze option. This opens a submenu where you can select one of a number of common defaults, such as linear, polynomial, exponential growth or decay, among several others. You also can open the Fit Wizard from this submenu.

This article has been only a very short introduction. Lots of other functions are available if you are willing to go through the manual. Also, you have the option of using Python as a scripting language within SciDAVis in order to do even more complicated data analysis. You might need to compile from source, however, if the binaries available to you don't include this functionality. Hopefully, you will take the time to learn what may be a very useful tool for analyzing your experimental data.

—JOEY BERNARD

1&1 DYNAMIC CLOUD SERVER

PRICE CONTROL

AS LOW AS

$0.06

PER HOUR*
PLUS $60 OFF
FIRST MONTH

✓ COMPLETE COST CONTROL

- **Full transparency** with accurate hourly billing
- **Parallels® Plesk Panel 11 included,** with unlimited domains

✓ FULL ROOT ACCESS

- The complete functionality of a root server with dedicated resources

✓ MAXIMUM FLEXIBILITY

- Configure the Processor Cores, RAM and Hard Disk Space
- Add up to 99 virtual machines

✓ FAIL-SAFE SECURITY

- Redundant storage and mirrored processing units reliably protect your server

OPTERON PROCESSOR AMD

SUSE

Parallels Plesk Panel

1&1®

MEMBER OF united internet

DOMAINS | E-MAIL | WEB HOSTING | eCOMMERCE | SERVERS

Call **1 (877) 461-2631** or buy online

1and1.com

* Other terms and conditions may apply. Visit www.1and1.com for full promotional offer details. Customer is billed monthly for minimum configuration ($0.06/ hour * 720 hours = $43.20/ month minimum). A $60 credit (valid only for the first month) will be applied to your first month of service. Program and pricing specifications and availability subject to change without notice. 1&1 and the 1&1 logo are trademarks of 1&1 Internet, all other trademarks are the property of their respective owners.

Google Drive for Linux?

For some reason, Google seems to dislike Google Drive users who prefer Linux. I find this particularly strange, since Google's Chrome OS is based on Linux. Thankfully, the folks over at Insync not only provide Linux support for Google Drive syncing, they do it with style.

Insync is a commercial, proprietary application that installs natively in Linux (**https://www.insynchq.com**). It offers selective sync, integration with several file managers and a nice tray icon showing sync activity. The coolest feature, however, is the seamless conversion from Google Docs format to LibreOffice format. You can edit your Google Drive documents with the native LibreOffice application, and then it automatically will sync to the cloud in the Web-based Google Docs format! I've tried only a few documents, but in my limited testing, the conversion and sync have been perfect.

Insync has a 15-day free trial and a one-time cost of $9.99. Packages are available for Ubuntu, Fedora, MEPIS and even an unofficial build for Arch Linux. Now that version 1.0 has been released, Insync is available for anyone to download. Due to its native Linux support and super-cool conversion/sync ability, Insync earns this month's Editors' Choice award.—**SHAWN POWERS**

The Open Source World comes to Raleigh, NC

ALL THINGS OPEN™

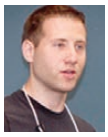
presented by IT-oLogy™

OCTOBER 23-24, 2013 | RALEIGH CONVENTION CENTER

SCHEDULED TO SPEAK

Scott Chacon
github SOCIAL CODING

Chris DiBona
Google



Chris Aniszczyk
twitter

Jessica McKellar
python

Whurley
CHAOTIC MOON STUDIOS

Andy Hunt
The Pragmatic Programmer

Angie 'Webchick' Byron
Acquia

Lee Congdon
redhat

Open Source in the Enterprise

ALLTHINGSOPEN.ORG

REUVEN M. LERNER

20 Years of Web Development

Reuven waxes nostalgic and reflects on the first two decades of Web development.

A few months ago, I was walking with my seven-year-old son, and I mentioned that when I was young, we had computers in the house. I pointed out that while I was growing up, we didn't have an Internet connection, and that the Web hadn't even been invented yet. "Really?" he asked, confused. "So what did you do with the computer?"

The Web has permeated our lives to such a large degree, it's easy to understand why my son is unable to comprehend what it would mean to be in a world without it. The notion of a computer without Internet access today is quite strange; I have given a number of lectures in military and otherwise-restricted areas where Internet access is strictly forbidden, and let me tell you, there's something accurate about the way in which Internet access is described as an addictive drug. If I can't get my fix of e-mail, or blogs, or newspapers or Hacker News, I feel cut off from the rest of the world.

This is, according to my count, my 201st At the Forge column. It is also, somewhat coincidentally, a bit more than 20 years since I started writing Web applications. So, I'd like to take the opportunity to look at where Web technologies have come from and where they're going—and to wax a bit nostalgic for the days when the Web was still the great unknown, something that we knew would one day be a part of everyone's lives, but that we couldn't express beyond those basic thoughts.

In the Beginning

In the beginning, of course, the Web was static. It took a while for people to realize that just because a Web browser was requesting a document didn't

mean that the server actually needed to send the contents of a document. Instead, the server could lie to the browser, creating a document on the fly, by executing a program. This sort of thing, which Web developers do every day, remains one of the funniest parts of the Web from my perspective—that browsers ask for the contents of a URL, as if they're asking for something static, without any way of knowing whether the server is creating something new for them or sending the same data everyone else received.

Back in those early days, it wasn't entirely obvious just what technologies people were going to use to implement their Web applications. Those of you who have seen, or used, programs in a "cgi-bin" directory might remember CGI, the API that standardized the way Web servers could invoke external programs. You might not remember that the cgi-bin directory was where the compiled C programs would go. Yes, the demonstration CGI programs that came with the NCSA HTTP server were written in C and shell. It took a little while before people discovered that higher-level languages—mostly Perl and Tcl in those days—could do the job just as easily, while allowing programmers to be far more productive.

And, although Tim Berners-Lee called it the World Wide Web, usage was far from universal in those early days. When we put MIT's student newspaper on the Web in 1993, almost no one, even among our readership, knew what a Web browser was. We had to take out advertisements in our own newspaper, telling people how to install the Mosaic browser, so they could read the newspaper on-line. And of course, the look and feel of the newspaper in those days was extremely primitive. A paper we submitted to the first-ever World Wide Web conference described how we had made it possible for people to search through our newspaper's archives—an amazing breakthrough in those days!

Although it quickly became clear that the Web was becoming dynamic, no one really thought about it as serious software development. I remember expressing surprise, in 1995, when I was given a business card that said "Web application developer" on it. While the Web was becoming dynamic, no one thought of Web development as serious software work. I snickered a bit to myself, because it seemed clear to me that applications were serious desktop software—not the toys that we were creating.

But it was in that year, 1995, that I began to understand where the Web was headed. It was then that I learned about relational databases and began

to understand just how powerful Web applications could be when connected to a powerful, flexible system for storing and retrieving data. It also was in that year that the Java and JavaScript languages were unveiled, tantalizing Web developers with the idea that we could have things happen inside the browser beyond simple text and images.

It was then that we began to imagine that the browser could be the beginning of a new platform for applications. Of course, when Marc Andreessen, then best known as author of Mosaic and a cofounder of Netscape, said things like that, we all laughed, wondering how the Web could turn into a platform for application development. But of course, that vision has turned into a reality. Desktop software hasn't disappeared entirely, but increasingly large categories of software are being pushed onto the Web, accessed through our browsers.

Libraries and Frameworks

It didn't take long before Web development really began to take off. Suddenly, the big thing was to have "a Web page" or even an entire site. The growth of the Web happened at about the same time that the term "open source" was coined, giving publicity and ammunition to the techies who wanted to rely on software such as Linux and Perl to run their systems. (It took a long time before some of my clients were convinced that running a Web server on Linux was a safe choice, or that it could handle the dozens of users who would visit their site in a given day.)

I don't think that it's a coincidence that open source and the Web grew together. Open-source operating systems and languages—in those days, Perl, Python and PHP—grew in popularity, both because they were free of charge and because they offered enormous numbers of standardized, debugged and highly useful libraries. CGI.pm became an enormously popular Perl module, because it made so much of Web development easy.

On the database side, things also were starting to improve: MySQL, which had been free of charge (but commercially licensed), was released under the GPL, and it was soon ubiquitous on the Web servers that you could license by the month or year. PostgreSQL also emerged on the scene and overcame the 8 kilobyte-per-row limit that had hobbled it for so long. Suddenly, you could get a full-fledged relational database sitting behind your application, using libraries that were fast, efficient and full of features.

Web technologies themselves were becoming more sophisticated as well. The standards for HTTP,

CSS and HTML were formalized and then grew to handle such things as caching. The W3C, formed to create and encourage standards for Web-related technologies, produced XML, a document format that remains pervasive in the computer world.

It was, on the one hand, easier to create sophisticated Web applications than ever before. But on the other hand, it was no longer possible to learn everything you needed to know about Web development in a few days or weeks. It was becoming a specialized set of skills, with people specializing in particular subsets. No longer did "Webmaster" refer to someone who knew everything about all parts of Web technologies. You didn't yet hear about "front-end" or even "full-stack" Web developers. But you did have people who were specializing in particular languages, servers, databases and even (to some degree) in Web design, helping ensure that sites were both aesthetically pleasing and user-friendly.

All of this meant that Web development was becoming serious, and the individual libraries, such as CGI.pm, were no longer enough. Web developers started to use frameworks instead of libraries. I once heard someone distinguish libraries from frameworks by saying that a library is something you add to your code, whereas a framework is something to which you add your code. Given the number of things a Web developer increasingly was having to worry about—user sessions, database connections, cookie persistence and caching—it shouldn't come as a surprise that the frameworks started to become increasingly serious, complex and sophisticated. Many (but not all) of these frameworks were released as open-source software, allowing developers to create complete, stem-to-stern Web applications using software that was inexpensive, robust and scalable, as well as written by the same people who used it.

Even Java got into the act. Although it took years for Java to be released under an open-source license, server-side Java has been around for some time and has gone through many incarnations. From the initial rollouts of servlets and JSP pages, to Java beans and Enterprise Java Beans, to the dozens of standards and open-source projects that have grown up in the Java world, the language that was originally designed to run applets in our browsers suddenly was a powerhouse to contend with.

And yet, Java, which was introduced as a simpler alternative to C++, without all the overhead and problems associated with that language, frustrated many with its large numbers of configuration

files, protocols and Java-centric words and libraries. Out of this frustration (to some degree, at least) came the first of a new generation of Web frameworks, Ruby on Rails (written in Ruby) and Django (written in Python). Both Rails and Django promised developers something that hadn't yet existed, namely the ability to write sophisticated Web applications, but without the enormous overhead and bloat that Java-based systems required. Not surprisingly, both were instant hits, and they managed to hook a new generation of Web developers, as well as pull many (myself included) away from other, older, less-sophisticated frameworks that we had been using until then.

Now I do most of my work in Ruby on Rails, and I continue to enjoy working with this framework every day. But even the biggest enthusiasts among us need to remember that Rails is nearly ten years old at this point. It hasn't reached the level of bloat and bureaucracy that we see in the Java world, but at the same time, we see that many other language communities have produced their own versions of Rails, using many of the ideas that were pioneered there. Moreover, as Rails has matured, it has needed to consider legacy software and offer developers ways to integrate Rails apps into an existing infrastructure. As a result, we've seen that the small, scrappy framework is now getting larger, a bit more bloated and certainly harder to learn than was the case when I first learned about it back in 2005.

Components, APIs and Browsers

Where does that leave us today? First, the news is all good, in that there never has been a better time to be a Web developer. The opportunities to learn, explore, extend, implement and use Web-based technologies already are overwhelming and seemingly limitless, and every day appears to bring new frameworks, libraries and techniques that make Web applications even more interactive and compelling than already was the case.

And yet, for all of the opportunities, Web development never has been more difficult to get into, simply because of the many technologies you need to master in order to create a Web application. Web development, as Philip Greenspun said so many years ago, requires that you be a generalist, understanding a little about a large number of different technologies. A modern Web developer needs to understand HTTP, HTML and CSS, at least one server-side language, SQL and JavaScript. It also helps a lot to know how to work with Linux or other server operating systems.

The reality also is that even the

most experienced Web developers are starting to specialize. True, I think of myself as what's increasingly called a "full-stack Web developer", understanding everything from server configuration to database optimization to JavaScript programming, but the march toward specialization continues, and I've seen that a growing number of jobs are aimed at people with expertise on the back end or the front end, without much overlap between the two. Configuration and deployment also are becoming their own specialties, as we've seen with the rise of devops during the last few years.

That said, if you're willing to learn new technologies and improve yourself constantly, there are limitless opportunities to do amazing things on the Web today.

Let's start with languages. It's clear that a huge proportion of Web development is being done with dynamic languages. Ruby, Python and PHP continue to dominate (outside of large corporations, at least). But we also see a large number of open-source languages based on the JVM, such as Clojure and Scala, each of which have their own Web development paradigms and frameworks. The rise of Go, a compiled, open-source language from Google, also is intriguing (and is something I hope to cover in an upcoming installment of this column). True, some languages are more popular than others, but you can find work in just about any language you want today, trying it out and learning how that language's paradigms fit into the tapestry of languages and frameworks.

Perhaps the most important language today is JavaScript. JavaScript, which never has been one of my favorite languages, now is required knowledge for anyone who works on the Web. It sits inside every browser and increasingly is used for server-side programs as well. JavaScript's ubiquity, and the fact that it is necessary for high-powered Web applications that live inside of the browser, means that such organizations as Google and Mozilla are spending enormous amounts of time, money and effort trying to get JavaScript to execute as efficiently as possible. The sophistication of browser-based programs never ceases to amaze me, and that's all due to JavaScript.

There's also a growing trend of languages that compile into JavaScript, such as CoffeeScript and ClojureScript, allowing us to write in one language but execute something else (that is, JavaScript) in the browser. And then there are the JavaScript frameworks, including Backbone.js and Angular.js, which are making it possible to create browser-based applications that not

only do amazing things, but which also are structured like the serious applications that they are. I've often said that today, developing software isn't that difficult, but maintaining software is, and thus, server-side Web developers eventually realized they needed to rely on libraries and frameworks in order not to lose their minds.

As the browser part of applications become more sophisticated, we see that server-side Web applications increasingly are becoming API servers, offering outsiders a chance to retrieve and modify data stored in a database of some sort. True, there always will be a need for server-side Web development, but it seems to me that as people learn to create better and better experiences within the browser, we're going to see the server as a remote storage mechanism, offering one of the many APIs that our browser contacts in order to bring up a fancy-looking application for our users.

Indeed, we see a trend of breaking Web applications into many little parts, each of which specializes in a different thing. You want to have real-time updates? Use Pusher or PubNub. You want to have nice-looking fonts? Use Google. You need more servers? Fire up some additional machines from Amazon or Rackspace. Or, if you want to think about your servers from an even higher level of abstraction, use Heroku, which costs more but reduces your IT staff to nearly zero. The same is true for login systems, e-commerce transactions and even commenting systems, each of which now can be outsourced, via an API and a few minutes of work, and then plugged in to your Web application. The use of APIs is particularly relevant today, when we have so many people accessing the Web not via traditional browsers, but rather via mobile devices and platform-specific apps. That app you installed might not look like it's contacting a Web site, but it's undoubtedly making a JSON API call over HTTP in the background.

In other words, we're finally seeing the start of Tim Berners-Lee's original vision, in which people all over the world are both retrieving and contributing information, no matter where they happen to be. I'm writing this from Chicago, where I'm spending the summer, but thanks to modern Web technologies, I'm able to see my family in Israel (including play Chess with my aforementioned son), order books and other merchandise to my apartment, read the news, check the weather, pay for airline and movie tickets and listen to music. And no, none of this should come as a surprise to anyone reading this magazine, or even anyone who has been using the Web at any time in the

last decade. But that's just the point. If we can step back a bit from the technologies we're using, we can see how much easier our lives have been as a result of the Web.

Web development was exciting when I started to do it 20 years ago, and the amazing thing is that it remains just as exciting today. If you've always wanted to be a Web developer, choose a language, learn how to use it to write Web applications and then see what you can do. I never cease to be amazed by the magic of programming, and the Web makes it possible to spread that magic to the entire world. Here's to the next 20 years of advances in Web technologies—and to the next 200 columns I'll write for *Linux Journal*!■

Web developer, trainer and consultant Reuven M. Lerner is finishing his PhD in Learning Sciences at Northwestern University. He lives in Modi'in, Israel, with his wife and three children. You can read more about him at http://lerner.co.il, or contact him at reuven@lerner.co.il.

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

DAVE TAYLOR

Web Administration Scripts—Redux

Dave explains how to add contextual memory to a script so it can keep track of trends in its environment. Then he explores why that's harder if the script stops and starts over time.

It's been months, and I'm still dealing with a DDOS (distributed denial of service) attack on my server—an attack that I can see is coming from China, but there's not really much I can do about it other than try to tweak firewall settings and so on.

In my last article, I started describing analyzing Apache log files and system settings to try to create some scripts that can watch for DDOS attacks and flag them before that inevitable e-mail from a customer about the site being down or incredibly slow.

Honestly, it's pretty disheartening dealing with these anonymous attacks, but that's another column. If you've had experience with this sort of long-term problem, I'd be interested in hearing about how you finally resolved it.

The first script I worked on simply tracks how many processes are running on the server. It's just a small step from monitoring the output of the `uptime` program, which, of course, could be done as easily as:

```
load="$(uptime | cut -d\  -f14 | cut -d. -f1)"
if [ $load -gt 4 ] ; then
  echo "Uptime greater than threshold. What's going on?"
  uptime
fi
```

That's marginally interesting, but let's go back to analyzing the number of processes running, as that's a better indication that there might be a DDOS happening. Last time, I used a loop of `ps ; sleep` and kept track

Once you've run this script for a few hours, you'll have a pretty good idea of typical traffic load on your server, which is critical for you to be able to detect out-of-pattern variations.

of the min/max values. After running for a while, the script's output was something like this:

```
Current ps count=90: min=76, max=150, tally=70 and average=107
```

Interpret this as after running for 90 cycles, the minimum number of Apache (httpd) processes running at any given time was 76, max was 150 and average was 107.

Once you've run this script for a few hours, you'll have a pretty good idea of typical traffic load on your server, which is critical for you to be able to detect out-of-pattern variations. If my average Apache process count is 107 and the server has 917 processes, there's a problem. On the other hand, if the average load is 325 processes, 600 isn't too far out of band and could be representative of a spike in traffic rather than the beginnings of an actual attack.

I wrapped up last month's article by showing a script suitable for running from `cron` that would look for abnormal traffic spikes. But it had some problems, as I highlighted at the end of the column, my nerd version of a cliffhanger—no murder unsolved, no car in the driveway, no police sirens, just some code that needs improvement. Hey, work with me here!

The core problem with the script was really a lack of context. That is, it's not the very first time that it sees abnormally high process counts that I want to be notified, but the third or fourth time in a row. And then once it has notified me, I want the script to be smart enough not to notify me again unless things settle down and then jump up again.

Context.

With some scripts, this sort of thing can be quite tricky, requiring temporary files and semaphores to ensure that the script doesn't step on another version of itself reading/writing the file simultaneously. It's doable, but you really have to think about worst-case situations, temporarily blocked I/O channels and so on.

But, what happens when there's then an iteration where the count isn't greater than the max threshold?

