Lua | App Inventor | LTSP | Pure Data | C on

LINUX JOURNAL

Since 1994: The Original Magazine of the Linux Community

HIGHLY
iSCSI
WITH DRB
PACEMA

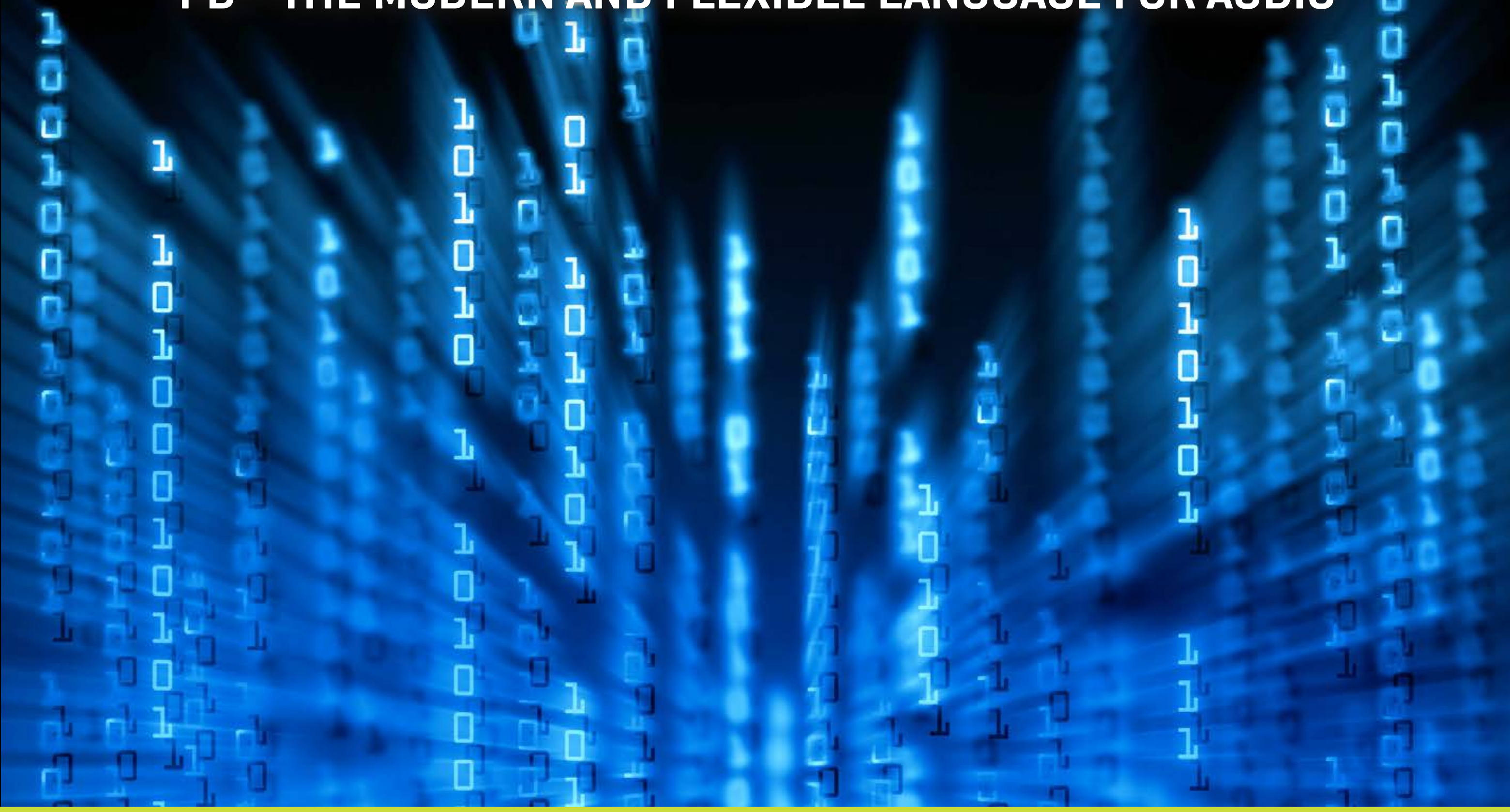
FREE TO SUBSCRIBERS
EPUB, Kindle, Android, iPhone & iPad editions

USE DESIGN FRAMEWORKS TO IMPROVE YOUR SITE

MAY 2012 | ISSUE 217 | www.linuxjournal.com

PROGRAMMING

LUA FOR OBJECT-ORIENTED PROGRAMMING
HOW TO: APP INVENTOR FOR ANDROID
INTRO TO PARALLEL PROGRAMMING WITH C AND PYTHON
PD—THE MODERN AND FLEXIBLE LANGUAGE FOR AUDIO



TRACK DOWN Bandwidth-Hogging Connections with iftop

SCALING LTSP in Large Environments

REVIEWED: ZaReason's Valta X79

If You Use Linux, You Should Be Reading LINUX JOURNAL™

» In-depth information providing a full 360-degree look at featured topics relating to Linux

» Tools, tips and tricks you will use today as well as relevant information for the future

» Advice and inspiration for getting the most out of your Linux system

» Instructional how-tos will save you time and money

Subscribe now for instant access! For only $29.50 per year—less than $2.50 per issue—you'll have access to *Linux Journal* each month as a PDF, in ePub format, on-line and through our Android app. Wherever you go, *Linux Journal* goes with you.

SUBSCRIBE NOW AT:
WWW.LINUXJOURNAL.COM/SUBSCRIBE

SILICON MECHANICS

visit us at www.siliconmechanics.com or call us toll free at 888-352-1173

RACKMOUNT SERVERS   STORAGE SOLUTIONS   HIGH-PERFORMANCE COMPUTING

intel® inside™ Xeon®

Kate rocks. Literally. When she's not coordinating the production of our customers' orders, **she's rocking out** as the lead guitar for her band. But Kate knows she's going to be pretty busy at Silicon Mechanics with the launch of the **Intel® Xeon® Processor E5-2600**.

These new processors offer up to 8 cores, 16 threads, 20 MB of shared L3 cache, and DDR3 memory up to 1600 MT/s. The ring bus architecture and PCIe on die reduces I/O latency. The **Intel Xeon Processor E5-2600** CPUs balance performance, power efficiency, and cost to deliver flexibility and value in dual-processor servers and storage.

Kate is pictured here with just a few of the servers from Silicon Mechanics featuring the **Intel Xeon Processor E5-2600:**

Rackform iServ R4410 – 2U 4-node server with optional InfiniBand

Rackform iServ R335 – 1U server with 16 DDR3 DIMMs and 8 hot-swap SAS/SATA drives

Rackform iServ R346 – 2U server with 16 DDR3 DIMMs, 8 hot-swap SAS/SATA drives, and 6 PCIe 3.0 expansion slots

Rackform iServ R331 – 1U server with 16 DIMMs and 4 hot-swap SAS/SATA drives

When you partner with Silicon Mechanics, you get more than rocking technology. You get an Expert like Kate.

To learn more about Kate's band, Alabaster, see them on Facebook at: https://www.facebook.com/alabastermusic

Expert included.