Fortunately, that's not the situation here. In fact, you can accomplish the addition of context by adding a couple state variables. Let's start by pulling the monitoring script back in:

```
#!/bin/sh
# DDOS - keep an eye on process count to
# detect a blossoming DDOS attack
pattern="httpd"
max=200   # avoid false positives
admin="d1taylor@gmail.com"
count="$(ps -C $pattern | wc -l)"
if [ $count -gt $max ] ; then
  echo "Warning: DDOS in process? Current httpd
  ➥count = $count" | sendmail
$admin
fi
exit 0
```

Let's use a "repeated" variable and set it to send a notification after the fourth occurrence of too many processes. That's just a matter of changing the conditional statement:

```
if [ $count -gt $max ] ; then
  repeated=$(( $repeated + 1 ))
  if [ $repeated -eq 4 ] ; then
    echo "Warning: DDOS in process? Current httpd count
    = $count" | sendmail $admin
  fi
fi
```

Not too difficult. But, what happens when there's then an iteration where the count isn't greater than the max threshold? That's also easily handled if you're willing to keep redundantly setting `repeated` to zero. The outer "fi" simply changes to:

```
else
  repeated=0
fi
```

These additions produce both of the desired updates actually, because `repeated` ensures that it won't notify of a problem until it's happened a few times, and the conditional `$repeated -eq 4` rather than, say, `$repeated -gt 4`, also means that if it's slammed for 15 iterations, you'll still see only one e-mail message.

Finally, visualize a sequence where you get a spike for a while, then it quiets down. Then another spike

hits for a few seconds, then it quiets down for an hour or two, and then it spikes again. That scenario will work as desired too, sending e-mail twice, once for each sustained spike, but not for the one-iteration spike in the middle (because in that instance, `repeated` never gets beyond 1).

If you've been looking closely at the script, you'll have noticed that what appears at first glance to be an innocuous "echo" statement is in fact producing an e-mail message to the administrator. It's the second half of the statement that's doing the heavy lifting:

```
echo "Warning: DDOS in process? Current httpd count
    = $count" | sendmail $admin
```

I'm old-school, so I go straight to the MTA (mail transport agent), but in reality, a better way to do this, an approach that would have the e-mail include a subject line, is to do something more like this:

```
echo | mail -s "DDOS Warning" $admin
```

Because sendmail is designed to be used behind the scenes, it lacks the refinement of things like a `-s` flag to specify the subject of the message.

In fact, you can make this even a bit more elegant by turning the entire command into a variable:

```
sendmsg="mail -s 'DDOS Warning' $admin"
```

Then you easily can tack it onto the end of the echo statement:

```
echo | $sendmsg
```

What's the biggest problem with this approach? That if there are any errors, the pipe is only for stdout, not stderr, and those errors will be lost.

If you were going to run the script every few minutes from `cron`, you could use its capability of e-mailing stdout + stderr, but that would involve a more complex contextual tracking solution (as discussed earlier).

Well, I'm off to install this script on my server and see how it does for a while. And you? Your assignment, dear reader, is to shoot in some proposed topics for me to explore in my column. What scripts would you like to see created, doing what?■

Dave Taylor has been hacking shell scripts for more than 30 years. Really. He's the author of the popular *Wicked Cool Shell Scripts* and can be found on Twitter as @DaveTaylor and more generally at http://www.DaveTaylorOnline.com.

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

KYLE RANKIN

Command-Line Cloud: gcalcli

Kyle starts a series on using the cloud (in this case, Google Calendar) from the comfort of the command line.

If you follow my columns in *Linux Journal*, you probably are aware that I'm a big fan of the command line. When it comes to getting things done efficiently, most of the time the command line can't be beat. This has led me to integrating services like instant messaging and Twitter into BitlBee so I can use them with my command-line IRC client Irssi (both written up in previous *LJ* issues).

You probably also are aware that unlike some other *Linux Journal* writers out there (I'm looking at you Bill Childers), I'm a bit more skeptical of cloud services. That said, in this day and age, it seems you can't escape cloud services, so what's a guy like me to do? Well, I've just figured out how to use cloud services from the comfort of my green-on-black terminal. So, I figured it might be interesting to write a few articles describing how you can access various cloud services from the command line.

The first cloud service I thought I'd cover also is one of the most prevalent: Google Calendar. Although a lot of calendaring options exist, in a corporate environment, you often are talking about either Exchange (usually larger, older-school companies) or Google Apps (usually smaller startups). Google Calendar can be pretty convenient as a shared calendar, particularly if you are issued an Android device at work. Plus, like with all major cloud services, you can collaborate and share your data with other people (like the NSA). My main problem with Google Calendar always has been having to open up a browser every time I want to view my calendar. Although I'm sure some people are logged in to the Gmail Web interface all day and get meeting notifications that way, I'm not. Because I'm usually glued to

After missing a few meetings, I decided I needed to find some way to put meeting notifications inside my terminal.

my terminal window, I often don't see those reminders. After missing a few meetings, I decided I needed to find some way to put meeting notifications inside my terminal.

The CLI to Google Calendar that worked for me was gcalcli. It provides a command-line interface to view your calendar in various formats and also lets you add events—all from the command line.

Install gcalcli

gcalcli itself was not packaged for my distribution, so installation took me a few extra steps. First, I cloned the project's git repository into a local directory named src that I use to keep track of source code (feel free to replace this with the directory you use for the same purpose). Then, I created a symlink to the included script so I could start identifying what Python libraries I'd need:

```
$ cd ~/src/
$ git clone https://github.com/insanum/gcalcli.git
$ sudo ln -s ~/src/gcalcli/gcalcli /usr/local/bin/gcalcli
```

At this point, I tried to launch the application but noticed I was missing a few required Python libraries, such as gflags, dateutil, httplib2 and google-api-python-client. The first three had Debian packages, so I was able to install them with:

```
$ sudo apt-get install python-gflags python-dateutil
➥python-httplib2
```

The final Python library wasn't packaged in Debian. For better or worse, it seems just about every modern scripting language has its own implementation of Perl's CPAN, so I fell back to using Python's pip (which I first had to install):

```
$ sudo apt-get install python-pip
$ sudo pip install --upgrade google-api-python-client
```

Initial Setup

With all of the libraries in place, you are ready to set up gcalcli to use your Google account. The first time gcalcli is run, it will set up your account, but if you are not running it from your local machine,

you will want to add an extra argument the first time you launch it so it can provide you with a URL to load locally to authorize gcalcli:

```
$ gcalcli agenda --noauth_local_webserver
```

Once it's set up, you can see your current calendar agenda with the `agenda` command (the events have been changed to protect the innocent):

```
$ gcalcli agenda

Mon Jul 15    9:15am  Space out and stare at screen
             10:00am  Meeting with the Bobs
             11:00am  Remodel Cubical Wall
             11:45am  Lunch at Flinger's
              1:00pm  Meeting with Michael
```

I've found the `agenda` output to be the most useful for seeing what's on my plate for the day; however, you also can use `calw` and `calm` to output ASCII-art-style weekly and monthly calendars, respectively (Figure 1).

Meeting Notifications in Screen

Having access to my calendar from

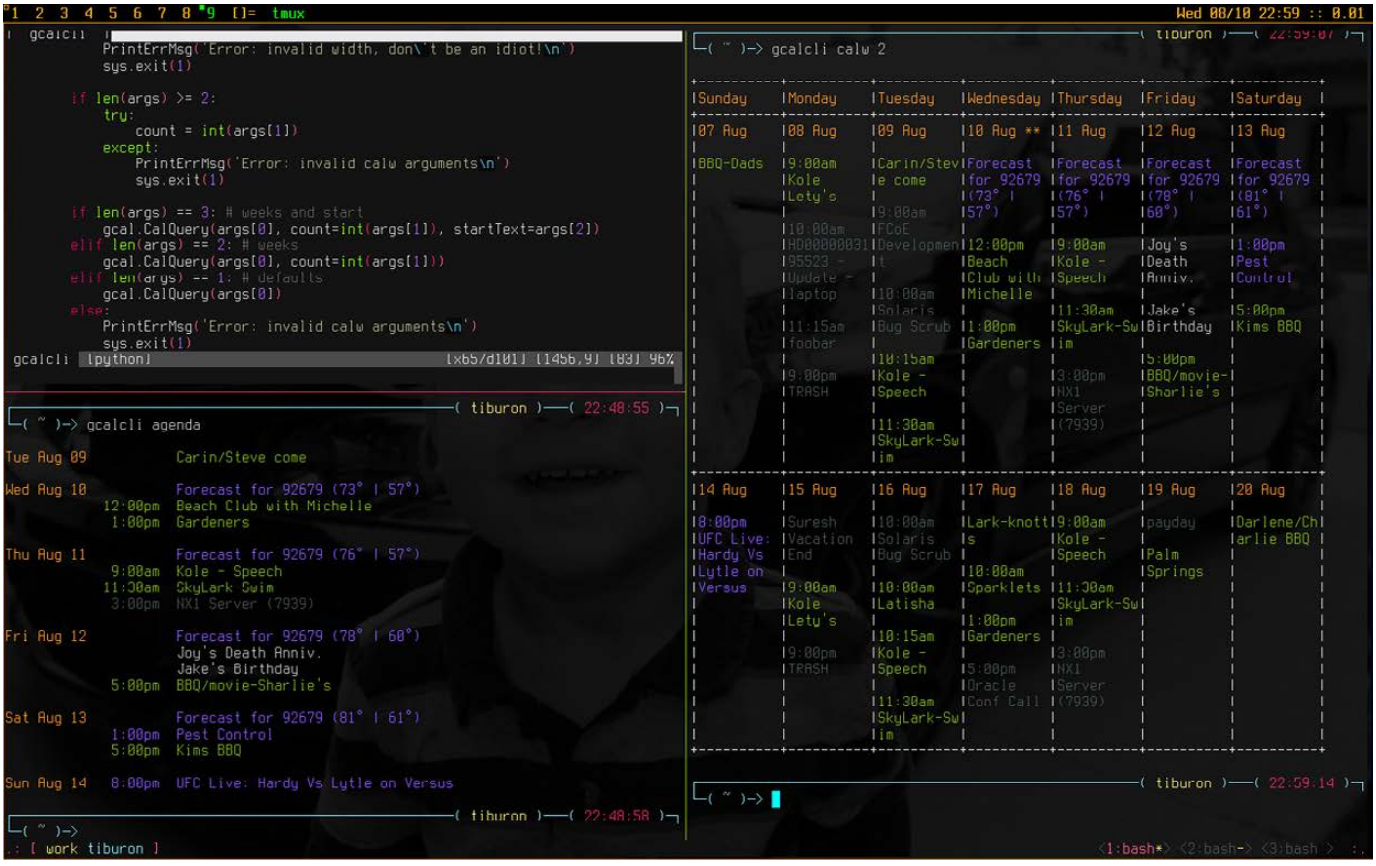


Figure 1. gcalcli Screenshot from the Official Project Page

Because notifications in screen are simply scripts that output short lines of text, what I want to do is tell gcalcli to display the next meeting in my agenda today that hasn't yet started, or if there are no more meetings today, to display nothing.

the command line is handy, but it still doesn't quite solve my problem of getting notifications so I don't miss an important meeting. I'm sure many people simply rely on the pop-up notification in their browsers, but I usually am not looking at my browser, and when I am, I'm not logged in to Gmail. I am, however, almost always looking at a screen session, so I found the best approach for me was to put notifications there.

I wrote about how to set up screen notifications in my February 2011 Hack and / column, so I recommend reading that column if you haven't set up your hardstatus line yet. Because notifications in screen are simply scripts that output short lines of text, what I want to do is tell gcalcli to display the next meeting in my agenda today that hasn't yet started, or if there are no more meetings today, to display nothing. That way, I can see at a glance what meeting is next.

The first step is to create a basic shell script in my local ~/bin/ directory called gagenda that calls gcalcli with extra arguments and then parses the results:

```
$ gcalcli --nocolor --nostarted agenda "`date`" 11:59pm |
➥grep -v 'No Events' | head -2 | tail -1 | grep -v '^$'
```

With this long one-liner, I tell gcalcli to show me all of the events on my agenda that haven't started (`--nostarted`) between now (`"`date`"`) and 11:59pm. Then if there are no events, I grep that out so I get empty output. If there are results, I use head and tail to pull out the line I want. Finally, if the line is empty I also grep that out. The end result is a single line of output suitable for screen.

Of course, I didn't want to stop there. What I wanted was for regular events to show up in

The key to having a status appear to change colors is to have different backtick commands show up in the same location, each assigned its own color.

white on my output, but then as the meeting got closer, I thought it would be nice if it turned to yellow and then finally to red. Also, when commands in your screen hardstatus line take a long time to run, the whole screen session can stall, so my solution was to have my gagenda script redirect the output to a few temporary files. The key to having a status appear to change colors is to have different backtick commands show up in the same location, each assigned its own color. Then as long as you make sure that only one of the commands will display output at any time, you get the illusion that you have only one status that changes color. Here is the resulting gagenda shell script to accomplish that:

```
#!/bin/bash

# first look for urgent events occurring
# in the next five minutes
gcalcli --nocolor --nostarted agenda "`date`" "`date -d
 ➥'now + 5 minutes'`" | grep -v 'No Events' | head -2 |
 ➥tail -1 | grep -v '^$' > /tmp/gagenda-urgent
# if there are no urgent events, look for warning
# events in the next 10 minutes
if [ ! -s /tmp/gagenda-urgent ]; then
  gcalcli --nocolor --nostarted agenda "`date`" "`date -d
 ➥'now + 10 minutes'`" | grep -v 'No Events' | head -2 |
 ➥tail -1 | grep -v '^$' > /tmp/gagenda-warning
# if there are no warning events, just grab the next
# event for the day
  if [ ! -s /tmp/gagenda-warning ]; then
    gcalcli --nocolor --nostarted agenda "`date`" 11:59pm |
 ➥grep -v 'No Events' | head -2 | tail -1 |
 ➥grep -v '^$' > /tmp/gagenda
  else
    cat /dev/null > /tmp/gagenda
  fi
else
  cat /dev/null > /tmp/gagenda-warning
  cat /dev/null > /tmp/gagenda
fi
```

Now I edited my local user's crontab with `crontab -e` to add gagenda:

```
* *     * * *    /home/greenfly/bin/gagenda
```

To configure screen, I updated my .screenrc with three new backtick

ManageEngine
ServiceDesk Plus

ITIL-READY HELP DESK SOFTWARE

WITH INTEGRATED ASSET & PROJECT MANAGEMENT

IT MADE EASY

Incident Management . Problem Management

Change Management . Service Catalog

Asset Management . CMDB . Project Management

www.servicedeskplus.com | demo.servicedeskplus.com

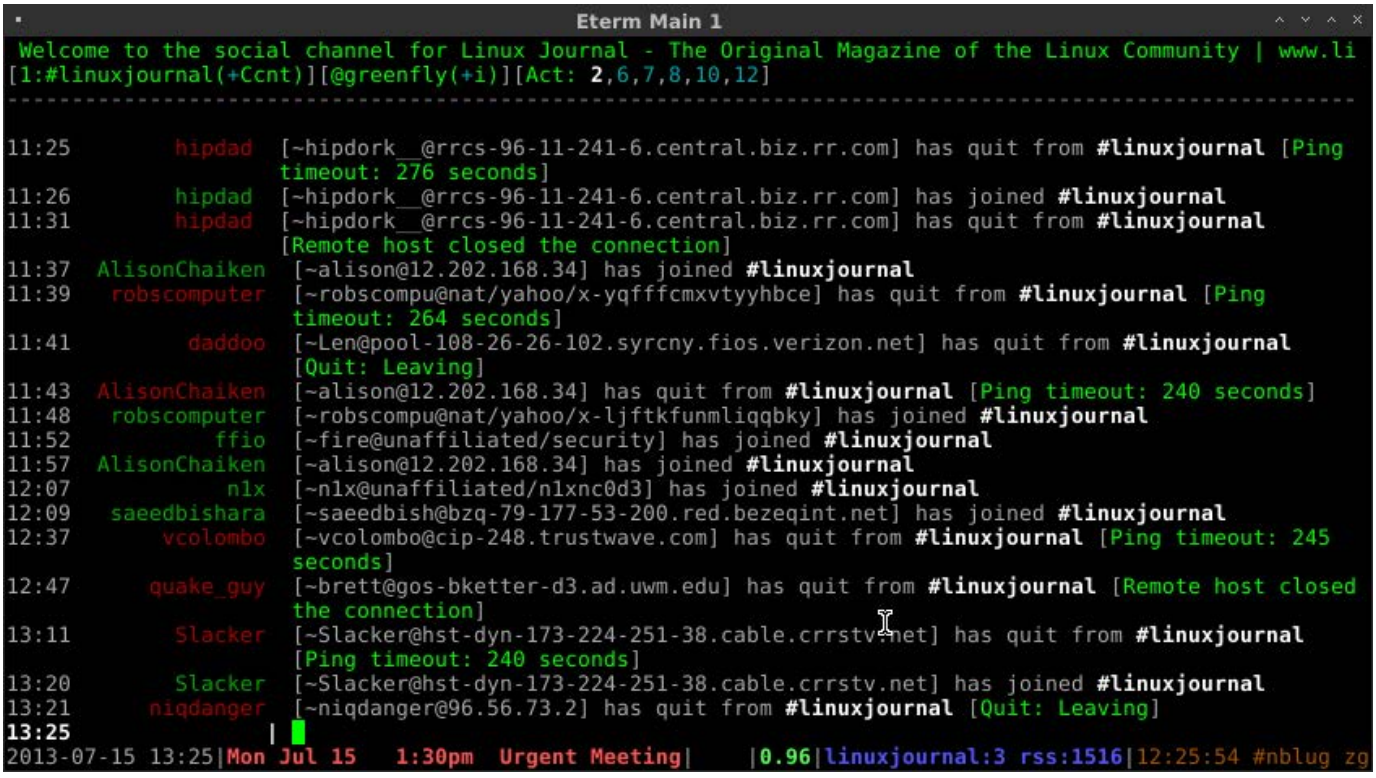


Figure 2. Urgent Meeting

commands that just cat the output of the temporary files:

```
backtick 111 37 37 cat /tmp/gagenda
backtick 112 37 37 cat /tmp/gagenda-warning
backtick 113 37 37 cat /tmp/gagenda-urgent
```

And finally, I referenced those backtick commands with the appropriate colors in my hardstatus line:

```
hardstatus string '%{= w}%Y-%m-%d %c%{= w}|%{= w}%111`%{+b
➥y}%112`%{+b r}%113`%{-}%{= w}|%= |%{+b g}%108`%{-}|%{+b
➥b}%105`%{-}|%{= y}%107`%{= w}'
```

The relevant section of the hardstatus line for these commands is:

```
|%{= w}%111`%{+b y}%112`%{+b r}%113`%{-}%{= w}|
```

Once I reload my .screenrc file, Figure 2 shows that I have an urgent meeting in a few minutes.

In this article, I just touched on using gcalcli to access your agenda from the command line, but in my next column, I'll expand on the topic to talk about how to add new events and do even more with your Google Calendar without having to leave your terminal.■

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.

When: **September 13, 14 & 15**

Where: **Greater Columbus Convention Center, Columbus, Ohio**

Keynotes:

Robyn Bergeron, Fedora

Marshall Kirk McKusick, FreeBSD

Mark Spencer, Digium

Jon “maddog” Hall

Other Info:

OLFI - one day of professional training for system administrators by companies you know

Saturday Expo

Friday Birds of Feathers sessions

...AND MORE

Ohio LinuxFest 2013

ohiolinux.org

SHAWN POWERS

The Way I Do the Things I Do

A peek into Shawn's toy box.

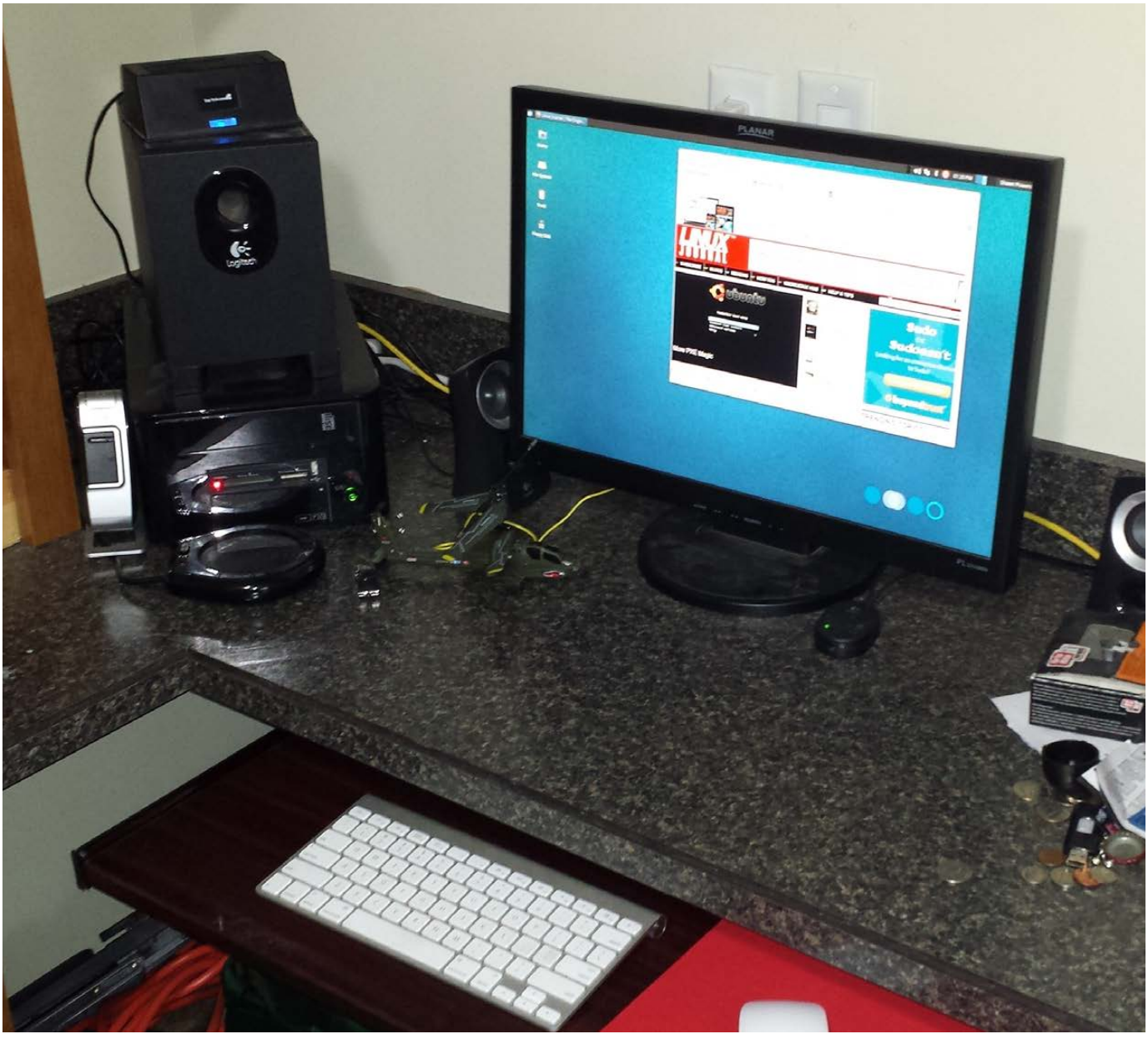


Figure 1. Booted in to Xubuntu. Model helicopter not included.

Since my "Trying to Tame the Tablet" article was published in the February 2013 issue of *LJ*, I've been getting a steady stream of e-mail messages asking about what tools and programs I use on a daily basis. Because this is our How-To issue, I figured it would be appropriate to give a rundown on how I do the things I do. I must warn you though, I'm not going to pull any punches—I use multiple operating systems every day. Thankfully, Linux is still king, so I'll try to keep my other dallyings to a minimum.

The Hardware—Desk

I have two main workspaces in my office. The first is for my day job, which requires me to use a giant Wacom monitor and Windows 7. I do Linux training during the day, so that Windows 7 machine has about a dozen Linux VMs and uses Camtasia for recording.

Everything else I do is at my personal workspace. That's the area I focus on here, because it's the place I've configured exactly the way I like. Figure 1 shows my workspace. The chair is just a cheap office chair from an office supply store. You'll notice there aren't any armrests on the chair, which is because I took them off (Figure 2). Armrests annoy me while I work, and if I'm doing any recording, I find the headphone/microphone cables get tangled. Plus, I sometimes sit cross-legged in my office chair, and that's not possible with armrests in the way.

The computer I use was purchased based on the TonyMac Hackintosh Mini-Mac build (**http://snar.co/customac**). My hardware is actually from last year's recommendations, but it still is able to dual-boot OS X 10.8.4 and Xubuntu 13.04. I'm

Figure 2. This chair isn't fancy, but it works.

I've had wonderful luck with Brother printers through the years, and they always seem to have decent Linux support.

very fond of Apple's Chiclet-style keyboard, so I'm using that and a Magic Mouse for input.

For simple recording or VoIP calls, I use my wireless Plantronics CS50 USB headset. The software for the device is Windows-only, but as a headset and microphone, it works just fine. I think the button on the side is a gimmick anyway, so I'm not upset by the lack of software support. If I'm recording something where sound quality matters, I have a Sennheiser USB headset and microphone combo. The model I use is long discontinued, but I love the quality of every Sennheiser I've ever owned.

My monitor is just a 24" 1080p monitor I got from http://woot.com for $350. My dream monitor is the 30" Dell Ultrasharp, but so far, Santa Claus hasn't brought me one. I'm connecting to the computer via DVI, but only because I couldn't find an HDMI cable in "the big box of tangled cable".

Currently, I don't have a printer. It's not due to my desire to go paperless, but rather because, like a fool, I went back to an inkjet printer a year ago, and it's been problematic ever since. My hatred for inkjets has been renewed 1000%. Because this article is meant to reflect my work environment, I can tell you that I hope to purchase a Brother color laser printer soon. I've had wonderful luck with Brother printers through the years, and they always seem to have decent Linux support. I haven't chosen a model number yet, but it will be a Brother for sure. (I'm trying to decide whether I should buy the multifunction machine with a FAX, or just let the old technology die. The problem is, I still need a FAX from time to time.)

The Hardware—Mobile

My mobile hardware recently has undergone a change, at least partially. My last job provided an iPhone, so that was what I used for years. My new training job allowed me to get whatever type of phone I wanted, so I chose the Samsung Galaxy S4. I tend to use my smartphone a lot during the day, so even with the S4's impressive power management, I had to buy an oversized battery. I bought

Figure 3. The blue matches my car. So does the size.

the Hyperion 5200mAh battery (**http://snar.co/s4battery**), which came with a custom phone back to fit the bulge. Hyperion also offers a case to fit the bulging phone, so I bought that as well (**http://snar.co/s4case**). It certainly isn't as svelte as the out-of-the-box S4, but I think it's less bulky than my old iPhone with the Mophie Juicepack (Figure 3).

I do still have my Google Nexus 7 tablet, whose original tale inspired this article. I've purchased at least five cases for that tablet, ranging from a neoprene sleeve to a Bluetooth keyboard case. In the end, I've defaulted to using a simple clamshell case. I also have the same stylus mentioned in the earlier article, and although I don't write with it, I find the precision of using a stylus helpful at times. (Crayon Physics, I'm looking at you.)

For the past ten years or so, my daily driver laptop has been a MacBook Pro, because that's what was provided by my employer. (No, not *Linux Journal*!) I don't have a work-provided laptop anymore, so I spend my couch-surfing time largely with the computer I'm typing on right now: the same Dell D420 I was given by a reader when our house burned down four years ago. It's been upgraded with a PATA SSD and a new battery, but this laptop has been faithful for years, and now as my full-time machine, I have no complaints. It's showing its age in that the screen hinges are loose, but that's truly my only complaint.

I also have a Samsung Chromebook (the ARM-based one Bill Childers reviewed in the July 2013 issue), and although my original plan was to use that as my main computer, my kids have had other plans. They use my Chromebook constantly, and I rarely get to play with it. If you wonder whether a Chromebook would be

a good fit for your kids, given my experience, the answer is a hearty *yes*.

I do still have my old Samsung Galaxy S2 phone, which no longer has cell service. I've been using it as an audiobook player and GPS in my car. I keep the audiobooks updated by having FolderSync (an Android app) run over Wi-Fi from the driveway overnight. As long as I route my trip while still in the driveway, Google Maps seems to cache the entire trip without the need for a constant data connection. If I need to route a new trip while away from the house, I just activate the mobile hotspot on my S4 and leave it on long enough to create the map cache. So far, it's been a good solution, and it keeps my phone in my pocket so I don't leave it in the car.

The Mobile Software

Let me start with my mobile devices. For the software section, I'll lump my laptops in with my desktop, because they use the same applications. My phones and laptops, being Android, run separate programs. Here's a list of things I regularly use:

- *Candy Crush Saga*: yes, I'm addicted, I'll admit. I resisted for a long time, because I didn't want to connect to Facebook and pollute my timeline with *Candy Crush* requests. It turns out, if you don't connect to Facebook, you still can keep playing by completing quests. If you've never tried *Candy Crush*, I'm not sure whether I should recommend it. You *will* become addicted. (I'm

Figure 4. I'm an addict. I dream about striped candy and power balls.

currently on level 133, and I've never purchased anything in the game, nor do I plan to.)

- Kindle: I've tried many e-book readers on many different platforms. The Kindle interface isn't my favorite, but I love the cross-device Whispersync, and I find E Ink to be my favorite for long reads. Because I can read a chapter or two on my phone, then continue reading on my Kindle keyboard (Wi-Fi-only) later that night, it's become my go-to reading app. Now that Whispersync works for personal documents, Kindle doesn't even annoy me anymore.

- Gas Buddy: this was more useful when we lived in Grand Rapids, Michigan, because gas prices could vary 20–25 cents a gallon at stations just a couple miles apart. Now that we're back in Small-Town USA, I don't use this as often, because the prices always are the same.

- LED Flashlight: I can't believe how much I use this app. I think I use it more now than ever before, because it's possible to launch from my phone's lockscreen. I've always found flashlight apps cumbersome to find and start, but no more. I really do use this every day, sometimes several times a day.

- Camera: I try to take funny photos as often as I can. Like everyone else on the planet, now I have a camera in my pocket. Like the flashlight, the ability to launch the camera from the lockscreen makes it more convenient, and more used.

- Google Music: I don't listen to music often, but I keep a few albums on Google Music, because I know I can reach it anywhere. I most often use it to play a Jonathan Coulton song for someone who's never heard it before.

- *Linux Journal*: I honestly don't use the *Linux Journal* app very often. By the time the issues go live, I've already been reading through them during the entire production process. I keep the app for the same reason I use Google Music though—to show others cool things!

- Yaste: we lose our television remote more often than I care to admit. Yaste is a great XBMC remote. It's not as nice as using the regular MCE remote, but it's usually easier to find.

- Dock Clock: I haven't had a clock radio in years, but I still miss the large digital time next to my bed. Dock Clock automatically activates when I plug in the charger, and it allows me to create the clock radio feel while still using the far-superior cell-phone alarm system (Figure 5).

- Google Mail: well, yeah. I almost forgot to include it, but I use it constantly.

- Chrome: Same here. I almost didn't include it, but I realize there are other Web browsers, and not everyone uses Chrome. The only annoyance I have is that there's no way to enable the "bookmark bar" like in the desktop version of Chrome. I understand it would be cramped on my phone, but on the tablet, it would be awesome.

- Smart Audiobook Player: Best. Audiobook. Player. Ever. I love everything about Smart Audiobook Player. I listen to most books at 1.4x or 1.5x depending on the narrator, and I don't notice any "chipmunk" effect at all. I do have the Audible app too, but I try my best to use straight up MP3 files for audiobooks and sync them with FolderSync.

Figure 5. My clock radio in 1993 only wishes it looked this cool.

- Social media: I'm a social-media junkie, so I have Twitter, Facebook, FourSquare, Instagram, SnapChat, Flickr and G+. I usually initiate most things with Twitter, but try to respond on most networks.

The Desktop Software

These days, most of my desktop applications are really just Web apps. There are a few native apps that I can't live without though:

- Pithos: this plays music from the Pandora network. It works flawlessly, and I love it. If I ever want to hear some music, I usually just turn on Pithos, because my personal music collection is rather small.

- TextRoom: in a pinch, I still use the Chrome app Writebox, but for most writing (like what I'm doing now), I still use TextRoom. For a while, it was difficult to use, because it wouldn't install on modern Ubuntu versions. Thankfully, a man named Otto has created a PPA for TextRoom that works perfectly. I owe Otto a beverage of his choosing!

- Dropbox: thanks to my Chromebook, I now have more than 100GB of space in Google Drive. I've been using Dropbox for so long, however, that it's hard to switch. Perhaps the program I mention in this issue's UpFront section, Insync, will help me transition, but for the present, my 24GB of Dropbox is where I keep most of my writing.

- Calibre: I have a fairly large collection of e-books, and without Calibre, they'd be a mess. Calibre has plugins for automatically sending books to my Kindle account, so I find myself using it often.

- Chromium: this really ends the list of native apps I use regularly. I'll occasionally play a game or something, but not very often. Everything else I use regularly is a Chrome extension or plugin.

Web Apps

One of the reasons I love Chrome/Chromium/Chromebooks is that extensions sync just like bookmarks. If I install an app on my laptop, it's waiting for me when I go back to my desk—even if that desk is running a different operating system! Here are the apps I use regularly:

- MightyText: now that my family has iOS devices and I don't, we can't use iMessage to talk to each other. Thankfully, MightyText brings texting to my browser, so while I'm at a keyboard, I can text at full speed.

- Evernote: I don't use Evernote as much as I did when I was working in an office setting as a manager, but I still use it as a knowledge repository. The Web app is really nice, and it's usually all I need.

- Lastpass: I forced myself to migrate all my passwords to Lastpass, and now that I use it all the time, I've come to love it. I'll admit, I didn't care for it at first.

- Tweetdeck: I've been on a quest to

find the best Twitter client since my beloved Twhirl died away. I haven't found one I love, but Tweetdeck's Chrome app comes the closest.

- CommaFeed: my Google Reader replacement, mentioned in the UpFront section of this very issue.

Green and Black

If Kyle Rankin wrote this article, every application would be replaced with a CLI equivalent. Although I have amazing respect for Kyle's way of computing, I also like shiny things. Still, there are quite a few applications I rely on that have nothing more than green text and a black screen:

- vim: you know you're a vim user when you are annoyed by "easier" text editors like nano. I can open, edit and save a file quicker than most people can find Ctrl-x on their keyboard to exit nano.

- Irssi: for IRC chatting, it's hard to beat Irssi. For me, this isn't because I prefer the Irssi interface. Truth be told, I prefer the way Pidgin handles IRC. I like to stay logged in to IRC all the time, however, so I can respond to questions posed while I'm away. The screen command coupled with Irssi accomplishes that for me.

- SSH: truly, truly the Swiss-Army knife on the command line. Remote shells, tunneling traffic, remote/reverse tunneling, keypair authentication, file copying...SSH is amazing. I use it every day, all day long.

Now for the Interactive Part of Our Show...

So there you have it. That's my computing world in a nutshell. Perhaps you see something helpful, or perhaps you're appalled at my lack of tool usage. Either way, I'd love to hear from you. In fact, if you have a unique application or method for accomplishing your day-to-day tasks, send me an e-mail! I love to highlight new or cool (preferably both) applications or Web sites in our UpFront section. If you send me something awesome, and we publish it, I'll be sure to give you credit for the heads-up. Just drop me an e-mail at shawn@linuxjournal.com and put something like "[MY WAY]" in the subject. I can't wait to hear how everyone computes!■

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.

AdaCore's GNAT Pro for Wind River Linux

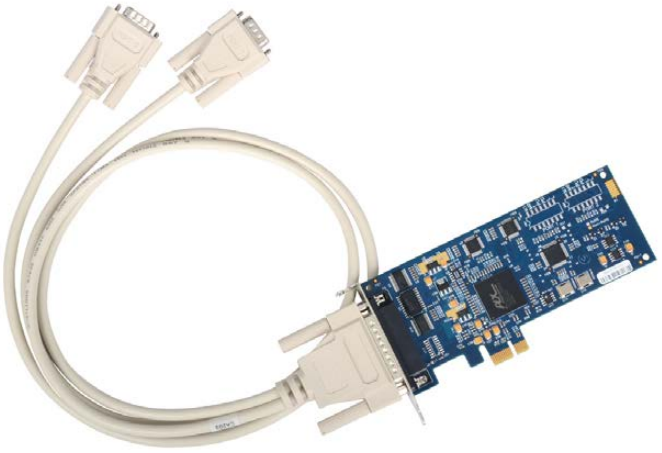
The slickest thing about the GNAT Pro 7.1 Ada development environment for Wind River Linux is the ability to use AdaCore and Wind River products together to develop applications that freely combine modules in Ada, C and C++. Wind River Linux is a leading commercial-grade Linux solution for embedded device development. In addition, developers can manipulate and analyze Ada applications via Wind River's Linux browser and tools. This new implementation of GNAT Pro supports all versions of Ada and is tightly integrated into the Wind River Workbench development environment. AdaCore says that the new GNAT Pro solution brings the Ada development community everything it needs to build and support highly differentiated solutions and deploy them on an industry-leading commercial-grade Linux solution, based on the Yocto Project open-source development infrastructure. GNAT Pro for Wind River Linux includes support for the Wind River Linux 4.3 platform, the PowerPC and Power PC e500v2 target platforms, and the Linux host.

http://www.adacore.com/gnatpro

Symantec's Data Insight

The rapid growth of unstructured data and the lack of control of "dark data" means many data centers are stocking petabytes of information but not using it as a business asset. Symantec's solution to help organizations better understand their unstructured data is Data Insight, which has been upgraded to version 4.0. Symantec notes how unstructured data—about which many organizations have minimal awareness—not only results in high storage costs but also presents the potential for compliance infractions and higher organizational risks. New enhancements to Data Insight 4.0 will allow users to identify and fix problems in a single tool, create information-rich reports, mitigate organizational and data security risks and implement effective information lifecycle management.

http://www.symantec.com/data-insight

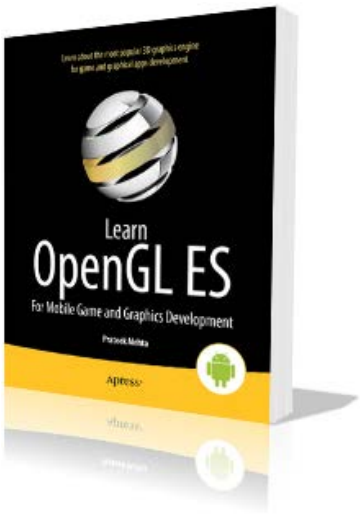


Sealevel Systems' 7202e

Sealevel Systems, provider of industrial computing, communications and I/O solutions, announced the release of the 7202e, a low-profile two-port RS-232 PCI Express serial interface adapter with advanced UART architecture. For maximum compatibility with a wide range of serial peripherals, the board supports all RS-232 modem control signals, making it a perfect solution for interfacing instrumentation, bar-code readers and other data acquisition/control devices. The 7202e's high-performance 16C950 UART includes 128-byte FIFOs for error-free operation in high-speed serial applications. All Sealevel PCI Express serial adapters include SeaCOM software for Linux and Windows, as well as WinSSD, a full-featured application for testing and diagnostics.