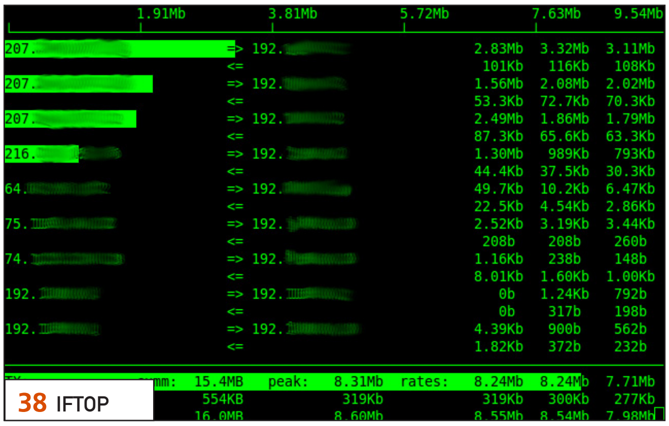
CONTENTS

MAY 2012
ISSUE 217

PROGRAMMING

FEATURES

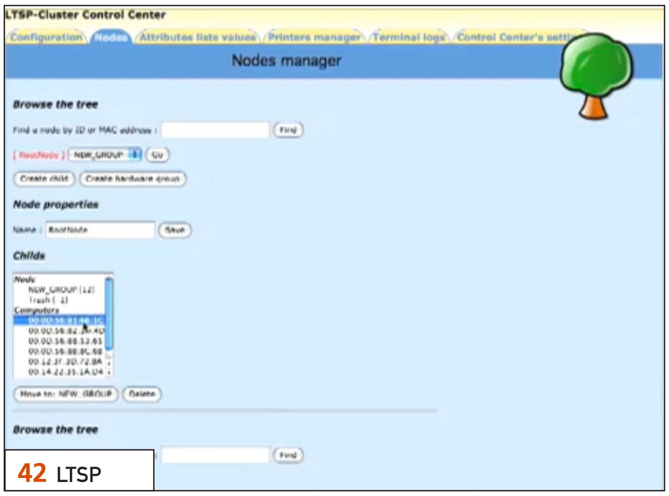
Cover Image © Can Stock Photo Inc. / AlexMax



```
              1.91Mb          3.81Mb          5.72Mb          7.63Mb     9.54Mb
207.[redacted]            => 192.[redacted]              2.83Mb  3.32Mb  3.11Mb
                          <=                              101Kb   116Kb   108Kb
207.[redacted]            => 192.[redacted]              1.56Mb  2.08Mb  2.02Mb
                          <=                             53.3Kb  72.7Kb  70.3Kb
207.[redacted]            => 192.[redacted]              2.49Mb  1.86Mb  1.79Mb
                          <=                             87.3Kb  65.6Kb  63.3Kb
216.[redacted]            => 192.[redacted]              1.30Mb   989Kb   793Kb
                          <=                             44.4Kb  37.5Kb  30.3Kb
64.[redacted]             => 192.[redacted]              49.7Kb  10.2Kb  6.47Kb
                          <=                             22.5Kb  4.54Kb  2.86Kb
75.[redacted]             => 192.[redacted]              2.52Kb  3.19Kb  3.44Kb
                          <=                              208b    208b    260b
74.[redacted]             => 192.[redacted]              1.16Kb   238b    148b
                          <=                             8.01Kb  1.60Kb  1.00Kb
192.[redacted]            => 192.[redacted]                0b    1.24Kb   792b
                          <=                               0b     317b    198b
192.[redacted]            => 192.[redacted]              4.39Kb   900b    562b
                          <=                             1.82Kb   372b    232b

              mm:  15.4MB   peak:  8.31Mb  rates:  8.24Mb  8.24Mb  7.71Mb
                    554KB           319Kb           319Kb   300Kb   277Kb
                   16.0MB          8.60Mb          8.55Mb  8.54Mb  7.98Mb
```

38 IFTOP



LTSP-Cluster Control Center

Nodes manager

Browse the tree

Find a node by ID or MAC address :

[RootNode] NEW_GROUP

Node properties

Name : RootNode

Childs

Node
NEW_GROUP (12)
Trash (-1)
Computers
00:00:56:81:48:1C
00:0D:56:82:20:4D
00:0D:56:88:52:65
00:0D:56:88:8C:68
00:12:3F:3D:72:8A
00:14:22:35:1A:D4

Browse the tree

42 LTSP

54 ZAREASON'S VALTA X79

LINUX JOURNAL (ISSN 1075-3583) is published monthly by Belltown Media, Inc., 2121 Sage Road, Ste. 310, Houston, TX 77056 USA. Subscription rate is $29.50/year. Subscriptions start with the next issue.

LINUX JOURNAL™

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

ENJOY:

Timely delivery

Off-line reading

Easy navigation

Phrase search and highlighting

Ability to save, clip and share articles

Embedded videos

Android & iOS apps, desktop and e-Reader versions

SUBSCRIBE TODAY!

LINUX JOURNAL

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

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

Proofreader	Geri Gale

Publisher	Carlie Fairchild publisher@linuxjournal.com
Advertising Sales Manager	Rebecca Cassity rebecca@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

Editorial Advisory Panel
Brad Abram Baillio • Nick Baronian • Hari Boukis • Steve Case
Kalyana Krishna Chadalavada • Brian Conner • Caleb S. Cullen • Keir Davis
Michael Eager • Nick Faltys • Dennis Franklin Frey • Alicia Gibb
Victor Gregorio • Philip Jacob • Jay Kruizenga • David A. Lane
Steve Marquez • Dave McAllister • Carson McDonald • Craig Oda
Jeffrey D. Parent • Charnell Pugsley • Thomas Quinlan • Mike Roberts
Kristin Shoemaker • Chris D. Stark • Patrick Swartz • James Walker

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

Subscriptions
E-MAIL: subs@linuxjournal.com
URL: www.linuxjournal.com/subscribe
MAIL: PO Box 16476, North Hollywood, CA 91615-9911 USA

LINUX is a registered trademark of Linus Torvalds.

iXsystems Servers + Intel® Xeon® Processor E5-2600 Family =

Unparalleled performance density

iXsystems is pleased to present a range of new, **blazingly fast servers** based on the Intel® Xeon® Processor E5-2600 family and the Intel® C600 series chipset.

The Intel® Xeon® Processor E5-2600 Family employs a new microarchitecture to boost performance by up to 80% over previous-generation processors. The performance boost is the result of a combination of technologies, including Intel® Integrated I/O, Intel® Data Direct I/O Technology, and Intel® Turbo Boost Technology.

The iXR-1204+10G features Dual Intel® Xeon® E5-2600 Family Processors, and packs up to 16 processing cores, 768GB of RAM, and dual onboard 10GigE NICs in a single unit of rack space. The robust feature set of the iXR-1204+10G makes it suitable for clustering, high-traffic webservers, virtualization, and cloud computing applications.

For computation and throughput-intensive applications, iXsystems now offers the iXR-22x4IB. The iXR-22x4IB features four nodes in 2U of rack space, each with dual Intel® Xeon® E5-2600 Family Processors, up to 256GB of RAM, and a 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.

iXR-1204+10G

- Dual Intel® Xeon® E5-2600 Family Processors
- 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® E5-2600 Family Processors 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

E5-2600

HIGH Throughput & INCREDIBLE Performance Density

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

IXR-22x4IB

intel inside Xeon®

iXsystems®

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

SHAWN POWERS

Rubies, Pythons and Perls!

It may sound like a new *Indiana Jones* movie or possibly a cheesy platform-style video game from the 1990s, but the title of this column actually refers to our focus this month—programming! Not that there's anything wrong with daring adventures in remote locations, it's just that all the red tape can be overwhelming. You know that somewhere there was a college intern filing environmental impact study reports, negotiating work visas for multi-country searches and adding venomous predator riders on the insurance policies. Dr Jones just grabbed his hat and found the treasure!

Here in the real world, we prefer to create our own treasures. That's where this issue comes into play. Reuven M. Lerner is actually a bit of a rebel this month, and in an ironic twist, he focuses on Web design frameworks. The worlds of Web design and programming certainly overlap, so we'll give him a pass. Dave Taylor wraps up his *Words With Friends* series by demonstrating how to calculate word point values. Dave's articles always are fun for me, because scripting is the only sort of programming I ever do (usually out of sysadmin need), and watching him create stuff that I understand is fun.

Kyle and I are system administrators by trade, so it's not a surprise that our columns this month show it. Kyle describes how to use iftop, which is like the monitoring tool top, except for bandwidth usage. If you are struggling to find where all your bandwidth is going, iftop can be invaluable, and in his article, you'll learn how to use it. For my column, it's quite clear where the bandwidth is going—to the thin clients! This month, I finish my three-part series on LTSP and discuss how to scale your thin-client environment. As with most complex problems, there isn't a single solution to LTSP scaling. I talk about a handful of methods and help you choose the best option for your environment.