http://www.sealevel.com

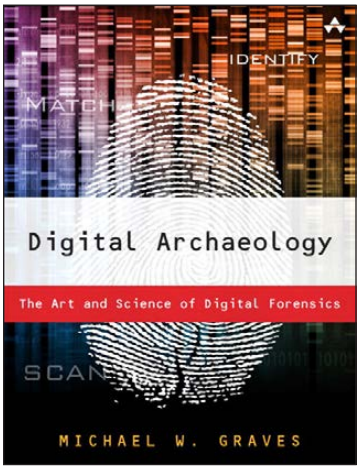
Prateek Mehta's *Learn OpenGL ES* (Apress)



If you are new to game development on mobile platforms, chances are that Prateek Mehta's *Learn OpenGL ES: For Mobile Game and Graphics Development* will help you build sophisticated, graphically rich game apps. Many of today's sophisticated game and graphics-intense apps rely on the language and rendering engine OpenGL ES. Whether you're an Android or iOS app developer, this book is for you. This book ramps up its readers with the primary skills of the new OpenGL ES 3.0 and gets them quickly into game app development without intense study of object-oriented programming techniques. This book also demonstrates the use of the Blender modeling software. Advanced topics include collision detection, player-room-obstacle classes and storage classes, as well as how to apply learned techniques in the context of the limited resources and processing capacity on mobile devices.

http://www.apress.com

Michael Graves' *Digital Archaeology* (Addison-Wesley Professional)



Publisher Addison-Wesley Professional markets Michael Graves' new book *Digital Archaeology* as "the definitive, up-to-date guide to digital forensics". The rapid proliferation of cyber crime is increasing the demand for digital forensics experts in both law enforcement and in the private sector. In *Digital Archaeology*, expert practitioner Michael Graves has written a thorough, realistic guide to the principles and techniques of modern digital forensics. After providing the legal issues surrounding computer forensics, Graves explains how to investigate computer systems systematically and thoroughly to unearth crimes or other misbehavior, and back it up with evidence that will stand up in court. Drawing on the analogy of archaeological research, Graves explains each key tool and method investigators use to uncover hidden information reliably in digital systems. Graves concludes with coverage of important professional and business issues associated with building a career in digital forensics, including licensing and certification requirements.
http://www.informit.com

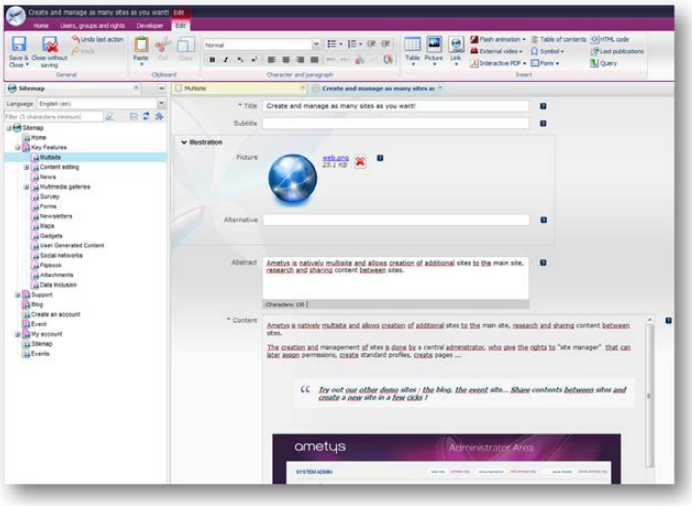
SUSE Linux Enterprise

The hallmarks of the new SUSE Linux Enterprise Service Pack 3 are greater scalability and security, not to mention additional industry-standard hardware support and open-source features and enhancements. SUSE calls SUSE Linux Enterprise 11 "the most interoperable platform for mission-critical computing—across physical, virtual and cloud environments". The service pack includes all patches and updates released since the introduction of SUSE Linux Enterprise 11 in 2009. SUSE adds that Service Pack 3 is an excellent scale-up choice to migrate resource-intensive workloads that run on RISC/UNIX systems or for virtualized workloads on the latest AMD Opteron 4000 and 6000 Series Platforms. Also included are support for and improvements in virtualization and power savings for the latest Intel Xeon E5 processor family and the fourth-generation Intel Core processor). Service Pack 3 supports large systems with up to 4096 logical CPUs and 16TiB of system memory.
http://www.suse.com/sle11sp3

Opengear IM7200

Bullet-proof and future-proof are two adjectives that Opengear uses to describe its new IM7200 infrastructure management solution. The state-of-the-art device streamlines remote management of network, server and power infrastructure in data-center and remote environments, ensuring efficient and secure access, monitoring and control for local and remote operations teams. Opengear points out how data centers, colocation facilities and network exchange points host a wide array of critical network, power and server infrastructure from multiple vendors. The scope and complexity of managing these environments securely and efficiently continues to grow with faster physical media and more rigorous security protocols. The Opengear IM7200 is a solution built to manage all rack infrastructure, even during network downtime. Built for high-performance remote management, the solution offers true out-of-band infrastructure management and proactive monitoring while integrating with existing tools and workflow.
http://www.opengear.com

Anyware Service's Ametys CMS



Version 3.5 is the current milestone for Ametys CMS, a Java-based, open-source multipurpose Web factory that can run corporate Web sites, blogs, portals and corporate wikis on the same server. Ametys CMS's other strength, says its producer, lies in its uncommonly intuitive look and feel among open-source CMSes, which is analogous to Microsoft Office applications. Noteworthy new features in version 3.5 include expanded content sharing between different sites, a common resources function, improved user accounts features and a plugin for YouTube playlists and several back-office tweaks.
http://www.ametys.org

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.

REVIEW

HARDWARE

Acer C7 Chromebook

Dual-booting and change-rooting—lots of fun on the Acer C7!

PETER COOK

Figure 1. Acer C7

The Acer C7 caught my attention when my wife and I determined we could no longer share our single Linux Mint laptop. My wife was frequently squinting at her cell phone's Internet browser, fighting my four-year-old daughter to use our Nexus 10 tablet, or struggling to remove my eight-year-old son from his Ubuntu laptop when I needed to use our shared laptop for school or work. I initially was drawn to the Acer because of its low $199 price. However, the C7's upgrade-ability really impressed me. So, I bought one, model C710-2847, in January 2013 at a local Best Buy along with a $25 4GB DDR 1300 memory module just in case the laptop's out-of-the-box performance was not adequate.

The 1.4kg Acer C7 laptop has a dual-core 1.1GHz Intel Celeron 847, 2GB DDR3 SDRAM, 5400 RPM 320GB hard drive and 4-cell battery. The screen, keyboard and touchpad are definitely sufficient for my wife to browse the Internet and read e-mail. The 11.6" glossy screen is nicely bright at 200 nit and easily viewed indoors. Rounding out the periphery hardware, the C7 has an SD/MMC media card reader, 1.3-megapixel Webcam, microphone, three USB 2.0 ports, VGA output and HDMI port.

The C7 uses the Chrome operating system, which is a lightweight operating system optimized for Web browsing. Do not confuse ChromeOS

NOTE: The first prototype Chromebook was the Cr-48 released in 2010, and several Chromebooks have been produced since then. Google released the latest Chromebook this year and called it the Pixel for good reason. The laptop shines with a beautiful 239 pixels per inch screen and other high quality features, but it comes at a steep $1,299 price. Most other Chromebooks contain average specifications with a 1.x GHz Intel Atom or Celeron processor, 2GB of RAM, 16GB solid-state drive and ~12" 1366 x 768 screen. Two notable exceptions to these specifications, besides the Pixel, are the Samsung ARM Series 3 and Acer C7. The Samsung is razor-thin with an Exynos 5250 reduced-instruction-set-computer (RISC) processor, matte screen and $249 price. The Acer's standout feature is a 320GB hard drive at a bottom-priced $199.

Bill Childers reviewed the Samsung ARM Chromebook in the July 2013 issue, and this month, Peter Cook reviews its closest competitor, the Acer C7 Chromebook. The two laptops differ significantly in hardware and upgrade-ability.

COST: I was further drawn to the Acer when I found out that Google offers two years of free 100GB Google Drive storage with the purchase of a C7. I could use the storage to back up my photos and documents. The 100GB plan costs $4.99 per month, making the offer's value just shy of $120. I didn't need a calculator to realize this brought the C7's cost down to $79!

The free plan will be applied to your Google account soon (that is, not immediately) after registering the Chromebook. Don't expect Google to figure out how to apply the free 100GB if you have an existing Google storage plan. The free storage will be added to your existing plan. For example, someone with an existing 25GB plan will see 125GB at $2.49 per month, which is the cost of 25GB. Simply cancel the existing storage plan, and the 100GB will remain at no cost. Data stored under the existing paid plan will be available automatically on the free plan.

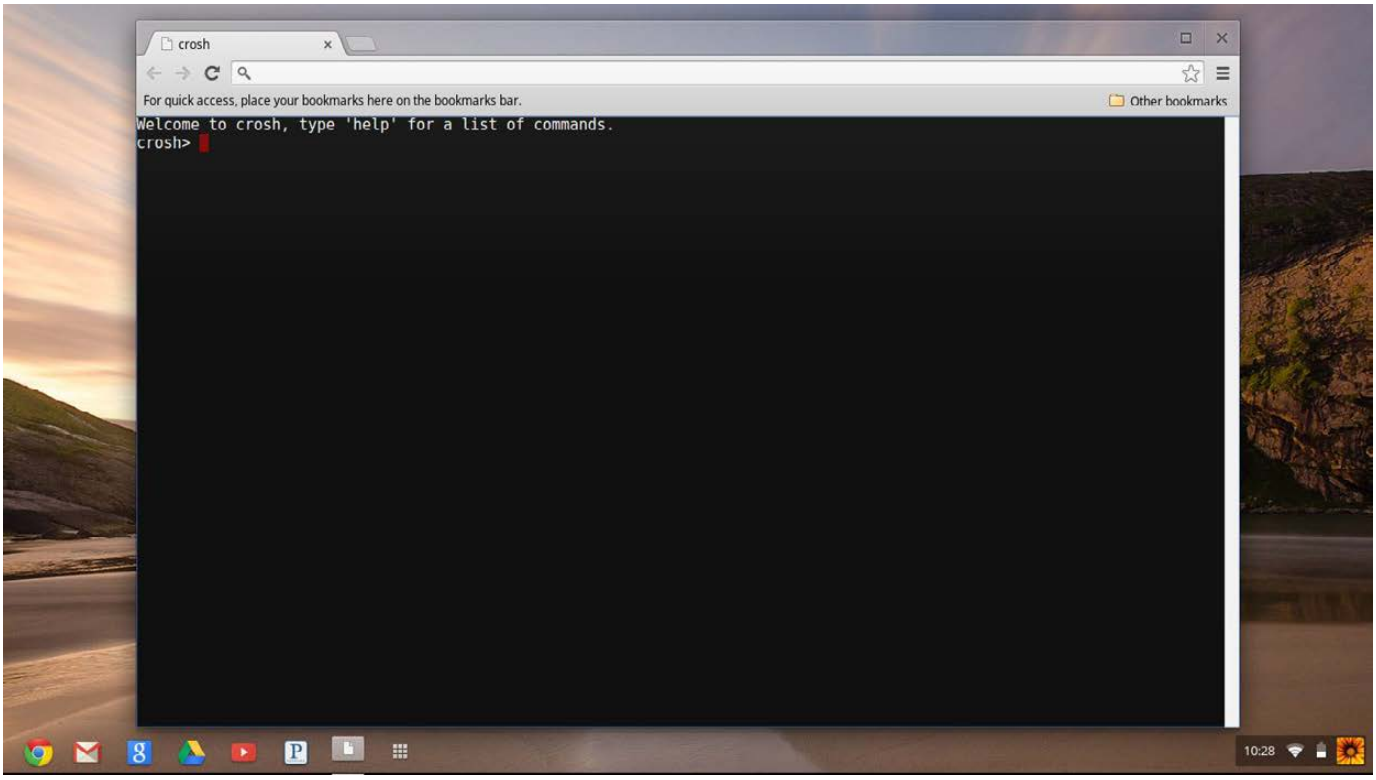


Figure 2. Acer C7 with crosh

with the open-source Chromium operating system. ChromeOS began from Ubuntu Linux in 2009 with a contracted engineering service from Canonical. A year later, Google switched to Gentoo Linux. Today, ChromeOS is not based on any Linux distribution and is released by Google

only for official devices. However, the source code is available through the Chromium OS project and can be installed on potentially any device.

ChromeOS provides a specialized debug terminal via the Chrome browser. Press Ctl-Alt-T, and a terminal will open with "Welcome to crosh, type 'help' for a list of commands." Type `help` to list seven commands including the always useful `top` and `ssh` commands. Enter `help_advanced` to produce a longer list of mostly network-related commands, such as `network_diag`, which performs a suite of network diagnostics.

Upgrades

Acer released the original C7 in November 2012 and then updated it in March 2013. The updated C7 has only two changes from the previous version. RAM has been doubled to 4GB, and the battery is upgraded to 6 cells. The RAM is perhaps a worthy change, but the larger battery noticeably protrudes from the laptop's bottom. The price also increased to $279. The increased price is questionable, since the original C7 can be upgraded easily. Common upgrades are memory, hard drive, battery and operating system.

Memory The C7 comes with one 2GB memory module installed in one slot and another empty slot. Maximum memory size available for the C7 is 8GB, allowing a total of 16GB. Memory may seem like a necessary upgrade, as the C7 does not use swap space. However, you can enable swap by opening the crosh terminal, typing `swap enable` and rebooting. A warning will appear in crosh stating that swap is an experimental feature and system instability may occur. Once rebooted, type `swap status` to verify the change. You should see /dev/zram0 listed under Filename with a size of 2932924 megabytes. Typing `swap disable` and rebooting will turn swap off.

However, based on my trials, the installed 2GB is sufficient, and swap does not need to be enabled. I opened ten tabs, some with a *Linux Journal* video running, a Netflix movie showing and a YouTube video playing, among several other static pages, and I still was able to play *Angry Birds* without issue. Moving between tabs also was seamless, and content was uninterrupted. I soon realized the extra 4GB memory I purchased was not needed. Unfortunately, Best Buy's return policy was 15 days, which surprised me because I expected 30 days. So, I installed the additional memory. I took off the C7's bottom

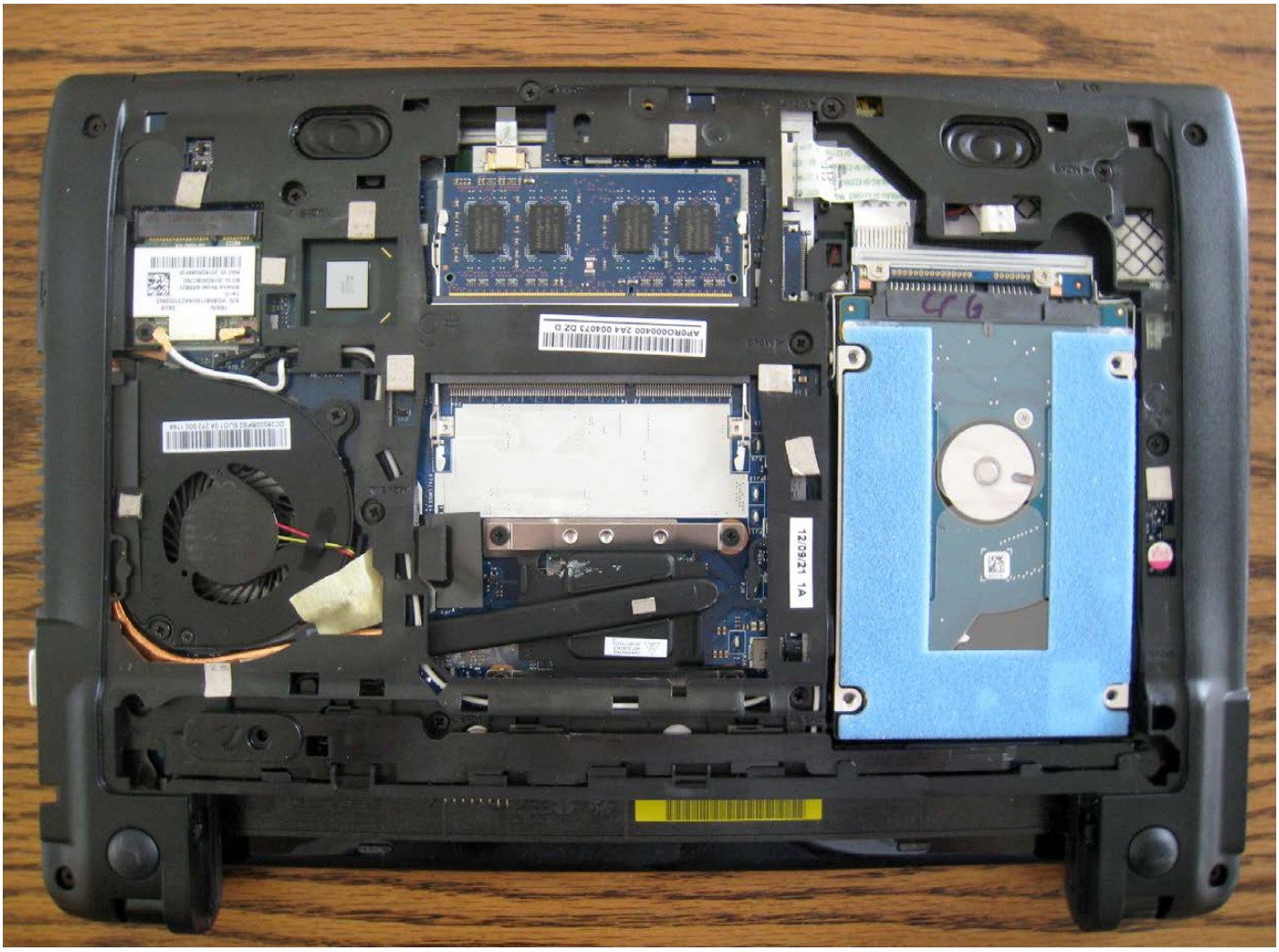


Figure 3. Acer C7 with Bottom Removed

plate and inserted the memory module. I verified the install by typing `chrome://system` in the Chrome address bar and then clicking Expand next to meminfo.

Hard Drive The existing hard drive can be updated with a solid-state drive (SSD), but there are trade-offs. You will give up a lot of storage space unless you purchase a large SSD, which can be expensive. Even the smaller SSDs are not cheap. The extra storage space may be handy if dual-booting the laptop with a Linux distribution. On the other hand, SSDs are lighter and probably more reliable than a traditional drive in a laptop. Don't expect all SSDs to perform spectacularly, because the C7 motherboard supports only SATA II 3GB/sec. An SSD probably would improve boot speed, but boot time is already a quick 17 seconds. Also, an SSD probably wouldn't produce much

Two items to keep in mind if you decide to upgrade: pick an SSD that is 7mm tall because that is the height of available space for the hard drive, and pick one that supports LBA-48.

performance improvement, because the C7 does not use swap space, and the upgrade to 6GB of RAM negated the need for swap space. Thus, I did not pursue the SSD upgrade.

Two items to keep in mind if you decide to upgrade: pick an SSD that is 7mm tall because that is the height of available space for the hard drive, and pick one that supports LBA-48. According to some Web posts, the C7 requires a 48-bit Logical Block Addressing (LBA) scheme. The LBA specifies the location of data blocks on storage devices. However, upgrading the hard drive may be a moot point if you haven't already purchased an Acer C7. Best Buy now offers the Acer C7 at the same $199 price with a 16GB SSD.

Battery I decided to skip the battery upgrade. The original 4-cell battery (2500mAh 37Wh) provides about three and a half hours of juice, which is plenty of time for intermittent use around the house, browsing the Internet and checking e-mail. People more disconnected from a power source may want to purchase a 6-cell battery. However, more powerful batteries can have form factors that counter the C7's slim dimensions.

Operating System Before upgrading the operating system, it is important to create a recovery image. Acer C7 recovery images are available from Google for download via Windows, Mac and Linux. A recovery image also can be created on the C7. Type "chrome://imageburner" in the C7 Chrome browser address bar to access the Image Burner tool. The tool requests 1) a 4GB or larger USB Flash drive or SD card be inserted and 2) acknowledgment that data on the C7 will be erased. These are the only two steps before the image is burned, which takes about eight minutes. Hold down Esc-F3, and tap the Power button to enter Recovery Mode.

Recovery Mode also provides entry to Developer Mode, which can be used to access system files and create a custom ChromeOS via a Bourne shell. Enter Recovery Mode first, and

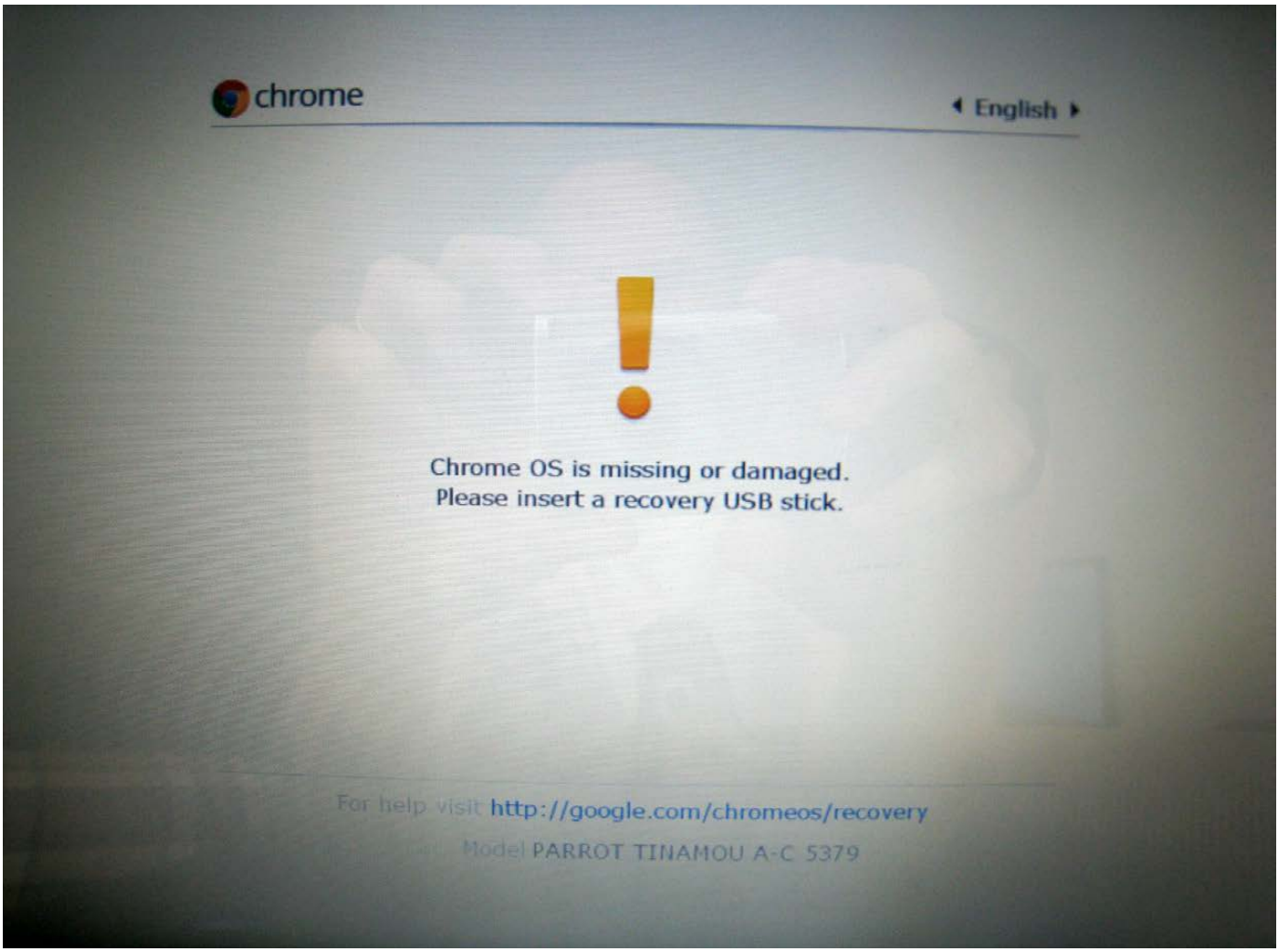


Figure 4. ChromeOS Missing or Damaged Screen

then press Ctl-D to enable Developer Mode. After the ChromeOS comes up, press Ctl-Alt-F2 to access the shell and Ctl-Alt-F1 to switch to the ChromeOS graphical user interface. Note, the C7 will not detect keystrokes while using the shell, so power management will dim and, eventually, turn off the screen. There are methods to disable/tweak power management, but my preference for turning the screen back on has been to switch to the GUI, press a key, and then switch back to the shell.

While in Developer Mode, an "OS verification is OFF" warning screen appears each time the laptop is restarted. Press the spacebar when the warning screen appears to exit Developer Mode and return to Normal Mode.

My time in the shell has been spent installing different Linux distributions, such as Ubuntu. The first approach I

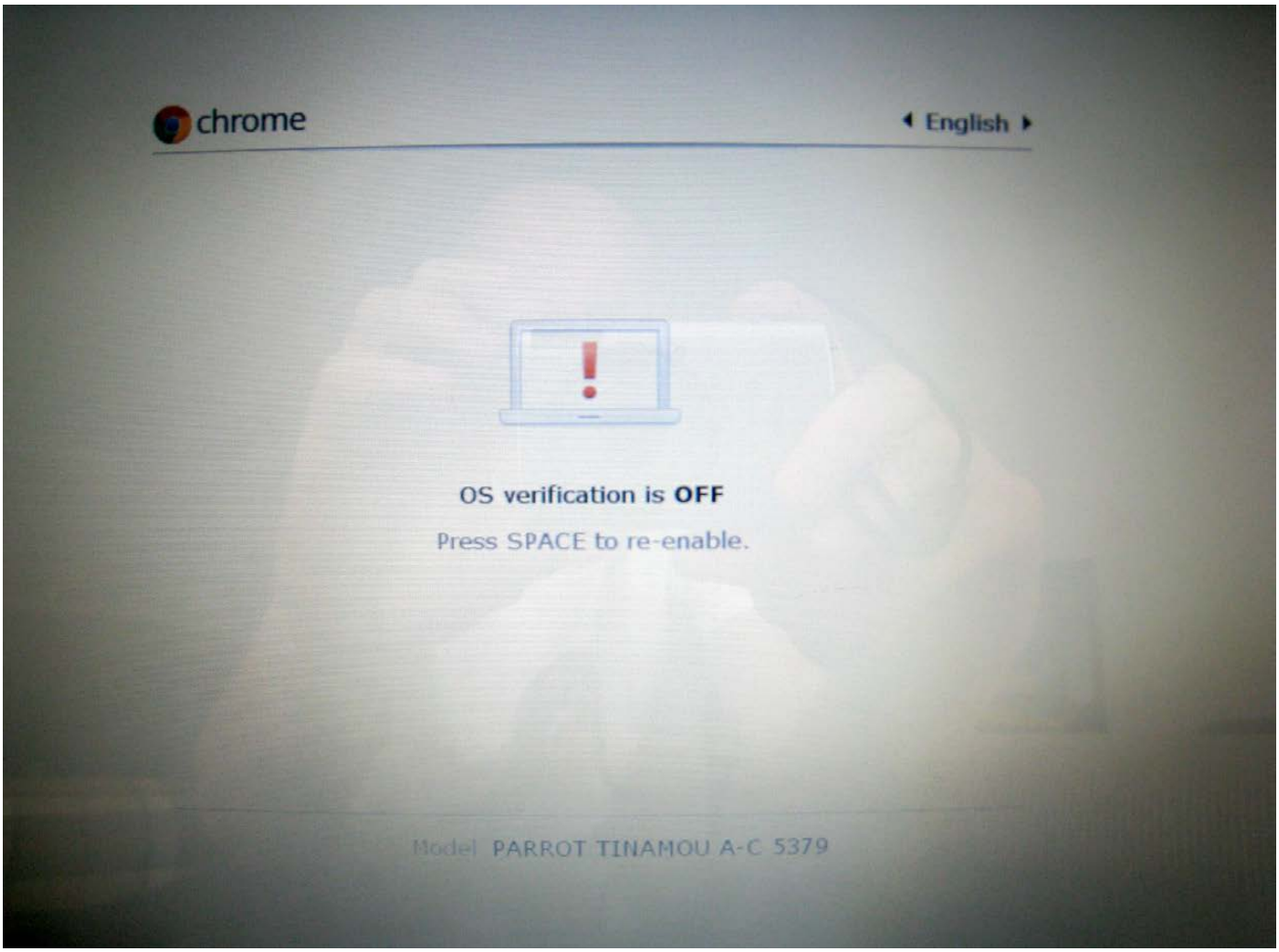


Figure 5. OS Verification Screen

tried was dual-booting Ubuntu from a USB Flash drive using ChrUbuntu 12.04. Jay Lee created ChrUbuntu for the first Cr-48 Chromebook in 2010, and since then he has expanded this special build of Ubuntu 12.04 to other Chromebooks. Unfortunately, it has not been optimized for the Acer C7 yet. I still decided to give it a try. I entered Developer Mode and after about 45 minutes had ChrUbuntu installed on an 8GB USB drive. Ubuntu worked very well, but failed to resume correctly after going to sleep. In many cases, the user interface would come back up and screen pointer would move, but the icons would not work. I jumped to a terminal and also found, using `top`, that polkitd and dbus-daemon were continually consuming >50% of the CPU, and console-kit-daemon and NetworkManager were consuming around 10–15% each. Further, I noticed the C7 ran very hot, and the

Figure 6. ChrUbuntu

fan was quite audible.

Next, I tried to install ChrUbuntu 12.04 on the C7's 320GB hard drive alongside ChromeOS. The install went well, but it still had the same CPU issues and ran hot. Jay Lee released a new ChrUbuntu script on May 31, 2013, that installs any flavor and version of Ubuntu, such as Xubuntu 12.04. I chose the standard install, which is Ubuntu 13.04. Unfortunately, I encountered the same CPU and heat issues as before. I provided this feedback to Jay Lee as requested on his blog.

I entered Recovery Mode and inserted the recovery media to restore the C7's original condition. Note that all data on the C7 will be erased, and there is no warning. However, Google-synced data, such as Bookmarks, will not be affected.

Then I tried to install ubermix. The ubermix distribution is built by educators for students and teachers. It has received a lot of positive press

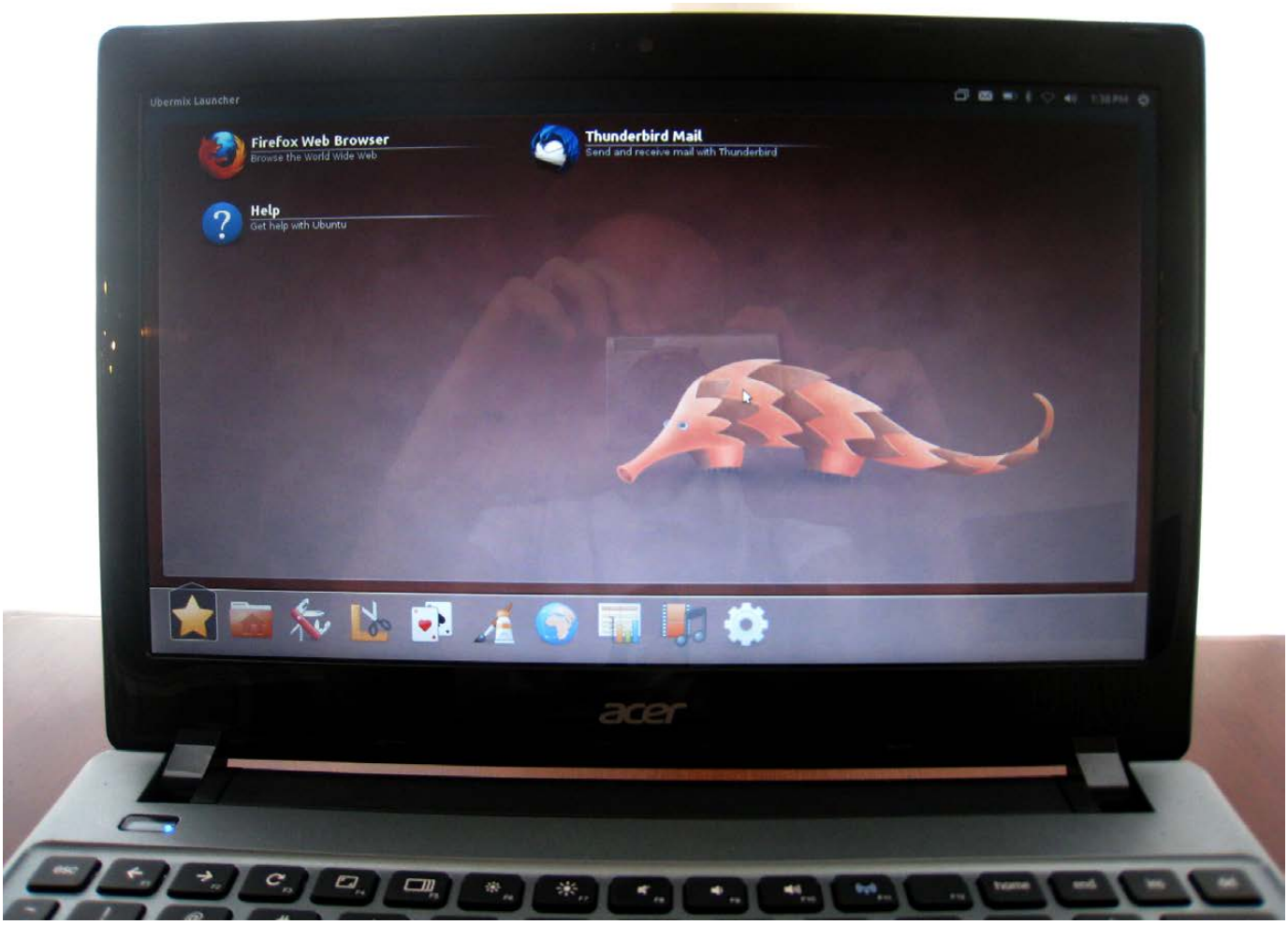


Figure 7. ubermix

and recognition from the education community. The current version uses Ubuntu 12.04, Linux kernel 3.4.0 and GNOME 3.4.2 with a streamlined user interface that is probably too simple for Linux power users. I downloaded the ubermix install key image to my Linux Mint laptop, placed it on a USB thumbdrive and copied it to the Acer C7 via Developer Mode. Install was very similar to ChrUbuntu, because you run the same install script twice, once to repartition the hard drive and second to install the new operating system. The two-step process took about 20 minutes. Happily, everything worked well except some special keys, such as brightness and sound. I followed the simple instructions on the Ubermix on Chromebooks wiki to resolve those issues. Although the ubermix user interface is fairly basic, it does have easy access to more advanced features like system settings, terminal window and so on. The single issue I found

Figure 8. Crouton with Xubuntu

with ubermix was that only 2GB of the installed 6GB memory was recognized, but system performance was still very good.

I reloaded the factory ChromeOS via Recovery Mode and tried to install ChRomium Os UbunTu chrOot enviroNment also known as Crouton. It was created by David Schneider and uses a chroot operation to run Ubuntu on ChromeOS devices. In theory, the chroot approach should work for any Linux distribution because they will all run by using ChromeOS—that is, the distribution is not booted by itself, but rather runs within ChromeOS. Thereby, reboot is not needed to switch between Ubuntu and ChromeOS. I downloaded Crouton, opened a crosh terminal via Ctrl-Alt-T, typed `shell`, and ran this command to install Xubuntu:

```
sudo sh -e ~/Downloads/crouton - xfce
```

The install completed in about 45

minutes. Entering `sudo startxfce4` in the crosh shell started Xubuntu, and everything worked well. I pressed Ctrl-Alt-F1 to get back to ChromeOS. Interestingly, I could not press Ctrl-Alt-F2 to get back to Xubuntu. I had to press Ctrl-Alt-F2, which brought up Developer Mode, and then press Ctrl-Alt-F3 to get back to Xubuntu. Other distributions are available with Crouton, such as Kubuntu, Lubuntu, Xubuntu and the standard Ubuntu. Type `sudo sh -e ~/Downloads/crouton -t help` to see available target distributions and correct nomenclature for downloading.

Conclusion

This journey to the Acer C7 Chromebook began when my wife and I determined we could no longer share our single Linux Mint laptop. Now we have determined we can no longer share our single Chromebook! It is a fantastic laptop that performs great for basic tasks. You can upgrade the C7's hardware, but it's not really necessary. You also can play around with the operating system, which is a lot of fun. There are some real opportunities to learn and grow Linux on the C7 with a custom ChromeOS, ChrUbuntu, ubermix and Crouton. Best of all, after playing around, the C7 can be restored to factory condition in about ten minutes, and everything will be synced back to your configuration automatically.■

Peter Cook is an active-duty US Coast Guard member working in the information technology field. He has been using Linux since 1998 and is a huge Google fan.

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

Resources

Chromium OS: **http://www.chromium.org/chromium-os**

ChrUbuntu: **http://chromeos-cr48.blogspot.com**

ubermix: **http://www.ubermix.org**

Crouton: **https://github.com/dnschneid/crouton**

TIME-SAVING TRICKS ON THE COMMAND LINE

Here are a few simple but very effective tips that will make you lightning fast on the command line.

JANOS GYERIK

I remember the first time a friend of mine introduced me to Linux and showed me how I didn't need to type commands and path names fully—I could just start typing and use the Tab key to complete the rest. That was so cool. I think everybody loves Tab completion because it's something you use pretty much every minute you spend in the shell. Over time, I discovered many more shortcuts and time-saving tricks, many of which I have come to use almost as frequently as Tab completion.

In this article, I highlight a set of tricks for common situations that make a huge difference for me:

- Working in screen sessions: core features that will get you a long way.
- Editing the command line: moving around quickly and editing quickly.
- Viewing files or man pages using `less`.
- E-mailing yourself relevant log snippets or alerts triggered by events.

While reading the article, it would be best to have a terminal window open so you can try using the tips right away. All the tips should work in Linux, UNIX and similar systems without any configuration.

Working in Screen Sessions

Screen has been covered in *Linux Journal* before (see Resources), but to put it simply, screen lets you have multiple "windows" within a single terminal application. The best part is that you can detach and reattach to a running screen session at any time, so you can continue your previous work exactly where you left off. This is most useful when working on a remote server.