I'm sure you're thinking, "I thought this was the programming issue!" Rest assured, it is. Alejandro Segovia explores Lua, a multiparadigm programming language. No, he doesn't take us to a Luau, with grass skirts and coconuts, but in his words, he "presents a reusable

With its drag-and-drop interface, App Inventor nips at the learning curve a bit for Android programming.

mechanism through which you can implement an object-oriented model using Lua's built-in constructs." On his heels, Amit Saha follows with a great article on using App Inventor to program for Android. With its drag-and-drop interface, App Inventor nips at the learning curve a bit for Android programming. Whether you want to create a fancy GUI application or a service that runs in the background, Amit's article will be very useful.

Dave Phillips introduces Pure Data (Pd), which is a graphic patching environment for audio production. If that sounds confusing, it's a bit like an old modular patching synthesizer on steroids. If even that sounds confusing, you should read the article. Dave discusses using Pd to program audio. It's pretty cool stuff, and you won't want to miss it.

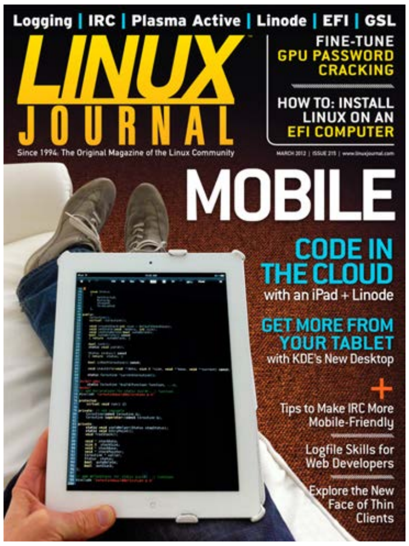
Amit is back for a second time with an article on parallel programming. It seems everything from pocket watches to Space Shuttles (oh, how we miss Space Shuttles...) have multicore processors, but not all programs take advantage of them. Amit shows how to parallel program in Python and C. Because every modern CPU has more than one core, it makes sense to understand how parallel processing works.

Every programmer needs reliable hardware on which to run their programs. We included Florian Hass' article on highly available iSCSI storage this month, because even the best code will fail if the server fails. Florian shows how to use DRBD and Pacemaker to keep iSCSI available even when systems fail. It's a great read for anyone concerned with reliability and uptime, and aren't we all concerned about that?

We also have the full gamut of *Linux Journal* goodies this month. I review the ZaReason Valta X79, we have a slew of new product announcements, and we have more tips and tricks than you can shake a stick at. So put on your leather fedora, grab a whip and start on your programming adventure! We'll sit back and worry about all the red tape. Try to avoid venomous spiders though; our "adventure insurance" doesn't cover that. ■

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


Logging | IRC | Plasma Active | Linode | EFI | GSL
LINUX JOURNAL
FINE-TUNE GPU PASSWORD CRACKING
HOW TO: INSTALL LINUX ON AN EFI COMPUTER
MOBILE
CODE IN THE CLOUD with an iPad + Linode
GET MORE FROM YOUR TABLET with KDE's New Desktop
Tips to Make IRC More Mobile-Friendly
Logfile Skills for Web Developers
Explore the New Face of Thin Clients

Not Renewing

My subscription was a gift, and is in the name of my then-future wife, sometime in 1996. With the demise of the paper edition, you have lost a customer. Electronic "search" capability does not compensate for "flicking through and finding something interesting". I can use Google to seek something in which I know I am interested. Goodbye.
—Ian Stirling

Ian, sorry to see you go. We wish you the best!—Ed.

Digital Subscription

Linux Journal was the *only* Linux magazine I could find that didn't insult my intelligence by putting a CD (with software I could download for free) in the magazine and tripling the price. And although this may have contributed to the exit of the print version, I thought it showed a great respect for your readers.

So, although I'm not crazy about the digital version, I'll adapt/evolve and enjoy it for many years to come because of your mentality to offer a high-quality product at a fair price.
—R. Scott Sanders

The adjustment is frustrating at times, I agree. I'm excited, however, when I dream about the future possibilities with digital. My ideas for flip-book animations were about as crazy as we could get with paper, but the digital format is new. Who knows what the future might hold!—Ed.

LJ Digital Format

I have no plans to buy a Kindle or any other tablet-type gizmo. That is money better spent on oscilloscopes and stuff like that. I was struggling with the PDF format on my Kubuntu Lucid laptop, when I discovered the EPUB format and Calibre. Now we are cooking. I prefer paper, but I want *LJ* to stay around a good long time, so I bought the CD to help support you. I do miss reading it in the bathroom though.
—Mike

For quick bathroom reading, smartphones aren't too bad either. I actually like to read the EPUB version using EPUBReader on Firefox. We're glad to have you as a subscriber, even if you're not interested in tablets!—Ed.

Now I'm a Believer

I couldn't figure out why I was unable to find *Linux Journal* on the magazine racks at my local bookstore. I even made a formal complaint with the manager. So when I found out you had made a permanent transition to digital format only, I was embarrassed. Intent on keeping this magazine and its articles flowing, I went out and bought a NOOK (I had been toying with the idea anyway due to many interesting hacks I had read about).

My one concern is regarding longevity. Eventually this device, the cloud service running it, or the corporation backing it, will fall. What then will I do when I want to read something printed by your magazine a few years before? Do you at *Linux Journal* not worry about the direction that the cloud and digital-distribution mentality is taking us? Am I the only IT professional in the world that sees the problem with this trend?
—Tex Standridge

Nope, you're not alone. I keep digital backups of every .pdf and .epub edition on my computer and backed up on my server. I'm actually really happy that as a company we've decided to offer the cloud-based interactive version, but also provide a DRM-free downloadable version for our customers. It helps me sleep better at night!—Ed.

Hacker Diet

I love the picture on the table contents page of the March 2012 issue: iPad, remote keyboard and half-eaten snack. This must truly symbolize the gentrification (I'd say "yuppinization", but I think yuppie is passé already) of hacking. Twinkies and Mountain Dew replaced by chocolate croissant and latte?
—John McClatchie

I'm a tea drinker myself, but I won't pass up a latte if offered. I can't say I've ever been a fan of Mountain Dew or Twinkies though. I'm not sure what that makes me!—Ed.

wc Is One Too Many

I was just reading my favorite column in *Linux Journal*, Work the Shell by Dave Taylor, and he pointed out that `wc` doesn't give an accurate count of characters [see Dave's column in the March 2012 issue]. For example:

```
$ echo linux | wc -c
6
```

This should be 5. First off, I want to thank Dave. I've used wc many times, and I never noticed this before (I guess I mostly use it to count lines and words). After realizing this, I used Google to find out why this is happening. It turns out, wc is counting the newline character. This is because `echo`, by default, ends the line with a new line "\n".

A simple fix for this, in the case of using `echo`, would be to use the `-n` argument, which tells `echo` not to put a new line at the end of the line. For example:

```
$ echo -n linux|wc -c
5
```

Thanks again to Dave Taylor for his great shell column. I love that he doesn't just show you how it's done, but that he takes you through his thought processes so you see why he's doing what he is doing.

—Kristofer Occhipinti

Dave Taylor replies: *Thanks for the kudos, Kristofer. I've never really understood why* `wc` *doesn't have a simple flag to skip counting nonprinting characters, which would save vast amounts of fuss. It's even more odd if you think about how C starts array indexing at zero, not one, so there's even more oddness in how things are counted.*

Anyway, you're of course right that `-n` *addresses the problem when you're using* `wc` *with an* `echo` *statement, but if you have a more complicated pipe like this:*

```
find . -name "*.doc" -print | rev | sed 's/[aeiou]//g' | wc -c
```

then it's hard to figure out where to put the `-n` *flag. Extra credit: what does the above actually do?*

iPad and Linode

After reading the article on replacing your laptop with an iPad [see "Swap Your Laptop for an iPad + Linode" by Mark O'Connor in the March 2012 issue], I took advantage of the new iPad release to pick up an iPad 2, and I was surprised to find myself enjoying it quite a bit.