Luckily, you really don't need to master screen to benefit from it greatly. You already can enjoy its most useful benefits by using just a few key features, namely the following:

- `screen -R projectx`: reattach to the screen session named "projectx" or create it fresh now.
- Ctrl-a c: create a new window.
- Ctrl-a n: switch to the next window.
- Ctrl-a p: switch to the previous window.
- Ctrl-a 0: switch to the first window; use Ctrl-a 1 for the second window, and so on.

- Ctrl-a w: view the list of windows.
- Ctrl-a d: detach from this screen session.
- `screen -ls`: view the list of screen sessions.

Note: in the above list, "Ctrl-a c" means pressing the Ctrl and a keys at the same time, followed by c. Ctrl-a is called the command key, and all screen commands start with this key sequence.

Let me show all of these in the context of a realistic example: debugging a Django Web site on my remote hosting server, which usually involves the following activities:

- Editing the configuration file.
- Running some commands (performing Django operations).
- Restarting the Web site.
- Viewing the Web site logs.

Of course, I could do all these things one by one, but it's a lot more practical to have multiple windows open for each. I could use multiple real terminal windows, but reopening them every time I need to do this kind of work would be tedious and slow. Screen can make this much faster and easier.

Starting Screen: Before you start screen, it's good to navigate to the directory where you expect to do most of your work first. This is because new windows within screen will all start in that directory. In my example, I first navigate to my Django project's directory, so that when I open new screen windows, the relevant files will be right there in front of me.

There are different ways of starting screen, but I recommend this one:

```
screen -R mysite
```

When you run this the first time, it creates a screen session named "mysite". Later you can use this same command to reconnect to this session again. (The `-R` flag stands for reattach.)

Creating Windows: Now that I'm in screen, let's say I start editing the configuration of the Django Web site:

```
vim mysite/settings.py
```

Let's say I made some changes, and now I want to restart the site. I could exit vim or put it in the background in order to run the command to restart the site, but I anticipate I will need to make further

changes right here. It's easier just to create a new window now, using the screen command Ctrl-a c.

It's easy to create another window every time you start doing something different from your current activity. This is especially useful when you need to change the directory between commands. For example, if you have script files in /some/long/path/scripts and log files in /other/long/path/logs, then instead of jumping between directories, just keep a separate window for each.

In this example, first I started looking at the configuration files. Next, I wanted to restart the Web site. Then I wanted to run some Django commands, and then I wanted to look at the logs. All these are activities I tend to do many times per debugging session, so it makes sense to create a separate window for each activity.

The cost of creating a new window is so small, you can do it without thinking. Don't interrupt your current activity; fire up another window with Ctrl-a c and rock on.

Switching between Windows: The windows you create in screen are numbered starting from zero. You can switch to a window by its number—for example, jump to the first window with Ctrl-a 0, the second window with Ctrl-a 1 and so on. It's also very convenient to switch to the next and previous windows with Ctrl-a n and Ctrl-a p, respectively.

Listing Your Windows: If you're starting to lose track of which window you are in, check the list of windows with Ctrl-a w or Ctrl-a ". The former shows the list of windows in the status line (at the bottom) of the screen, showing the current window marked with a *. The latter shows the list of windows in a more user-friendly format as a menu.

Detaching from and Reattaching to a Session: The best time-saving feature of screen is reattaching to existing sessions. You can detach cleanly from the current screen session with Ctrl-a d. But you don't really need to. You could just as well simply close the terminal window.

The great thing about screen sessions is that whatever way you disconnected from them, you can reattach later. At the end of the day, you can shut down your local PC without closing a remote screen session and come back to it the next day by running the same command you used to start it, as in this example with `screen -R mysite`.

You might have multiple screen sessions running for different purposes. You can list them all with:

```
screen -ls
```

If you are disconnected from screen abruptly, sometimes it may think you are still in an attached state, which will prevent you from reattaching with the usual command `screen -R label`. In that case, you can append a `-D` flag to force detach from any existing connections—for example:

```
screen -R label -D
```

Learning More about Screen: If you want to learn more, see the man page and the links in the Resources section. The built-in cheat sheet of shortcuts also comes handy, and you can view it with Ctrl-a ?.

I also should mention one of screen's competitor: tmux. I chose screen in this article because in my experience, it is more available in systems I cannot control. You can do everything I covered above with tmux as well. Use whichever is available in the remote system in which you find yourself.

Finally, you can get the most out of screen when working on a remote system—for example, over an SSH session. When working locally, it's probably more practical to use a terminal application with tabs. That's not exactly the same thing, but probably close enough.

Editing the Command Line

Many highly practical shortcuts can make you faster and more efficient on the command line in different ways:

- Find and re-run or edit a long and complex command from the history.
- Edit much more quickly than just using the backspace key and retyping text.
- Move around much faster than just using the left- and right-arrow keys.

Finding a Command in the History: If you want to repeat a command you executed recently, it may be easy enough just to press the up-arrow key a few times until you find it. If the command was more than only a few steps ago though, this becomes unwieldy. Very often, it's much more practical to use the Ctrl-r shortcut instead to find a specific command by a fragment.

To search for a command in the past, press Ctrl-r and start typing any fragment you remember from it. As you type, the most recent matching line will appear on the command line. This is an incremental search, which means you can keep typing or deleting letters, and the matched command

will change dynamically.

Let's try this with an example. Say I ran these commands yesterday, which means they are still in my recent history but too far away simply to use the up arrow:

```
...
cd ~/dev/git/github/bashoneliners/
. ~/virtualenv/bashoneliners/bin/
activate
./run.sh pip install --upgrade
django
git push beta master:beta
git push release master:release
git status
...
```

Let's say I want to activate the virtualenv again. That's a hassle to type again, because I have to type at least a few characters at each path segment, even with Tab completion. Instead, it's a lot easier to press Ctrl-r and start typing "activate".

For a slightly more complex example, let's say I want to run a `git push` command again, but I don't remember exactly which one. So I press Ctrl-r and start typing "push". This will match the most recent command, but I actually want the one before that, and I don't remember a better fragment to type. The solution is to press Ctrl-r again, in the middle of my current search, as that jumps to the next matching command.

This is really extremely useful, saving not only the time of typing, but also often the time of thinking too. Imagine one of those long one-liners where you processed a text file through a long sequence of pipes with sed, awk, Perl and whatnot; or an rsync command with many flags, filters and exclusions; or complex loops using "for" and "while". You can bring those back to your command line quickly using Ctrl-r and some fragment you remember from them.

Here are a few other things to note:

- The search is case-sensitive.
- You can abort the search with Ctrl-c.
- To edit the line before running it, press any of the arrow keys.

This trick can be even more useful if you pick up some new habits. For example, when referring to a path you use often, type the absolute path rather than a relative path. That way, the command will be reusable later from any directory.

Moving Around Quickly and Editing Quickly: Basic editing on the

command line involves moving around with the arrow keys and deleting characters with Backspace or Delete. When there are more than only a few characters to move or delete, using these basic keys is just too slow. You can do the same much faster by knowing just a handful of interesting shortcuts:

- Ctrl-w: cut text backward until space.
- Esc-Backspace: cut one word backward.
- Esc-Delete: cut one word forward.
- Ctrl-k: cut from current position until the end of the line.
- Ctrl-y: paste the most recently cut text.

Not only is it faster to delete portions of a line chunk by chunk like this, but an added bonus is that text deleted this way is saved in a register so that you can paste it later if needed. Take, for example, the following sequence of commands:

```
git init --bare /path/to/repo.git
git remote add origin /path/to/repo.git
```

Notice that the second command uses the same path at the end. Instead of typing that path twice, you could copy and paste it from the first command, using this sequence of keystrokes:

1. Press the up arrow to bring back the previous command.
2. Press Ctrl-w to cut the path part: "/path/to/repo.git".
3. Press Ctrl-c to cancel the current command.
4. Type `git remote add origin`, and press Ctrl-y to paste the path.

Some of the editing shortcuts are more useful in combination with moving shortcuts:

- Ctrl-a: jump to the beginning of the line.
- Ctrl-e: jump to the end of the line.
- Esc-b: jump one word backward.
- Esc-f: jump one word forward.

Jumping to the beginning is very useful if you mistype the first words of a long command. You can jump to the beginning much faster than with the left-arrow key.

Jumping forward and backward is very practical when editing the

middle part of a long command, such as the middle of long path segments.

Putting It All Together: A good starting point for learning these little tricks is to stop some old inefficient habits:

- Don't clear the command line with the Backspace key. Use Ctrl-c instead.
- Don't delete long arguments with the Backspace key. Use Ctrl-w instead.
- Don't move to the beginning or the end of the line using the left- and right-arrow keys. Jump with Ctrl-a and Ctrl-e instead.
- Don't move over long terms using the arrow keys. Jump over terms with Esc-b and Esc-f instead.
- Don't press the up arrow 20 times to find a not-so-recent previous command. Jump to it directly with Ctrl-r instead.
- Don't type anything twice on the same line. Copy it once with Ctrl-w, and reuse it many times with Ctrl-y instead.

Once you get the hang of it, you will start to see more and more situations where you can combine these shortcuts in interesting ways and minimize your typing.

Learning More about Command-Line Editing: If you want to learn more, see the bash man page and search for "READLINE", "Commands for Moving" and "Commands for Changing Text".

Viewing Files or man Pages with `less`

The less command is a very handy tool for viewing files, and it's the default application for viewing man pages in many modern systems. It has many highly practical shortcuts that can make you faster and more efficient in different ways:

- Searching forward and backward.
- Moving around quickly.
- Placing markers and jumping to markers.

Searching Forward and Backward: You can search forward for some text by typing / followed by the pattern to search for. To search backward, use ? instead of /. The search pattern can be a basic regular expression. If your terminal supports it, the search results are highlighted with inverted foreground and background colors.

You can jump to the next result by pressing n, and to the previous result by pressing N. The direction of next and previous is relative to the direction of the search itself. That is, when searching forward with /, pressing n will move you forward in the file, and when searching backward with ?, pressing n will move you backward in the file.

If you use the vim editor, you should feel right at home, as these shortcuts work the same way as in vim.

Searching is case-sensitive by default, unless you specify the `-i` flag when starting `less`. When reading a file, you can toggle between case-sensitive and insensitive modes by typing `-i`.

Moving Around Quickly: Here are a couple shortcuts to help you move around quickly:

- g: jump to the beginning of the file.
- G: jump to the end of the file.
- space: move forward by one window.
- b: move backward by one window.
- d: move down by a half-window.
- u: move up by a half-window.

Using Markers: Markers are extremely useful in situations when you need to jump between two or more different parts within the same file repeatedly.

For example, let's say you are viewing a server log with initialization information near the beginning of the file and some errors somewhere in the middle. You need to switch between the two parts while trying to figure out what's going on, but using search repeatedly to find the relevant parts is very inconvenient.

MARKERS ARE EXTREMELY USEFUL IN SITUATIONS WHEN YOU NEED TO JUMP BETWEEN TWO OR MORE DIFFERENT PARTS WITHIN THE SAME FILE REPEATEDLY.

A good solution is to place markers at the two locations so you can jump to them directly. Markers work similarly as in the vim editor: you can mark the current position by pressing m followed by a lowercase letter, and you can jump to a marker by pressing ' followed by the same letter. In this example, I would mark the initialization part

with mi and the part with the error with me, so that I could jump to them easily with 'i and 'e. I chose the letters as the initials of what the locations represent, so I can remember them easily.

Learning More Shortcuts: If you are interested in more, see the man page for the less command. The built-in cheat sheet of shortcuts also comes handy, and you can view it by pressing h.

E-mailing Yourself

When working on a remote server, getting data back to your PC can be inconvenient sometimes—for example, when your PC is NAT-ed and the server cannot connect to it directly with `rsync` or `scp`. A quick alternative might be sending data by e-mail instead.

Another good scenario for e-mailing yourself is to use alerts triggered by something you were waiting for, such as a crashed server coming back on-line or other particular system events.

E-mailing a Log Snippet: Let's say you found the log of errors crashing your remote service, and you would like to copy it to your PC quickly. Let's further assume the relevant log spans multiple pages, so it would be inconvenient to copy and paste from the terminal window. Let's say you can extract the relevant part using a combination of the `head`, `tail` and `grep` commands. You could save the log snippet in a file and run `rsync` on your local PC to copy it, or you could just mail it to yourself by simply piping it to this command:

```
mailx -s 'error logs' me@example.com
```

Depending on your system, the `mailx` command might be different, but the parameters are probably the same: `-s` specifies the subject (optional), the remaining arguments are the destination e-mail addresses, and the standard input is used as the message body.

Triggering an E-mail Alert after a Long Task When you run a long task, such as copying a large file, it can be annoying to wait and keep checking whether it has finished. It's better to arrange to trigger an e-mail to yourself when the copying is complete—for example:

```
the_long_task; date | mailx -s 'job done' me@example.com
```

That is, when the long task has completed, the e-mail command will run. In this example, the message body simply will be the output of the date command. In a real

situation, you probably will want to use something more interesting and relevant as the message—for example `ls -lh` on the file that was copied or even multiple commands grouped together like this:

```
the_long_task; { df -h; tail some.log; } | \
    mailx -s 'job done' me@example.com
```

Triggering an E-mail Alert by Any Kind of Event: Have you ever been in one of the following situations?

- You are waiting for crashed serverX to come back on-line.
- You are tailing a server log, waiting for a user to test your new evolution, which will trigger a particular entry in the log.
- You are waiting for another team to deploy an updated .jar file.

Instead of staring at the screen or checking repeatedly whether the event you are waiting for has happened, you could use this kind of one-liner:

```
while :; do date; CONDITION && break; sleep 300; \
done; MAILME
```

This is essentially an infinite loop, with an appropriate `CONDITION` in the middle to exit the loop and, thus, trigger the e-mail command. Inside the loop, I print the date, just so that I can see the loop is alive, and sleep for five minutes (300 seconds) in each cycle to avoid overloading the machine I'm on.

`CONDITION` can be any shell command, and its exit code will determine whether the loop should exit. For the situations outlined above, you could write the `CONDITION` like this:

- `ping -c1 serverX`: emit a single ping to serverX. If it responds, ping will exit with success, ending the loop.
- `grep pattern /path/to/log`: search for the expected pattern in the log. If the pattern is found, `grep` will exit with success, ending the loop.
- `find /path/to/jar -newer /path/to/jar.marker`: this assumes that before starting the infinite loop, you created a marker file like this: `touch -r /path/to/jar /path/to/jar.marker` in order to save a copy of the exact same timestamp as the .jar file you want to monitor. The `find` command will exit with success

after the jar file has been updated.

In short, don't wait for a long-running task or some external event. Set up an infinite loop, and alert yourself by e-mail when there is something interesting to see.

Conclusion

All the tips in this article are standard features and should work in on Linux, UNIX and similar systems. I have barely scratched the surface here, highlighting the minimal set of features in each area that should provide the biggest bang for your buck. Once you get used to using them, these little tricks will make you a real ninja in the shell, jumping around and getting things done lightning fast with minimal typing.■

Janos Gyerik is a passionate coder and Linux fan. He runs the open-source Web site http://bashoneliners.com, makes simple but handy Android applications, and he recently wrote the book *Bazaar Version Control*. You can contact him on Twitter as @janosgyerik, or via his Web site: http://janosgyerik.com.

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

Resources

Type `man screen`.

Type `man bash`, and search for "READLINE", "Commands for Moving" and "Commands for Changing Text".

Type `man less`.

Janos Gyerik's Talk on Time-Saving Tips on the Command Line:
https://speakerdeck.com/janosgyerik/time-saving-tricks-on-the-command-line

"Power Sessions with Screen" by Adam Lazur, *LJ*, January 2013:
http://www.linuxjournal.com/article/6340

"Status Messages in Screen" by Kyle Rankin, *LJ*, March 2011:
http://www.linuxjournal.com/article/10950

"Transfer Your Terminal with Screen" (Video) by Shawn Powers:
http://www.linuxjournal.com/video/transfer-your-terminal-screen

Advanced Hard Drive Caching Techniques

Enhance the longevity of your slower mechanical hard drive and limit its power consumption by caching data to SSD or RAM.

PETROS KOUTOUPIS

With the introduction of the solid-state Flash drive, performance came to the forefront for data storage technologies. Prior to that, software developers and server administrators needed to devise methods for which they could increase I/O throughput to storage, most of which resulted in low capacity caching to random access memory (RAM) or a RAM drive. Although not as fast as RAM, the Flash drive was almost a dream come true, but it had its limitations—one of which was its low capacities packaged in the NAND-based chips. The traditional spinning disk drive provided users' desired capacities but lacked in speedy accessibility. Even with the 6Gb SATA protocol, sequential data access at best performed at approximately 150MB per second (or MB/s) for both read and write operations, while random access varied between 2–5MB/s as the seeking across multiple sectors laid out in multiple tracks across multiple spinning platters proved to be an extremely disruptive bottleneck. The solid-state drive (SSD) with no movable components significantly decreased these access latencies, thus rendering this bottleneck almost nonexistent.

Even today, the consumer SSD cannot compare to the capacities provided by the magnetic hard disk drive (or HDD), which is why in this article I intend to introduce readers to proven methods for obtaining near SSD performance with the traditional HDD. Multiple open-source projects exist that can achieve this, all but one of which utilizes an SSD as a caching node, and the other caches to RAM. The device drivers I cover here are dm-cache, FlashCache and the RapidDisk/RapidCache suite; I also briefly discuss bcache and EnhanceIO.

NOTE: To build the kernel modules shown in this article, you need to have either the full kernel source or the kernel headers installed for your current kernel image revision.

In my examples, I am using a commercial SATA III (6Gbps) SSD with an average performance of the following:

- Sequential read: 231MB/s
- Sequential write: 74MB/s
- Random read: 230MB/s
- Random write: 72MB/s

This SSD provides the caching layer for a slower mechanical SATA III HDD

that performs at the following:

- Sequential read: 115MB/s
- Sequential write: 72MB/s
- Random read: 2MB/s
- Random write: 2MB/s

In my environment, the SSD is labeled as /dev/sdb, and the HDD is /dev/sda3. These are non-intrusive transparent caching solutions intended to achieve the performance benefits of SSDs. They can be added and removed to existing storage targets without issue or data loss (assuming that all cached data has been flushed to disk successfully). Also, all the examples here showcase a write-back caching scheme with the exception of RapidCache, which instead will be used in write-through mode. In write-back mode, newly written data is cached but not immediately written to the destination target. Write-through mode always will write new data to the target while still maintaining it in cache for future reads.

NOTE: The benchmarks shown here were obtained by using FIO, a file I/O benchmarking and test tool designed for data storage technologies. It is maintained by Linux kernel developer Jens Axboe. Unless noted otherwise, all captured I/O is written at the typical 4KB page size, asynchronously to the storage target 32 transfers at a time (that is, queue depth).

dm-cache

dm-cache has been around for quite some time—at least since 2006. It originally made its debut as a research project developed by Dr Ming Zhao through his summer internship at IBM research. The dm-cache module just recently was integrated into the Linux kernel tree as of version 3.9. Whether you choose to enable it in a recently downloaded kernel or compile it from the official project site, the results will be the same. To load the module, you need to invoke modprobe or insmod:

```
$ sudo modprobe dm-cache
```

Now that the module is loaded, you need to inform that module about which drive to point to for the cache and which to point to for the destination. The dm-cache project site provides a Perl script to simplify this process called dmc-setup.pl. For example, if I wanted to use the entire SSD in write-back caching mode with a 4KB block size, I would type:

```
$ sudo perl dmc-setup.pl -o /dev/sda3 -c /dev/sdb -n cache -b 8 -w
```

This script is a wrapper to the equivalent dmsetup command below:

```
$ echo 0 20971520 cache /dev/sda3 /dev/sdb 0 8 65536 16 1 |
➥dmsetup create cache
```

The dm-cache documentation hosted on the project site provides details on each parameter field, so I don't cover them here.

You may notice that in both examples, I named the mapping to both drives "cache". So, when I need to access the drive mapping, I must refer to it as "cache".

The following mapping passes all data requests to the caching driver, which in turn performs the necessary magic to process the requests either by handling it entirely out of cache or both the cache and the slower device:

```
$ ls -l /dev/mapper
total 0
lrwxrwxrwx 1 root root       7 Jun 30 12:10 cache -> ../dm-0
crw------- 1 root root 10, 236 Jun 30 11:52 control
```

Just like with any other device-mapper-enabled target, I also can pull up detailed mapping data:

```
$ sudo dmsetup status cache
0 20971520 cache stats: reads(83), writes(0),
➥cache hits(0, 0.0),replacement(0), replaced dirty blocks(0)
$ sudo dmsetup table cache
0 20971520 cache conf: capacity(256M), associativity(16),
➥block size(4K), write-back
```

If the target drive already is formatted with data on it, you just need to mount it; otherwise, format it to your specified filesystem:

```
$ sudo mke2fs -F /dev/mapper/cache
```

Remember, these solutions are non-intrusive, so if you have existing data that needs to remain on that disk drive, skip the above step and go straight to mounting it for data accessibility:

```
$ sudo mount /dev/mapper/cache /mnt/cache
$ df|grep cache
/dev/mapper/cache  10321208 1072632   8724288  11% /mnt/cache
```

Using a benchmarking utility, the numbers will vary. On read operations, it is wholly dependent on whether the desired data resides in cache or whether the module needs to retrieve it from the slower disk. On write operations, it depends on the Flash technology itself, and whether it needs to go through a typical programmable erase (PE) cycle to write the new data. Regardless of this, the random read/write access to the slower drive has

FlashCache is a project developed and maintained by Facebook. It was inspired by dm-cache. Much like dm-cache, it too is built from the device-mapper framework.

been increased significantly:

- Sequential read: 105MB/s
- Sequential write: 50MB/s
- Random read: 67MB/s
- Random write: 51MB/s

You can continue monitoring the cache status by typing:

```
$ sudo dmsetup status cache
0 20971520 cache stats: reads(301319), writes(353216),
 ➥cache hits(24485, 0.3),replacement(345972),
 ➥replaced dirty blocks(92857)
```

To remove the cache mapping, unmount the drive and invoke dmsetup:

```
$ sudo umount /mnt/cache
$ sudo dmsetup remove cache
```

FlashCache

FlashCache is a project developed and maintained by Facebook. It was inspired by dm-cache. Much like dm-cache, it too is built from the device-mapper framework. It currently is hosted on GitHub and can be cloned from there. The repository encompasses the kernel module and administration utilities. Once built and installed, load the kernel module and in a similar fashion to the previous examples, create a mapping of the SSD and HDD:

```
$ sudo modprobe flashcache
$ sudo flashcache_create -p back -b 8 cache /dev/sdb /dev/sda3
cachedev cache, ssd_devname /dev/sdb, disk_devname /dev/sda3
 ➥cache mode WRITE_BACK block_size 8, md_block_size 8,
 ➥cache_size 0
FlashCache metadata will use 223MB of your 3944MB main memory
```

The flashcache_create administration utility is similar to the dmc-setup.pl Perl script used for dm-cache. It is a wrapper utility designed to simplify the dmsetup process. As with the dm-cache module, once the mapping has been created, you can view

mapping details by typing:

```
$ sudo dmsetup table cache
0 20971520 flashcache conf:
    ssd dev (/dev/sdb), disk dev (/dev/sda3) cache mode(WRITE_BACK)
    capacity(57018M), associativity(512), data block size(4K)
     ➥metadata block size(4096b)
    skip sequential thresh(0K)
    total blocks(14596608), cached blocks(83), cache percent(0)
    dirty blocks(0), dirty percent(0)
    nr_queued(0)
Size Hist: 4096:83
$ sudo dmsetup status cache
0 20971520 flashcache stats:
    reads(83), writes(0)
    read hits(0), read hit percent(0)
    write hits(0) write hit percent(0)
    dirty write hits(0) dirty write hit percent(0)
    replacement(0), write replacement(0)
    write invalidates(0), read invalidates(0)
    pending enqueues(0), pending inval(0)
    metadata dirties(0), metadata cleans(0)
    metadata batch(0) metadata ssd writes(0)
    cleanings(0) fallow cleanings(0)
    no room(0) front merge(0) back merge(0)
    disk reads(83), disk writes(0) ssd reads(0) ssd writes(83)
    uncached reads(0), uncached writes(0), uncached IO requeue(0)
    disk read errors(0), disk write errors(0) ssd read errors(0)
     ➥ssd write errors(0)
    uncached sequential reads(0), uncached sequential writes(0)
    pid_adds(0), pid_dels(0), pid_drops(0) pid_expiry(0)
```

Mount the mapping for file accessibility:

```
$ sudo mount /dev/mapper/cache /mnt/cache
```

Using the same benchmarking utility, observe the differences between FlashCache and the previous module:

- Sequential read: 284MB/s
- Sequential write: 72MB/s
- Random read: 284MB/s
- Random write: 71MB/s

The numbers look more like the native SSD performance. However, I want to note that this article is not intended to prove that one solution performs better than the other, but instead to enlighten readers of the many methods you can use to accelerate data access to existing and slower configurations.

To unmount and remove the drive mapping, type the following in the terminal:

```
$ sudo umount /mnt/cache
$ sudo dmsetup remove /dev/mapper/cache
```

RapidDisk and RapidCache

Currently at version 2.9, RapidDisk is an advanced Linux RAM disk whose features include the capabilities to allocate RAM dynamically as a block device, use

it as standalone disk drives, or even map it as caching nodes to slower local disk drives via RapidCache (the latter of which was inspired by FlashCache and uses the device-mapper framework). RAM is being accessed to handle the data storage by allocating memory pages as they are needed. It is a volatile form of storage, so if power is removed or if the computer is rebooted, all data stored within RAM will not be preserved. This is why the RapidCache module was designed to handle only read-through/write-through caching, which means that whatever is intended to be written to the slower storage device will be cached to RapidCache and written immediately to the hard drive. And, if data is being requested from the hard drive and it does not pre-exist in the RapidCache node, it will read the data from the slower device and then cache it to the RapidCache node. This method will retain the same write performance as the hard drive, but significantly increase sequential and random access read performance to cached data.

Once the package, which consists of two kernel modules and an administration utility, is built and installed, you need to insert the modules by typing the following on the command line:

```
$ sudo modprobe rxdsk
$ sudo modprobe -r rxdsk
```

Let's assume that you're running on a computer that contains 4GB of RAM, and you confidently can say that at least 1GB of that RAM is never used by the operating system and its applications. Using RapidDisk to create a RAM drive of 1GB in size, you would type:

```
$ sudo rxadm --attach 1024
```

Remember, RapidDisk will not pre-allocate this storage. It will allocate RAM only as it is used.

A quick benchmark test of just the RAM drive produces some overwhelmingly fast results with 4KB I/O transfers:

- Sequential read: 1.6GB/s
- Sequential write: 1.6GB/s
- Random read: 1.3GB/s
- Random write: 1.1GB/s

It produces the following with 1MB I/O transfers:

- Sequential read: 4.9GB/s
- Sequential write: 4.3GB/s
- Random read: 4.9GB/s
- Random write: 4.0GB/s

Impressive, right? To utilize such a speedy RAM drive as a caching node to a slower drive, a mapping must be created, where /dev/rxd0 is the node used to access the RAM drive, and /dev/mapper/rxc0 is the node used to access the mapping of the two drives:

```
$ sudo rxadm --rxc-map rxd0 /dev/sda3 4
```

You can get a list of attached devices and mappings by typing:

```
$ sudo rxadm --list
rxadm 2.9
Copyright 2011-2013 Petros Koutoupis

List of rxdsk device(s):

 RapidDisk Device 1: rxd0
    Size: 1073741824

List of rxcache mapping(s):

 RapidCache Target 1: rxc0
0 20971519 rxcache conf:
    rxd dev (/dev/rxd0), disk dev (/dev/sda3) mode (WRITETHROUGH)
    capacity(1024M), associativity(512), block size(4K)
    total blocks(262144), cached blocks(0)
 Size Hist: 512:663
```

As with the previous device-mapper-based solutions, you even can list detailed information of the mapping by typing:

```
$ sudo dmsetup table rxc0
0 20971519 rxcache conf:
    rxd dev (/dev/rxd0), disk dev (/dev/sda3) mode (WRITETHROUGH)
    capacity(1024M), associativity(512), block size(4K)
    total blocks(262144), cached blocks(0)
 Size Hist: 512:663

$ sudo dmsetup status rxc0
0 20971519 rxcache stats:
    reads(663), writes(0)
    cache hits(0) replacement(0), write replacement(0)
    read invalidates(0), write invalidates(0)
    uncached reads(663), uncached writes(0)
    disk reads(663), disk writes(0)
    cache reads(0), cache writes(0)
```

Format the mapping if needed and mount it:

```
$ sudo mount /dev/mapper/rxc0 /mnt/cache
```

A benchmark test produces the following results:

- Sequential read: 794MB/s

- Sequential write: 70MB/s
- Random read: 901MB/s
- Random write: 2MB/s

Notice that the write performance is not very great, and that's because it is not meant to be. Write-through mode promises only faster read performance of cached data and consistent write performance to the original drive. The read performance, however, shows significant improvement when accessing cached data.

To remove the mapping and detach the RAM drive, type the following:

```
$ sudo umount /mnt/cache
$ sudo rxadm --rxc-unmap rxc0
$ sudo rxadm --detach rxd0
```

Other Solutions Worth Mentioning

bcache: bcache is relatively new to the hard drive caching scene. It offers all the same features and functionalities as the previous solutions with the exception of its capability to map one or more SSDs as the cache for one or more HDDs instead of one volume to one volume. The project's maintainer does, however, tout its superiority over the other solutions when it comes to data access performance from the cache. From what I can tell, bcache is unlike the previous solutions where it does not rely on the device-mapper framework and instead is a standalone module. At the time of this writing, it is set to be integrated into release 3.10 of the Linux kernel tree. Unfortunately, I haven't had the opportunity or the appropriate setup to test bcache. As a result, I haven't been able to dive any deeper into this solution and benchmark its performance.

EnhanceIO: EnhanceIO is an SSD caching solution produced by STEC, Inc., and hosted on GitHub. It was greatly inspired by the work done by Facebook for FlashCache, and although it's open-source, a commercial version is offered by the company for those seeking additional support. STEC did not simply modify a few lines of code of FlashCache and republish it. Instead, STEC rewrote the write-back caching logic while also improving other areas, such as memory footprint, failure handling and more. As with bcache, I haven't had the opportunity to install and test EnhanceIO.

Summary

These solutions are intended to

provide users with near SSD speeds and HDD capacities at a significantly reduced cost. From the data center to your home office, these solutions can be deployed almost anywhere. They also can be tuned to operate more appropriately in their intended environments. Some of them even offer a variety of caching algorithm options, such as Least Recently Used (LRU), Most Recently Used (MRU), hybrids of the two or just a simple first-in first-out (FIFO) caching scheme. The first three options can be expensive regarding performance, as they require the tracking of cached data sets for what has been accessed and how recently in order to determine whether to discard it. FIFO, however, functions as a circular buffer in which the oldest cached data set will be discarded first. With the exception of RapidCache, the SSD-focused modules also preserve metadata of the cache to ensure that any disruptions, including power cycles/outages, don't compromise the integrity of the data.■

Petros Koutoupis is a full-time Linux kernel, device driver and application developer for embedded and server platforms. He has been working in the data storage industry for more than eight years and enjoys discussing the same technologies.

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

Resources

dm-cache: **http://visa.cs.fiu.edu/tiki/dm-cache**

FlashCache: **https://github.com/facebook/flashcache**

EnhanceIO: **https://github.com/stec-inc/EnhanceIO**

bcache: **http://bcache.evilpiepirate.org**

RapidDisk: **http://www.rapiddisk.org**

FIO Git Repository: **http://git.kernel.dk/?p=fio.git;a=summary**

Wikipedia Page on Caching Algorithms: **http://en.wikipedia.org/wiki/Cache_algorithms**

MANAGE YOUR CONFIGS with vcsh

Stop making tarballs of your configuration directories—vcsh gives you source control and syncing.

Bill Childers

If you're anything like me (and don't you want to be?), you probably have more than one Linux or UNIX machine that you use on a regular basis. Perhaps you've got a laptop and a desktop. Or, maybe you've got a few servers on which you have shell accounts. Managing the configuration files for applications like mutt, Irssi and others isn't hard, but the administrative overhead just gets tedious, particularly when moving from one machine to another or setting up a new machine.

Some time ago, I started using Dropbox to manage and synchronize my configuration files. What I'd done was create several folders in Dropbox, and then when I'd set up a new machine, I'd install Dropbox, sync those folders and create symlinks from the configs in those directories to the desired configuration file in my home directory. As an example, I'd have a directory called Dropbox/conf/mutt, with my .muttrc file inside that directory. Then, I'd create a symlink like `~/.muttrc -> Dropbox/conf/mutt/.muttrc`. This worked, but it quickly got out of hand and became a major pain in the neck to maintain. Not only did I have to get Dropbox working on Linux, including my command-line-only server machines, but I also had to ensure that I made a bunch of symlinks in just the right places to make everything work. The last straw was when I got a little ARM-powered Linux machine and wanted to get my configurations on it, and realized that there's no ARM binary for the Dropbox sync dæmon. There had to be another way.

...and There Was Another Way

It turns out I'm not the only one who's struggled with this. vcsh developer Richard Hartmann also had this particular itch, except he came up with a way to scratch it: vcsh. vcsh is a script that wraps both git and mr into an easy-to-use tool for configuration file management.

So, by now, I bet you're asking, "Why are you using git for this? That sounds way too complicated." I thought something similar myself, until I actually started using it and digging in. Using vcsh has several advantages, once you get your head around the workflow. The first and major advantage to using vcsh is that all you really need is git, bash and mr—all of which are readily available (or can be built relatively easily)—so there's no proprietary dæmons or services required. Another advantage of using vcsh is that it leverages git's

workflow. If you're used to checking in files with git, you'll feel right at home with vcsh. Also, because git is powering the whole system, you get the benefit of having your configuration files under version control, so if you accidentally make an edit to a file that breaks something, it's very easy to roll back using standard git commands.

Let's Get Started!

I'm going to assume you're on Ubuntu 12.04 LTS or higher for this, because it makes installation easy. A simple `sudo apt-get install vcsh mr git` will install vcsh and its dependencies. If you're on another Linux distro, or some other UNIX derivative, you may need to check out vcsh and mr, and then build git if it's not packaged. I'm also going to assume you've got a working git server installed on another machine, because vcsh really shines for helping keep your configs synchronized between machines.

Once you've installed vcsh and its dependencies, it's time to start using vcsh. Let's take a fairly common config file that most everyone who's ever used a terminal has—the config file for vim. This file lives in your home directory, and it's called .vimrc. If you've used vim at all before, this file will be here. I'm going to show you how to get it checked into a git repository that is under vcsh's control.

First, run the following command to initialize vcsh's git repository for vim:

```
bill@test:~$ vcsh init vim
vcsh: info: attempting to create '/home/bill/.config/vcsh/repo.d'
vcsh: info: attempting to create '/home/bill/.gitignore.d'
Initialized empty Git repository in
➥/home/bill/.config/vcsh/repo.d/vim.git/
```

I like to think of the "fake git repos" that vcsh works with to be almost like chroots (if you're familiar with that concept), as it makes things easier to work with. You're going to "enter a chroot", in a way, by telling vcsh you want to work inside the fake git repo for vim. This is done with this command:

```
bill@test:~$ vcsh enter vim
```

Now, you're going to add the file .vimrc to the repository you created above by running the command:

```
bill@test:~$ git add .vimrc
```

You're using normal git here, but inside the environment managed by vcsh. This is a design feature of vcsh to make it function very similarly to git.

Now that your file's being tracked by the git repository inside vcsh, let's

commit it by running the following git-like command:

```
bill@test:~$ git commit -m 'Initial Commit'
master (root-commit) bc84953 Initial Commit
 Committer: Bill Childers bill@test.home
 1 file changed, 2 insertions(+)
 create mode 100644 .vimrc
```

Now for the really cool part. Just like standard git, you can push your files to a remote repository. This lets you make them available to other machines with one command. Let's do that now. First, you'll add the remote server. (I assume you already have a server set up and have the proper accounts configured. You'll also need a bare git repo on that server.) For example:

```
bill@test:~$ git remote add origin git@gitserver:vim.git
```

Next, push your files to that remote server:

```
bill@test:~$ git push -u origin master
    Counting objects: 3, done.
    Compressing objects: 100% (2/2), done.
    Writing objects: 100% (3/3), 272 bytes, done.
    Total 3 (delta 0), reused 0 (delta 0)
    To git@gitserver:vim.git
    * new branch      master -> master
    Branch master set up to track remote branch master from origin.
bill@test:~$ exit
```

Note the `exit` line at the end. This exits the "vcsh fake git repo". Now your .vimrc file is checked in and copied to a remote server! If there are other programs for which you'd like to check in configurations, like mutt, you simply can create a new repo by running `vcsh init mutt`, and then run through the process all over again, but this time, check your files into the mutt repository.

Move Your Configuration to Another Machine

To sync your configuration to another machine, you just need to install vcsh, git and mr, and then run a similar process as the steps above, except you'll do a git pull from your server, rather than a push. This is because you don't have the .vimrc file you want locally, and you want to get it from your remote git repository.

The commands to do this are:

```
bill@test2:~$ sudo apt-get install vcsh git mr
bill@test2:~$ vcsh enter vim
bill@test2:~$ git remote add origin git@gitserver:vim.git
bill@test2:~$ git pull -u origin master
From gitserver:vim
* branch            master     -> FETCH_HEAD
bill@test2:~$ exit
```

Now you've got your checked-in

.vimrc file on your second host! This process works, but it's a little clunky, and it can become unwieldy when you start spawning multiple repositories. Luckily, there's a tool for this, and it's called mr.

Wrapping It All Up with mr

If you plan on using multiple repositories with vcsh (and you should—I'm tracking 13 repositories at the moment), getting a configuration set up for mr is essential. What mr brings to the table is a way to manage all the repositories you're tracking with vcsh. It allows you to enable and disable repositories simply by adjusting one symlink per repository, and it also gives you the ability to update all your repos simply by running one easy command: `mr up`.