Of course, I had trouble with iSSH because of issues with the Alt and Ctrl key, so I'd recommend using the app called "Prompt" instead. My only problem with it is that I have to use Esc as my meta-key. Otherwise, great article!

—Chris King

Why I Will Not Be Renewing My Subscription after Many Years

In September 2011, you decided to stop publishing a portable, permanent, easy-to-use edition of *Linux Journal*. For six months, I have received monthly e-mails announcing new issues, and when I've had time, I've downloaded the electronic edition, but—and this is *crucial*—I have not read it. I don't have time to read when I have a computer in front of me. I don't have a touchpad, and not being able to put three or more bookmarks in the magazine, so I can easily move around in it (and see non-adjacent pages almost simultaneously) all make it much less convenient. I was allowed to take paper into areas where my classified computers are maintained too, but I can't take electronic devices in with me (not if I hope to take them out again). Print resolution and contrast can be better on paper too, so the physical act of reading has become less pleasurable. I haven't read *Linux Journal* in six months, and I see no likelihood of doing so any time soon. If I'm not reading it, I don't get any benefit from it.

By eliminating the print edition, you have reduced the usefulness of *Linux Journal* to the point where I might as well not subscribe. This is not a threat; I don't expect you to reconsider. I simply wanted you to know why you have lost a reader.

I hope you retain enough readers to continue publishing. It was fun while it lasted.
—Mark Fishman

Sorry the new format doesn't work for you. I know a few folks are printing issues out so they can leaf through them, but that seems to be a bit of a kludge at best. Perhaps someday digital readers will evolve enough to create the tactile feel most people miss.—Ed.

LJ Digital and Tablets

Since you went digital, I have enjoyed reading *Linux Journal* in PDF format on my desktop or notebook computer, but this past weekend, I purchased a Toshiba Thrive tablet, and one of the first things I did was `scp` the latest issue to the tablet. I have to say, the digital version and tablets are a marriage made in digital heaven, and I can once again take my *LJ* with me wherever I go. I vote "yes" to the digital version.
—Howard Pepper

I agree. Although I enjoy Linux Journal *on an E Ink device, there's something magical about seeing it on a full-color tablet. It's almost a bit surreal for some reason. I'm glad you're enjoying the tablet experience!—Ed.*

Subscription Site and TrueCrypt

I just got my February 2012 issue, and the digital edition is what most of the Letters are about. I'm just getting used to it. The only complaint I have is that I need my mailing label to access the subscription site (**https://www.pubservice.com/SubInfo.aspx?PC=LJ&do=D**), and I have to fetch a printed issue to get it.

My second comment is related to the Non-Linux FOSS column in regards to TrueCrypt [see the February 2012 Upfront section]. The distro I work with is Fedora, and there is no TrueCrypt package. The reason literally is "The TrueCrypt software is under a poor license, which is not only non-free, but has the potential to be actively dangerous to end users or distributors who agree to it, opening them to possible legal action even if they abide by all of the licensing terms, depending on the intent of the upstream copyright holder.

Fedora continues to make efforts to try to work with the TrueCrypt upstream to fix all of the issues in their license so that it can be considered Free, but have not yet been successful." (Source: **http://fedoraproject.org/wiki/Forbidden_items#TrueCrypt**.)

On the same page, Fedora suggests "tc-play (**https://github.com/bwalex/tc-play**) is an independently developed TrueCrypt-compatible program under the BSD license. A tc-play package has been submitted for package review for possible future inclusion in Fedora."

Otherwise, I really enjoy reading *LJ* from beginning to end. Keep up the great work!
—JSchiavon

We have fixed the subscriber login page, so you now are able to log in with your e-mail address and zip code. In addition to the login page, the monthly issue download notification e-mail that you receive contains links to your issue archive, so you can download any issues you may have missed, as well as access your subscription account. There is also a Manage your subscription link that allows you to modify your account, renew or access the digital archive.

Also, thanks for the info on TrueCrypt and tc-play. I always feel a bit like I'm in uncharted waters when I highlight a Non-Linux FOSS program. Your feedback is appreciated by me and our readers as well.—Ed.

LJ Digital and Monitors

I was, like many of February's letter-writers, somewhat dismayed by the end of the print edition, but as the compositor for a couple printed journals (no color permitted, adequate paper for halftones, though not great, and certainly not glazed), which have had to be priced at a great deal more per issue than *LJ* ever was, I do understand the pressures *LJ* was facing. Some of the problems mentioned don't seem to me really to exist, however. Jim Fuqua, for instance, condemns the full-spread layout of the PDF version, but with any recent version of Adobe Reader, the button just to the right of the zoom percentage box brings up the pages sequentially, one by one, and you can run through the entire journal with the mouse if you want to. (That makes me dizzy. The Enter key works fine for me.) A properly shaped Reader panel will ensure that the paging really works in "fit page" mode.

I am using FC15 at present, and that leads to another consideration. FC15 does a neat job of setting up an auxiliary screen, which can be turned from landscape to portrait mode easily. It is a bit pricier than an e-reader and can't be carried around, but I, like Jim, am a geezer. I ride free on Philadelphia buses, and by this time, I don't even have to show ID. I can't survive the eyestrain caused by any of the e-readers I have yet seen (I fall asleep), but my 24" diagonal auxiliary screen has put up this "Contact *Linux Journal*" page at something more than 18" diagonal. I can read it with mid-range glasses, sitting back comfortably and not squinnying—at about a 30" distance from the text.

Of course, this is not in the least bit portable, but neither, in any real sense, is my Dell Latitude laptop. If there is an *LJ* article that I really find useful, I print it off in black and white. One could run through color ink cartridges pretty fast with the saturated colors that the PDF comes in, so I don't even think of printing the entire journal.

I did like the printed version, but as I prune down my set to the last print issue, I will have some much-needed shelf space for other things. The digital edition has the virtue that it doesn't take up much room. I am close enough to the publishing industry and its woes to know that *LJ* has made the only possible decision.
—Pierre

I love a secondary monitor in portrait mode. One of these days, I'm actually going to buy a monitor designed for

---

Linux Journal Exclusive Offer:

SAVE 60%* on the new
Intel E5-2600 dual processor server.

**NEW Intel Xeon E5-2620
Dual Processor Server:**

- 16GB DDR3-1333 ECC RAM
- 1u Rackmount Chassis (4 bay hotswap)
- 1TB Enterprise 7.2K SATA HDD
- 10TB Bandwidth
- CIDR/29 IP Reassignment
- $199.95 Monthly (compared to $499 at our largest competitor)

* This 60% savings is strictly limited to the first 15 reservations.

Go to www.ace-host.net/linuxjournal to claim yours now.

it (Dell makes a really nice one, possibly the one you use). For anyone interested in trying it with a "regular" monitor, it's important to realize some monitors have poor viewing angles when you tip them on their side. For looking at a document in portrait mode, however, it's hard to beat!—Ed.

eJournal Is Still Pleasurable to Read

At first, I wasn't sure about the paperless *LJ*; I really prefer to read stuff on paper. PDFs tend to have small print, even on my old 19" 1280x1024 LCD. But you seem to have boosted the point size a little, and reading on-screen really is rather pleasurable.

As to paying for a subscription to the magazine, I paid for access to the information, not the paper. In fact, I started with *IEEE Micro*. When that went away, I subscribed to *Embedded Linux Journal*. When that folded, my subscription was transferred to *LJ*, which has now become, in a way, *eLJ*. And, I'm still reading interesting articles about different aspects of Linux, hardware and computing in general.

The more things change, the harder it is to peel the orange. Change is inevitable; what matters is how we manage that change and cope with it.
—fest3er8

Thank you for the kind words! I have noticed the layout, while still magazine-shaped on the PDF, is a bit different and looks nice on a computer screen. I don't know the exact details about what changed, but I think it looks good both on a computer and on a color e-reader. I'm glad you're enjoying it.—Ed.

WRITE *LJ* A LETTER We love hearing from our readers. Please send us your comments and feedback via **http://www.linuxjournal.com/contact**.

LINUX JOURNAL

At Your Service

SUBSCRIPTIONS: *Linux Journal* is available in a variety of digital formats, including PDF, .epub, .mobi and an on-line digital edition, as well as apps for iOS and Android devices. Renewing your subscription, changing your e-mail address for issue delivery, paying your invoice, viewing your account details or other subscription inquiries can be done instantly on-line: **http://www.linuxjournal.com/subs**. E-mail us at subs@linuxjournal.com or reach us via postal mail at *Linux Journal*, PO Box 16476, North Hollywood, CA 91615-9911 USA. Please remember to include your complete name and address when contacting us.

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

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

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

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

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

Linaro

CONNECT

Join the industry's largest event focused on the development of Linux on ARM at Linaro Connect Q2.12 Hong Kong, May 28 - June 1

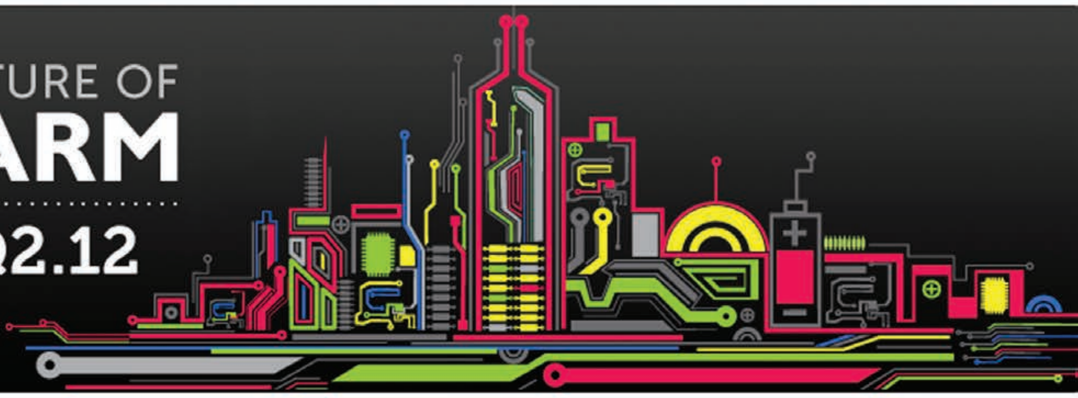
Register for free here:
connect.linaro.org/register

Around 200 of the world's most influential Linux on ARM software engineers will be planning and coding the future of Linux on ARM in an intensive week of discussion, planning, engineering and socializing. Join them to work on developing the toolchain, Linux kernel, key distributions such as Android and Ubuntu, and next generation technologies.

Registration is also open to remote participants, who can participate in individual sessions through IRC and video feeds.

BUILDING THE FUTURE OF
Linux on ARM

Linaro Connect Q2.12
May 28 - June 1, 2012
Gold Coast Hotel, Hong Kong



If you are passionate about, or just want to get involved in, the latest Linux on ARM developments, visit connect.linaro.org

diff -u
WHAT'S NEW IN KERNEL DEVELOPMENT

The **GNU General Public License** is pretty powerful. By ensuring that derivative works can be distributed only if the source code is made available as well under the same license, it ensures that a piece of free software can't suddenly "go dark", if a given person or company wants to fork the code and release a proprietary version.

Since the very early days of Linux, however, there has been a nagging question surrounding this issue: are **kernel drivers** actually derivative works? If the courts ever decide they are, the vast and growing array of binary-only kernel drivers suddenly will find themselves faced with a tough question: either release their source code under the GPL or stop distributing the driver to anyone.

Linus Torvalds has said that he's fine with binary drivers, but other folks have said that it's really the license itself, and not his opinion, that determines whether someone has violated the GPL. Thus far, the courts have not ruled either way, so the question remains open.

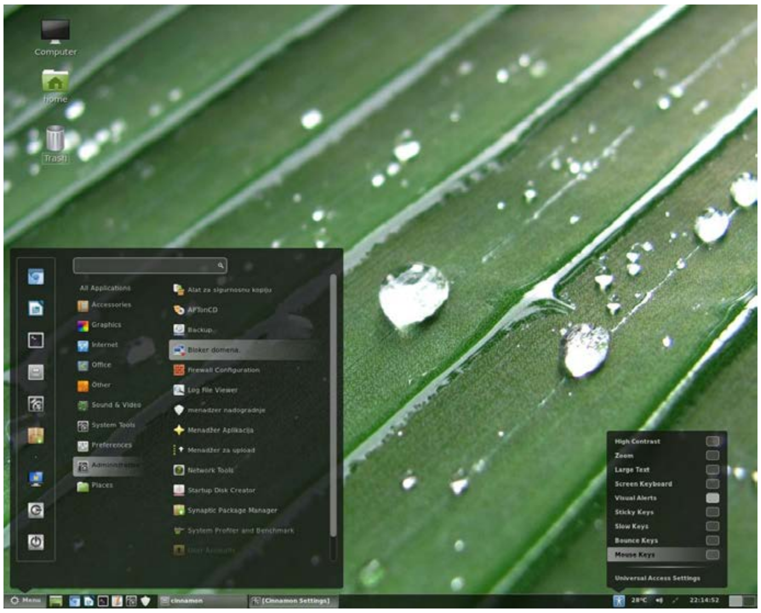
From a practical standpoint, however, it's possible for binary drivers to insinuate themselves so deeply into the kernel that they really do constitute a derived work, anyway you slice it. So, a long time ago, the kernel started using **EXPORT_SYMBOL_GPL** to mark kernel interfaces that truly were considered intrinsic, and that could be used only to code that identified itself as being released under a GPL-compatible license. So, for a long time now, binary-only drivers have had access only to a controlled subset of kernel interfaces that prevent them from essentially forking the kernel into a proprietary version.

That's the history, and it's a balance that seems to work well enough for everyone. But recently, **Christoph Hellwig** submitted a patch, which was accepted, that dramatically changed that balance in the **Virtual Filesystem** (VFS). As **Anton Altaparmakov** said in response to the patch, Christoph's code would make it virtually impossible for binary-only drivers to read or write files the way they always had before. The drivers would have to re-implement from scratch all the VFS interfaces for reading and writing.

Ultimately, it turned out that Christoph had not intended to add these restrictions, but the discussion revealed a fairly big gap between the opinions and preferences of various kernel developers. **Alan Cox**, for example, indicated (as nearly as I can make out) that he opposed all binary-only drivers and did not think they were legal, regardless of the existence of EXPORT_SYMBOLS_GPL. He said he'd never given permission for them to use any of his code. And, since his code is virtually everywhere in the kernel, that's saying a lot.

Alexander Viro, who's been the VFS maintainer for many years, said that all interfaces exported from the VFS should be made available to binary-only drivers, unless there was "a damn good reason" to restrict them. He also said that such restrictions would need to be clearly documented.—**ZACK BROWN**

Spice Up Your Desktop with Cinnamon!