Perhaps the best way to get started using mr is to clone the repo that the vcsh author provides. This is done with the following command:

```
bill@test2:~$ vcsh clone
➥git://github.com/RichiH/vcsh_mr_template.git mr
Initialized empty Git repository in
➥/home/bill/.config/vcsh/repo.d/mr.git/
remote: Counting objects: 19, done.
remote: Compressing objects: 100% (14/14), done.
remote: Total 19 (delta 1), reused 15 (delta 0)
Unpacking objects: 100% (19/19), done.
From git://github.com/RichiH/vcsh_mr_template
* new branch      master     -> origin/master
```

Now that you've got your mr repo cloned, you'll want to go in

Setting up a Remote Git Repo

A quick note on setting up a remote git repo: you'll need to set up passwordless authentication using SSH keys (see Resources for more information). Once you have that going using a "git" user, you simply need to create a git repo as the git user. That's done easily enough, just run the command:

```
git@gitserver:~$ git init --bare vim.git
Initialized empty Git repository in /home/git/vim.git/
```

Your bare repo will be ready for your vcsh client to check in stuff!

and edit the files to point to your setup. The control files for mr live in ~/.config/mr/available.d, so go to that directory:

```
bill@test2:~/.config/mr/available.d$ ls
mr.vcsh  zsh.vcsh
```

Rename the zsh.vcsh file to vim.vcsh, because you're working with vim, and change the repository path to point to your server:

```
bill@test2:~/.config/mr/available.d$ mv zsh.vcsh vim.vcsh
bill@test2:~/.config/mr/available.d$ vi vim.vcsh
[$HOME/.config/vcsh/repo.d/vim.git]
checkout = vcsh clone git@gitserver:vim.git vim
```

Also, edit the mr.vcsh file to point to your server as well:

```
bill@test2:~/.config/mr/available.d$ vi mr.vcsh
[$HOME/.config/vcsh/repo.d/mr.git]
checkout = vcsh clone git@gitserver:mr.git mr
```

The mr tool relies on symlinks from the available.d directory to the config.d directory (much like Ubuntu's Apache configuration, if you're familiar with that). This is how mr determines which repositories to sync. Since you've created a vim repo, make a symlink to tell mr to sync the vim repo:

```
bill@test2:~/.config/mr/available.d$ cd ../config.d
bill@test2:~/.config/mr/config.d$ ls -l
total 0
lrwxrwxrwx 1 bill bill 22 Jun 11 18:14 mr.vcsh ->
 ➥../available.d/mr.vcsh
bill@test2:~/.config/mr/config.d$ ln -s
 ➥../available.d/vim.vcsh vim.vcsh
bill@test2:~/.config/mr/config.d$ ls -l
total 0
lrwxrwxrwx 1 bill bill 22 Jun 11 18:14 mr.vcsh ->
 ➥../available.d/mr.vcsh
lrwxrwxrwx 1 bill bill 23 Jun 11 20:51 vim.vcsh ->
 ➥../available.d/vim.vcsh
```

Now, set up mr to be able to sync to your git server:

```
bill@test2:~/.config/mr/config.d$ cd ../..
bill@test2:~/.config$ vcsh enter mr
bill@test2:~/.config$ ls
mr  vcsh
bill@test2:~/.config$ git add mr
bill@test2:\~/.config$ git commit -m 'Initial Commit'
[master fa4eb18] Initial Commit
Committer: Bill Childers [bill@test2.home]
3 files changed, 4 insertions(+), 1 deletion(-)
create mode 100644 .config/mr/available.d/vim.vcsh
create mode 120000 .config/mr/config.d/vim.vcsh
bill@test2:\~/.config$ git remote add origin git@gitserver:mr.git
fatal: remote origin already exists.
```

Oh no! Why does the remote origin exist already? It's because you cloned the repo from the author's repository.

Remove it, then create your own:

```
bill@test2:~/.config$ git remote show
origin
bill@test2:~/.config$ git remote rm origin
bill@test2:~/.config$ git remote add origin git@gitserver:mr.git
bill@test2:~/.config$ git push -u origin master
Counting objects: 28, done.
Compressing objects: 100% (21/21), done.
Writing objects: 100% (28/28), 2.16 KiB, done.
Total 28 (delta 2), reused 0 (delta 0)
To git@gitserver:mr.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.
bill@test2:~/.config$ exit
```

That's it! However, now that mr is in the mix, all you need to do to set up a new machine is do a `vcsh clone git@gitserver:mr.git mr` to clone your mr repository, then do an `mr up`, and that machine will have all your repos automatically.

Conclusion

vcsh is a very powerful shell tool, and one that takes some time to adapt your thought processes to. However, once you do it, it makes setting up a new machine (or account on a machine) a snap, and it also gives you a way to keep things in sync easily. It's saved me a lot of time in the past few months, and it's allowed me to recover quickly from a bad configuration change I've made. Check it out for yourself!■

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

Resources

vcsh Home Page: **http://github.com/RichiH/vcsh**

mr Home Page: **http://joeyh.name/code/mr**

vcsh Background Slides: **https://raw.github.com/RichiH/talks/slides/2012/fosdem/vcsh/fosdem-2012-vcsh-talk.pdf**

How to Set Up Your Own Git Server: **http://tumblr.intranation.com/post/766290565/how-set-up-your-own-private-git-server-linux**

Set Up Passwordless SSH Key-Based Authentication: **http://askubuntu.com/questions/46930/how-can-i-set-up-password-less-ssh-login**

A BZ Media Event

Big Data gets real at Big Data TechCon!

Discover how to master Big Data from real-world practitioners – instructors who work in the trenches and can teach you from real-world experience!

Come to Big Data TechCon to learn the best ways to:

- Collect, sort and store massive quantities of structured and unstructured data
- Process real-time data pouring into your organization
- Master Big Data tools and technologies like Hadoop, Map/Reduce, NoSQL databases, and more

Over 60 how-to practical classes and tutorials to choose from!

- Learn HOW TO integrate data-collection technologies with analysis and business-analysis tools to produce the kind of workable information and reports your organization needs
- Understand HOW TO leverage Big Data to help your organization today

Keynote

Doug Cutting
Founder of Hadoop

"Big Data TechCon is great for beginners as well as advanced Big Data practitioners. It's a great conference!"
—Ryan Wood, Software Systems Analyst, Government of Canada

"If you're in or about to get into Big Data, this is the conference to go to."
—Jimmy Chung, Manager, Reports Development, Avectra

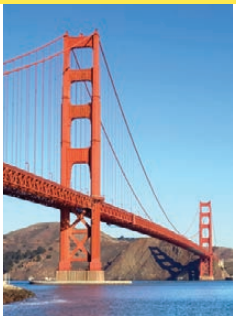
BigData TECHCON

San Francisco

October 15-17, 2013

www.BigDataTechCon.com

The HOW-TO conference for Big Data and IT professionals

EOF

Linux vs. Bullshit

DOC SEARLS

What's wrong with marketing on-line today will lose to what's been right with Linux all along.

Linux doesn't lie, any more than gravity lies, or geology lies, or atmosphere lies. Like those other natural things, Linux has no guile, no agenda beyond supporting the entirety of use-space. In rough words, there's no bullshit about it, and that's one reason it gets used. Let me explain.

Prior to discovering Linux (or having it discover me) I was peripherally involved in high-level UNIX debates, while helping Sun Microsystems promote its SPARC microprocessor architecture. At that time (late '80s, early '90s), UNIX was a horse race, and Sun's steed was Sun OS, which was then at V4, as I recall. I attended many meetings at Sun, where representatives of various commercial factions worked toward reconciling Sun OS with AT&T's System V R4 (called SVR4 then). Somehow this led to Solaris, but that was after I had moved on.

What stuck with me from those meetings was that every company involved had a self-interested stake in how UNIX evolved. I suppose that's one reason Linux fascinated me from the moment I first learned about it. As Andrew Morton explained to me in 2005 (**http://www.linuxjournal.com/article/8664**), "On the kernel team we are concerned about the long-term viability and integrity of the code base. We're reluctant to put stuff in for specific reasons where a commercial company might do that."

In other words, no bullshit.

I mean *bullshit* seriously, as does the philosopher Harry Frankfurt in his landmark book *On Bullshit* (Princeton University Press, 2005). Back when I first read the book, I page-flagged this passage:

> Wittegenstein once said that

It should then be no surprise that Linux folk have a heightened tendency to resist advertising, at least in their browsers.

> the following bit of verse by Longfellow could serve him as a motto:
>
> In the elder days of art
> Builders wrought with greatest care
> Each minute and unseen part,
> For the Gods are everywhere.

That's the same point as "Linus's Law" (coined by Eric S. Raymond and named for Linus): "given enough eyeballs, all bugs are shallow". Frankfurt goes on to observe:

> Is the bullshitter by his very nature a mindless slob? Is his product necessarily messy or unrefined? The word *shit* does, to be sure, suggest this. Excrement is not designed or crafted at all; it is merely emitted, or dumped. It may have a more or less coherent shape, or it may not, but it is in any case not *wrought*.
>
> The notion of carefully wrought bullshit involves, then, a certain inner strain. Thoughtful attention to detail requires discipline and objectivity. It entails accepting standards and limitations that forbid the indulgence of impulse or whim. It is this selflessness that, in connection with bullshit, strikes us as inaposite. But in fact, it is not out of the question at all. The realms of advertising and public relations, and the nowadays closely related realm of politics, are replete with instances of bullshit so unmitigated that they can serve among the most indisputable and classic paradigms of the concept.

It should then be no surprise that Linux folk have a heightened tendency to resist advertising, at least in their browsers. Confirmation of this comes via "Ad Blocking, Measured" (**http://clarityray.com/Content/ClarityRay_AdBlockReport.pdf**), a May 2012 research report by ClarityRay (**http://clarityray.com**), a company in the business of thwarting ad blocking. In it they find an "overall rate of ad-blocked impressions in the

In specialized publications like these, ads tend to enhance rather than to diminish editorial content.

US and Europe" of 9.26%. Among six content domains, the ad-blocking rate was lowest in Business & Finance at 6.11% and highest in Tech at 17.79%. Among browsers, the lowest ad-blocking rate was earned by Explorer at 3.86% and the highest by Firefox at 17.81%. Among operating systems, the ad-blocking rate was lowest with iOS at 1.33% and highest with Linux and Ubuntu at 29.04%.

So it also should be no surprise that Microsoft is now turning on Do Not Track by default in Explorer, and Mozilla (parent of Firefox) is battling the DAA (Digital Advertising Alliance, **http://www.aboutads.info**) and the IAB (Internet Advertising Bureau, **http://www.iab.net**) over proposed blocking of third-party cookies. (Apple's Safari already does that.) On the whole, browser makers are on the users' side. (One exception is Google's Chrome, which has no plans at this time for blocking third-party cookies. Google is the biggest player in the on-line advertising business.)

To be fair, both the DAA and the IAB would like advertising to be as wrought as possible, and for consumers to appreciate the good intentions and effects of their business. I know that because I've talked to them about it. Those organizations see themselves, correctly, as advocates for good behavior in a business rife with the opposite. They also know consumer appetite for advertising, such as it is, tends to be highest for that which is most wrought—for example, Super Bowl ads. Among less expensively wrought forms of advertising, there are breeds that also enjoy a degree of demand by consumers. You'll find these in *Linux Journal* and *Vogue*. Subscribers and advertisers both pay for those magazines, and there is a kind of symbiosis in the middle. In specialized publications like these, ads tend to enhance rather than to diminish editorial content.

So this topic is close to home for us here at *Linux Journal*. It's also close to home for me personally, since I labored in the advertising business for much of my adult

life. The "inner strain" of which Frankfurt speaks is one I experienced in that business, and still feel today as I try to make sense of what it has become in the on-line world.

Advertising in the pre-digital days was an obvious business. You knew how an ad got to be where it was, and what it was doing there. Because of that, advertising also carried what economists call a *signal*. That is, a message of sufficiency. In "The Waste in Advertising is the Part that Works" (*Journal of Advertising Research*, December 2004), Tim Ambler and E. Ann Hollier say advertising's base-level purpose—aside from its specific message—is akin to a male peacock's fanned-out tail. It speaks of the company's substance, and the fact that it can afford to advertise.

With today's "adtech" business, the provenance of many ads is unknown. If you see an ad for badger-hide gloves when you're reading the *Daily Feh*, and then again when you whimsically look for vacations on Pluto, it's not clear why that ad is there, or why it's following you around. That's because the ad might be targeted by way of some combination of the following, only the first two of which existed in the old pre-Internet advertising world:

- Advertisers
- Agencies
- Agency Trading Desks
- SSPs (Supply Side Platforms)
- DSPs (Demand Side Platforms)
- RTB (Real Time Bidding)
- Exchanges
- Creative Optimization
- Retargeting
- Verification/Privacy
- Media Planning and Attribution
- Tag Management
- Measurement and Analytics
- Data Suppliers
- DMPs (Data Management Platforms) and Data Aggregators
- Ad Networks—Horizontal, Vertical/Custom, Targeted Networks/AMPs, Performance, Mobile
- Media Management Systems
- Ad Operations
- Ad Servers
- Publisher Tools
- Web Analytics
- Gamification
- Real Time Message/Offers
- Sharing Data/Social Tools

Most of that list comes from Luma Partners' "Lumascapes" (**http://www.lumapartners.com/resource-center/lumascapes-2**):

Customers are represented (I'm not kidding) as empty beakers moving down a conveyor belt at the bottom of this whole thing.

amazing graphical representations of marketing businesses, each sandwiching dozens of companies and categories between first source (such as marketer or advertiser) and the consumer. Many arrows run between collections of companies in one specialty or another. In some cases, the arrows point recursively back toward the source. Randall Rothenberg, IAB's CEO, tells me this whole collection is a "black box" to the consumer and a problem he and others at the IAB want to fix.

Another view of that box comes to us by way of IBM and the research firm Aberdeen, which together diagram "The Big Datastillery" (**http://www.ibmbigdatahub.com/blog/big-datastillery-strategies-accelerate-return-digital-data**). Copy at the top describes it as "Best-in-Class Strategies to Accelerate the Return on Digital Data" and "a revolutionary new appliance to condense terabyte scale torrents of customer, transactional, campaign, clickstream and social media data down to meaningful and actionable insights that boost response rates, conversions and customer value".

Below that is a maze of pipes pouring stuff into a hopper of "Best-in-Class companies" that are "2.8 times more likely than Laggards to incorporate unstructured data into analytical models". The pipes are called:

- Customer Sentiment
- E-mail Metrics
- CRM
- Clickstream Data
- PPC (Pay Per Click)
- SEO Data
- Social Media
- Marketing History
- Ad Impressions
- Transactional Data

Coming out of the hopper are boxes and tanks, connected to more piping. These are accompanied by blocks of text explaining what's going on in that part of the "datastillery". One says "Ability to generate customer

behavioral profile based on real-time analytics". Another says "Ability to optimize marketing offers/Web experience based on buyer's social profile". Another says BIC (Best in Class) outfits "merge customer data from CRM with inline behavioral data to optimize digital experience".

Customers are represented (I'm not kidding) as empty beakers moving down a conveyor belt at the bottom of this whole thing. Into the beakers pipes called "customer interaction optimization" and "marketing optimization" excrete orange and green flows of ones and zeroes. Gas farted upward by customers metabolizing goop fed by the first two pipes is collected by a third pipe called "campaign metrics" and carried to the top of the datastillery, where in liquid form it gets poured back into the hopper. Text over a departing beaker says "137% higher average marketing response rate for Best-in-Class (6.2%) vs. All Others (2.6%)". (The 137% is expressed in type many times larger than the actual response rates.) The reciprocal numbers for those rates are 93.8% and 97.4%—meaning that nearly all the beakers are not responsive, even to Best-in-Class marketing.

In fact, advertising and marketing have always been good at bullshitting themselves. Consider, for example, the old saying (often attributed to John Wanamaker, who is not known to have actually said it) "I know half my advertising is wasted, I just don't know which half." The correct answer is that most of it is wasted, and the industry has known that for the duration. They just don't want to talk about it. And, on the whole, neither do we. Frankfurt explains:

> In fact, people do tend to be more tolerant of bullshit than of lies, perhaps because we are less inclined to take the former as a personal affront. We may seek to distance ourselves from bullshit, but we are more likely to turn away from it with an impatient or irritated shrug than with the sense of violation or outrage that lies often inspire.

Lately, however, people are feeling somewhat more violated, especially by tracking, thanks to Edward Snowden's revelations of how the NSA is spying on everybody. We now know that the Feds and marketing mills are both harvesting massive amounts of personal data without revealing to us what they know, and that the two are actually in cahoots, at least some

We now know that the Feds and marketing mills are both harvesting massive amounts of personal data without revealing to us what they know, and that the two are actually in cahoots, at least some of the time.

of the time. This is especially vexing, because the feds should be the ones protecting us from bad actors, rather than bad actors themselves.

But let's set that stuff aside and just look at the bullshit aspects of the whole thing. How much good is all this data collection and manipulation *actually doing* for its perpetrators? And how much are they also bullshitting themselves?

When I was doing research for *The Intention Economy*, the most important input I got came from Doug Rauch, the retired president of Trader Joe's. One big reason for Trader Joe's success, he told me, is that it minimizes marketing bullshit. It has no loyalty program, no coupons, no discounts and none of the expenses any of those involve, including the cost of running a big data mill. Doug's job as president of the company, he said, was to shop along with customers. Talking in person, in stores, with customers, was his main form of research. One result, he told me, is that there is hardly a product on the shelves at Trader Joe's that isn't influenced directly by customers talking to workers in the store. Trader Joe's also doesn't go to retailing tradeshows, Doug told me, because too much of what goes on at those things is all about manipulating the customer. These manipulations are highly complex and therefore come at high costs to the stores as well. By avoiding this kind of thing, Trader Joe's spares itself the cognitive overhead required to rationalize complicating the living shit out of everything, which is what marketing tends to do—and does now, more than ever, with Big Data. Thanks to Big Data and the perceived need to run big complex marketing mills, the Chief Marketing Officer (CMO)—a title that didn't exist twenty years ago—runs an overhead-fattening operation that

can dwarf the old IT budget.

Meanwhile, only a tiny percentage of the output of these mills is actually useful to us. The rest is cognitive overhead, and we don't like it, even if we appreciate all the free-as-in-beer stuff it pays for.

What we need are fixes on the Linux model: stuff that's useful.

This issue of *Linux Journal* will hit the streets on the seventh anniversary of the month I made it my my mission in life to encourage developers to create tools that make individuals both independent of companies yet better equipped to engage with them. That work is happening (http://blogs.law.harvard.edu/vrm), and I continue to have faith that it will succeed, and improve the way business works for everybody. That faith comes from watching how Linux and other free and open-source developments, by simply being useful, defeat bullshit.■

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

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

Advertiser Index

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

NOVEMBER 3-8, 2013 • WASHINGTON D.C.

Lucky LISA '13

27th Large Installation System Administration Conference

Keynote Address: "Modern Infrastructure: The Convergence of Network, Compute, and Data" by Jason Hoffman, *CTO, Joyent*

Join us for **6 days of practical training** on topics including:

- **SRE University: Non-Abstract Large System Design for Sysadmins** by John Looney, *Google*
- **Root Cause Analysis** by Stuart Kendrick, *Fred Hutchinson Cancer Research Center*
- **PowerShell Fundamentals** by Steven Murawski, *Stack Exchange*
- **Introduction to Chef** by Nathen Harvey, *Opscode*

The **3-day Technical Program** includes:

- Plenaries by Hilary Mason, *bitly,* and Todd Underwood, *Google*
- Invited Talks by industry leaders such as Ariel Tseitlin, *Netflix;* Jeff Darcy, *Red Hat;* Theo Schlossnagle, *Circonus;* Matt Provost, *Weta Digital;* and Jennifer Davis, *Yahoo!*
- Paper presentations, workshops, vendor exhibition, posters, Guru Is In sessions, BoFs, and more!

New for 2013: The LISA Lab Hack Space!

Register by October 15 and save. Additional discounts are available!

www.usenix.org/lisa2013

Sponsored by usenix ASSOCIATION in cooperation with LOPSA