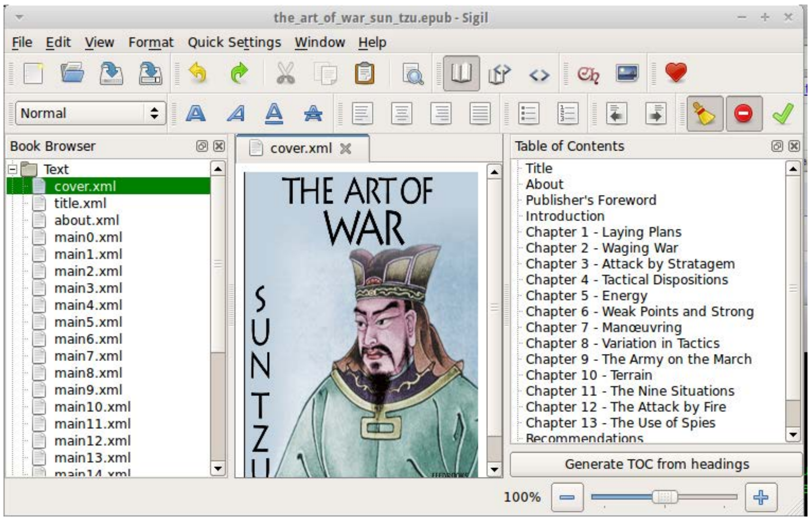


(Screenshot from **linuxmint.com**.)

If you are disgruntled by the new interfaces provided by recent distribution releases, namely GNOME 3 and Unity, you might want to take a look at Cinnamon. With its traditional feel and extreme theme-ability, Cinnamon is a desktop interface bound to spice up anyone's computer. The general feel is that of GNOME 2, or perhaps XFCE, but its polished look and downloadable themes make it truly exciting to behold.

Development has revolved around Linux Mint, but Cinnamon also can be installed on Ubuntu for those so inclined. If you've never seen it in action, be sure to give it a whirl: **cinnamon.linuxmint.com**.
—**SHAWN POWERS**

Open Formats, Open Editors



E-books are currently quite a hot topic in the publishing world. Heck, for the past few months, it's been quite a hot topic here as well! Thankfully, digital publication doesn't have to mean proprietary formats and DRM-laden files.

If you want to delve into the world of e-book creation, you should check out Sigil. It's a cross-platform WYSIWYG tool for creating EPUB-format e-books. It supports images, table of contents, and all the other features that make e-books so powerful. If you have a book you want to convert into EPUB format, or if you just have a text file you'd like to add markup language to for e-readers, Sigil is a powerful tool you won't want to miss.

Download the latest version from **code.google.com/p/sigil**.—SHAWN POWERS

They Said It

Buying the right computer and getting it to work properly is no more complicated than building a nuclear reactor from wristwatch parts in a darkened room using only your teeth.

—Dave Barry

There is only one satisfying way to boot a computer.

—J.H. Goldfuss

In a way, staring into a computer screen is like staring into an eclipse. It's brilliant and you don't realize the damage until its too late.

—Bruce Sterling

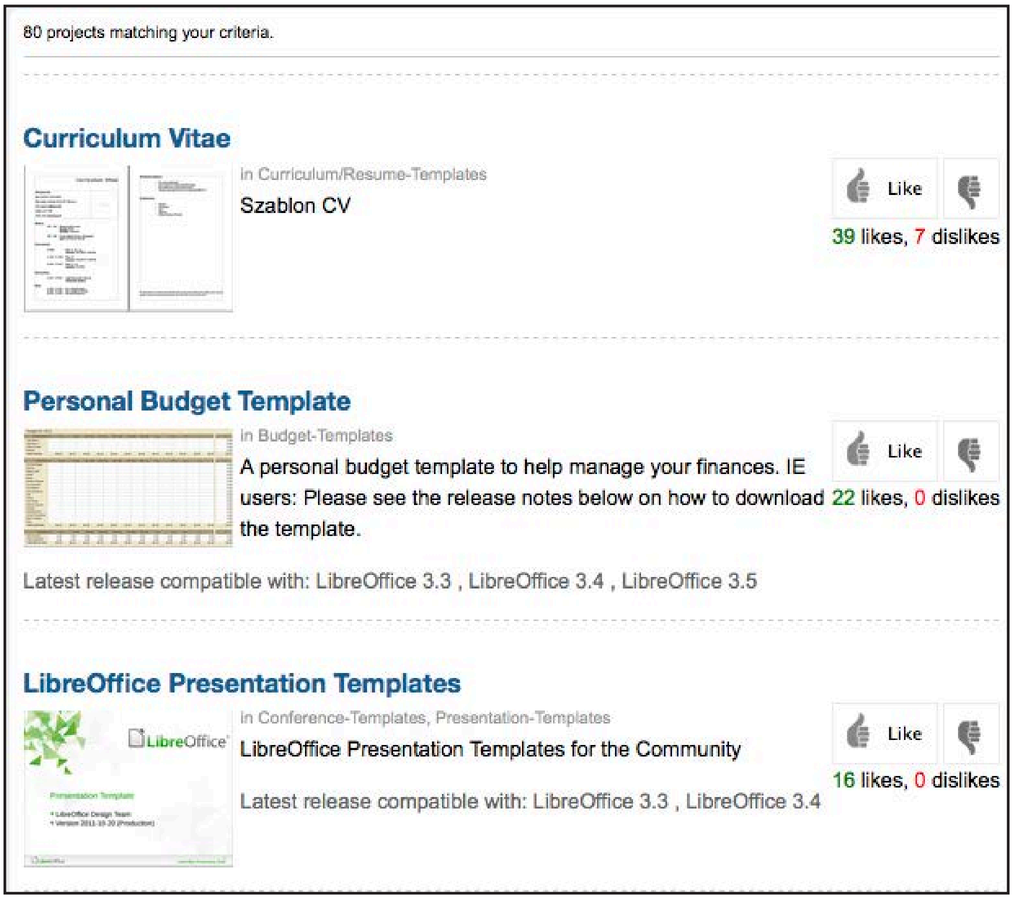
If the automobile had followed the same development as the computer, a Rolls-Royce would today cost $100, get a million miles per gallon, and explode once a year killing everyone inside.

—Robert Cringely

A computer once beat me at chess, but it was no match for me at kick boxing.

—Emo Philips

That’s a Beautiful $DOCUMENT_TYPE You’ve Got There


80 projects matching your criteria.

Curriculum Vitae
in Curriculum/Resume-Templates
Szablon CV
39 likes, 7 dislikes

Personal Budget Template
in Budget-Templates
A personal budget template to help manage your finances. IE users: Please see the release notes below on how to download the template.
22 likes, 0 dislikes
Latest release compatible with: LibreOffice 3.3 , LibreOffice 3.4 , LibreOffice 3.5

LibreOffice Presentation Templates
in Conference-Templates, Presentation-Templates
LibreOffice Presentation Templates for the Community
16 likes, 0 dislikes
Latest release compatible with: LibreOffice 3.3 , LibreOffice 3.4

One of the biggest frustrations most new LibreOffice (or OpenOffice.org) users have is the lack of templates and clip art. We’ve addressed this problem before, but with the recent surge of LibreOffice, it’s important to know how to improve your powerful office suite!

LibreOffice offers both templates and extensions to add creativity and flair to your documents. Whether you’re looking for a template to aid with your home budgeting spreadsheet or want some clip art to advertise your garage sale, the LibreOffice Web site holds plenty of options.

A quick click to **templates.libreoffice.org** or **extensions.libreoffice.org** will give you a huge list of available add-ons. You also will have the opportunity to add your own templates or extensions to the Web site so others can benefit from your creations! The site features a voting system to help promote the highest quality submissions, and a search feature narrows down the available items to fit your needs. Check it out $DATE_THIS_ARTICLE_IS_READ! **—SHAWN POWERS**

Programming for Scientists

My last several articles have covered different software packages that are useful to scientists trying to do computational science. I tried to explore as broad a spectrum of subjects as I could in those pieces, and I even covered some basic programming constructs like MPI or scipy. But, I always have been limited by the amount of space a venue like this allows. All I can do is provide a taste of what is out there and hope that readers take it away and learn more on their own. Also, in many cases people find themselves doing research in areas that never have been done by anyone else before. This means there will not be an appropriate software package, and researchers will need to write their own software from scratch.

One major problem for computational science researchers is that they simply do not have the time to attend normal classes over the span of a term or two in order to learn the skills they need to do their work. They need to be able to jump-start their research and essentially go from zero to 100mph in no time flat. Part of my day job is to help them do this. I provide crash courses in most of the subjects that they may need. But, what can they do when they leave and try to apply this information several days or weeks later? Enter the Software Carpentry site (**software-carpentry.org**), a resource that should be on every researcher's bookmark list. I have no association with the author and maintainer of the site. I'm just glad to have a high-quality source of information to which I can point my users.

The first level of resources available is a set of self-paced on-line workshops. These workshops are distributed under a creative commons license, specifically the Creative Commons Attribution License. This means you are free to use the material and remix it, as long as you properly attribute the author. These workshops cover a vast number of subjects and are available as both PDF and PowerPoint files. For some of the workshops, video screencasts even are available, so it's almost like having an instructor right there with you. Each topic is broken down into smaller sections to make digesting them easier. Additionally, exercises are available so you can review the material.

Many new researchers, graduate students and post-docs have had little or no experience in computational science at all. Many never even have seen any type of UNIX environment. This is quite a stumbling block, as most high-performance computing centers that I know of run Linux. So you probably will want to start with the workshop The Shell. This workshop is broken down into the following sections:

- Introduction
- Files and Directories

- Creating and Deleting
- Pipes and Filters
- Permissions
- Finding Things
- Job Control
- Variables
- Secure Shell (SSH)

This list of items should make new Linux users comfortable enough to use the command line effectively. Because most HPC clusters are accessed through an SSH connection, being comfortable on the command line is essential.

The next workshop you should look at is the Version Control workshop. To my mind, this is one of the most important subjects to learn for computational science research. This is a field where code constantly is being toyed with, by many different people over long periods of time. It is of utmost importance to be able to back out experimental changes in the code when something breaks or when you've gone down the wrong path in your research. But, almost no one uses a version control system. So, just to make my life easier as a research consultant, please go ahead and check out this particular workshop. It will be one of the most useful workshops you could attend.

As far as programming workshops go, the only language explicitly covered is Python. The Python workshop is relatively complete, and it covers the following:

- Basics: running Python, variables, comparison operators.
- Control Flow: while loops and conditionals.
- Lists: creating, deleting and maintaining lists, for loops.
- Input and Output: dealing with files.
- Strings: handling strings, formatting, concatenation.
- Aliasing: what it is and how it can cause problems.
- Functions: what they are and how to use them effectively.
- First-Class Functions: binding functions to variables, passing functions to functions.
- Libraries: importing modules, dealing with namespaces.
- Tuples: creating and indexing, unpacking lists.
- Slicing: slicing vs. indexing, and so on.
- Text: how lines and characters are stored, dealing with unicode.

Python is relatively similar to other languages (like C), so you should be able to apply what you learn here to those other languages with just minor syntax translations. Also, Python is growing in popularity in scientific programming circles due to its clean formatting rules and the relative ease of incorporating external high-performance libraries written in C or FORTRAN. In this sense, you almost can consider Python to be a glue language, but it has quite a lot of capability available directly through external libraries like numpy and scipy. You could do worse as a computational scientist than learn Python. With its growing popularity, there is also a greater chance that the specific problem area you're researching already has tools or libraries available.

Once you have at least one language under your belt, it is time to learn more of the details involved in programming itself. These workshops cover the following:

- Program Design: goes through a simple example of designing, debugging and improving a program.
- Testing: how you should test your software, handle exceptions and do unit tests.
- Make: how to use rules, patterns and macros to build your software.

These topics cover a lot of the extra items you need to know in order to program effectively, but they aren't strictly programming proper. That topic is covered by the following workshops:

- Sets and Dictionaries: using associative data structures to represent data that doesn't really fit into a list.
- Regular Expressions: how to use regular expressions for pattern matching.
- Databases: an introduction to SQL.
- Data Management: an introduction to managing your data.
- Matrix Programming: using numpy to handle numerical processing.
- Multimedia Programming: programming using sound, pictures and other media files.
- Spreadsheets: using spreadsheets for analysis and visualization.

With these workshops, you will learn many of the programming elements and structures that will be of use to you in scientific programming. After this, you should have covered enough, hopefully, to be able to program a solution to the problem you are studying. Again, all of these workshops include exercises, so you actually can try applying what you have learned. I'm a firm believer that you don't learn anything until you actually try to use it.

In-person workshops and boot camps also are available. Because all of the material is available for free reuse, you simply can use the workshop materials to put on your own workshop or boot camp. The team behind Software Carpentry also is available to do in-person boot camps. You can contact them through the Web site to make arrangements. These boot camps are two- or three-day crash courses to cover the bulk of the material, and they're always being offered at different places around the globe—follow the blog to see when one is being offered in your neck of the woods. If you do decide to run your own, the team at Software Carpentry is happy to help out and spread the word through its network. A forum is available at the Web site for each of the workshop topics where you can discuss the material with other attendees or other presenters.

Finally, I suggest that you actually subscribe to the blog RSS feed. New workshops always are being added, and new boot camps always are being planned. Watching the RSS feed will keep you informed about these additions. As always, feel free to contact me if you have anything specific you'd like to see covered here. Hopefully, I've been able to plant the seed and give you ideas on how you can pick up the skills you need.
—**JOEY BERNARD**

Non-Linux FOSS

(Image from **www.xonotic.org**.)

Some say Windows has all the best games, and in the case of Xonotic, it's partially true. Xonotic is a free, open-source, first-person shooter designed by the developers of Nexuiz. Xonotic uses a highly modified version of the Quake engine called DarkPlaces. Sporting both single and multiplayer modes, the fast-paced action and detailed maps will please even the fussiest gamers. Download the Windows version of Xonotic from the Web site: **www.xonotic.org**.

Oh, and for those folks *not* running Windows? Yeah, it's available for you too. Xonotic runs on Windows, OS X and Linux. Get your frag on!—**SHAWN POWERS**

REUVEN M. LERNER

Design Frameworks

Want your site to look good, even though you're not a designer? Try a design framework.

For as long as I can remember, I've known how to use a pencil. I can write with it, and I even can draw with it—although in my case, saying I can draw is something of a sad exaggeration. I might know how to use a pencil and thoroughly understand its technology, but that technical knowledge doesn't mean I can draw something aesthetically pleasing.

I mention this because I always think of my poor drawing abilities when I work with CSS. I understand the technology and have used it for many years. I'm comfortable working with and modifying stylesheets, and using complex selectors. And yet, any Web application that I create on my own looks ugly. Despite all of my technical knowledge of CSS, I'm generally unable to make a nicer, more pleasing design.

Fortunately for people like me, a new type of framework exists to help with such problems. Just as server-side frameworks like Ruby on Rails and Django make it easy to create Web applications, and client-side frameworks like Backbone.js and Knockout.js make it easy to create in-browser applications, design frameworks make it easy to decorate and design your applications.

Now, when I first heard about design frameworks (sometimes known as CSS frameworks), I wondered how they could possibly help. After all, I already know CSS; what could they contribute? The answer is: a great deal. By embracing a design framework, you gain a number of CSS classes and IDs that make it very easy to lay out your page. By choosing the right classes, you can make a site look more than reasonable, even if you have my graphic design skills. Using a design framework means that even if you don't have any design skills, you can make a site that looks fairly pleasing. If you do have design skills, the framework will allow you to do more with less effort.

In this article, I describe some of the leading design frameworks, leading up to the latest and most interesting one, Twitter's Bootstrap. In my next article, I'll look more closely at Bootstrap and why it has taken the world by storm.

Blueprint

The first design framework I'm aware of was (and is) Blueprint. The idea behind Blueprint is pretty simple. You download and refer to Blueprint's three CSS files: one for the screen, one for print and one with special definitions for Internet Explorer:

```
<link rel="stylesheet" href="css/blueprint/screen.css"
 ➥type="text/css" media="screen, projection">
<link rel="stylesheet" href="css/blueprint/print.css"
 ➥type="text/css" media="print">
<!--[if lt IE 8]>
   <link rel="stylesheet" href="css/blueprint/ie.css"
 ➥type="text/css" media="screen, projection">
<![endif]-->`
```

By including these stylesheets, Blueprint resets the CSS values for the user's browser, removing any defaults that might have been set elsewhere for margins, padding and borders.

Blueprint also resets the defaults for typography, ensuring that your headline and paragraph tags look better, and more consistent, than the defaults defined by each browser. Again, even without any additional effort on your part, Blueprint ensures that your typography is a bit more in line with established principles of design. As the Blueprint Web site says, these principles predated the Web and still are relevant in our electronic age.

The real magic of Blueprint, however, is in its layout. Blueprint offers a "grid layout", which in practical terms means it provides you with CSS classes that let you indicate how wide a particular element should be, from one column across to 24 columns across. Do you want to display two divs, side by side, with the left-hand div taking up two-thirds of the space? No problem—just define them as follows:

```
<div class="container">
    <div class="span-16">This is the left-hand div</div>
    <div class="span-8 last">This is the right-hand div</div>
</div>
```

By using these CSS classes, the divs automatically will take up the right amount of space on a user's screen. Moreover, because the content is inside a div named "container", the width of the screen (and of the divs) will adjust automatically according to the width of the user's browser window.

Before Blueprint, Web designers would do all this work themselves, implementing and re-implementing such columnar layouts on their own, specifically for each site. The advantage of Blueprint's 24-column grid is that you have enormous flexibility in the layout of your pages, letting each element extend across however many columns you think are appropriate. So long as the span-* classes add up to 24, and the final element on each row has the "last" class, the layout will look pretty good. This is far better, in every way, from what many people would do to ensure a certain layout, namely the use of tables. Tables are great, but not for laying out a page. I can't tell you how many times I've seen tables triple-nested inside other tables, just to get the layout to work appropriately.

Another advantage of Blueprint is that it has tried and tested these CSS definitions on a variety of browsers. You no longer need to worry about whether things will work on Chrome, Firefox, IE and Safari, because Blueprint has taken care of that for you. The special IE-only CSS file ensures that Internet Explorer also will work correctly, even if your users are running an old version.

Of course, because Blueprint is a bunch of CSS files, you always can change it to suit your needs. And indeed, much of the (extensive) Blueprint documentation describes how you can customize not only the layouts, but also the CSS itself.

The biggest criticism of Blueprint is that its class names violate the spirit of CSS. Sure, you can have classes named "span-5" and "span-10" to indicate that you want text to extend across five or ten columns of your 24-column layout. But, wasn't CSS supposed to free you from such calculations, allowing you to think more in terms of semantic names? My response is that, yes, this is a good point, but Blueprint's layouts are so useful and so easy to understand, it's unhelpful to focus on the original intent of CSS's designers.

Sass, SCSS and Compass

Another problem with Blueprint isn't inherent to Blueprint itself, but rather to the technology on which it's based. CSS is a very clever technical standard, but it has been demonstrably difficult for people to learn and understand, and also for many designers to use. True, some people are able to make beautiful, clever designs with CSS, and I don't mean to diminish their capabilities in any way. But, CSS has a number of deficiencies, many of which I hadn't thought about until I discovered Sass a few years ago.

Sass (Syntactically Awesome Stylesheets) is a CSS preprocessor. That is, it is a file format that you compile, with the output of the compilation being a CSS file. Sass has a large number of advantages over straight CSS, starting with the fact that you can set variables. This is a tremendous thing when you're creating a site that should have a consistent color scheme. You can set the colors in one place and then use them in many different places.

Sass also lets you nest CSS definitions, such that if you have several elements that share definitions, you no longer have to repeat those similarities in different places, but can take advantage of something akin to variable scoping in a programming language. Nesting also means that complex selectors become easier both to write and to read, since they break the path down into digestible parts.

Sass also offers a type of function or macro definition, known as a "mixin", allowing you to keep your CSS DRY ("Don't Repeat Yourself"). This means you can define a CSS rule in one place and then include it in multiple other places.

Now, Sass has been around for several years, and it has worked well during that time. It originally was written by developer Hampton Catlin (who also developed the Haml template format), but has since been taken over by Nathan Weizenbaum and Chris Eppstein. The original Sass syntax resembled Python in some ways, in that trailing semicolons were removed and indentation was a mandatory part of the syntax. That is, the indentation of your declaration, or even of the definition of your mixin, mandated indentation, and the level of indentation reflected the scope within which your definitions would take effect. For example, here is a simple set of definitions in Sass, taken from the sass-lang.com home page:

```
table.hl
  margin: 2em 0
  td.ln
    text-align: right

li
  font:
    family: serif
    weight: bold
    size: 1.2em
```

You then would run a Sass compiler over the Sass file, producing a CSS file that could be interpreted by a Web browser.

This syntax still exists, and if you prefer it, Sass happily will work with you. However, the preferred syntax is now known as SCSS, and it is a superset of the existing CSS syntax. My impression is that the syntax changed partly because the differences between CSS and Sass were too great for many people, and the mandatory indentation caused problems for a number of users. Thus, the Sass shown above can be expressed in SCSS as follows:

```
table.hl {
  margin: 2em 0;
  td.ln {
    text-align: right;
  }
}

li {
  font: {
    family: serif;
    weight: bold;
    size: 1.2em;
  }
}
```

As you can see, the syntax is quite similar to that of standard CSS, making it easier for people to make the transition from CSS.

The entire Sass framework, including both syntaxes, is mainly implemented in Ruby and is in widespread use among Ruby-language Web developers. However, you aren't required to use Ruby as your Web development language. Many people use this system, compiling it from Sass/SCSS into CSS without knowing or caring that they're using Ruby. That said, the Ruby on Rails framework has adopted SCSS as a standard part of its infrastructure, such that anyone using Rails likely will know and use it.

Compass

Neither Sass nor SCSS is a design framework; they are stylesheet languages and are meant to simplify your CSS development. But the authors of Sass have created the Compass design framework (which they call a "CSS authoring framework"), which predefines a large number of classes and mixins for use in your own projects. So if you want to have rounded corners or standardized fonts or a set palate of colors, Compass will make it easy for you to do so.

Moreover, Compass comes with a Blueprint compatibility mode. This means if you have a site that already is working with Blueprint, you can (mostly) just replace the Blueprint CSS files with the Compass CSS files, and you're done.

Compass is both clever and powerful, and I'm very impressed with what I have seen and how it works. Nevertheless, I have been disappointed with its level of documentation, especially on issues having to do with the Blueprint compatibility. Yes, I was able to find all the answers I wanted in the end, and the e-mail list is particularly active and helpful. But I was surprised by how much I had to turn back to the Blueprint documentation to understand what the Compass compatibility mode was doing, rather than just being able to access the documentation directly.

None of this takes away from the power Compass provides and the flexibility it offers users. If you're interested in working with Compass, and not just with Sass/SCSS, either of the books that I mention in the Resources section probably will be useful, given the detail in which they describe the Compass framework.

LESS

LESS is another stylesheet language, and it's very similar in many ways to the SCSS syntax from the Sass framework. Indeed, my impression is that SCSS and LESS learned from and influenced one another, although it's not clear to me just who influenced whom, and at what times. The bottom line is that if you're comfortable with SCSS, you're likely to be comfortable with LESS, and vice versa. There are some differences in the syntax, but they're pretty minor.

One difference between LESS and Sass is that LESS files can be compiled in a number of different ways. They can be turned into CSS using a compiler—originally written in Ruby, but currently written in JavaScript—running on the developer's computer, either before or during the deployment process. However, because the implementation is in JavaScript, another option is available. You can download the LESS stylesheet, as is, into your browser and then use a JavaScript library to translate it into CSS on the fly. For example:

```
<link rel="stylesheet/less" type="text/css" href="styles.less">

<script src="less.js" type="text/javascript"></script>
```

For performance reasons, it's best to compile the LESS file into CSS beforehand. However, this is useful when the compiler is not available, if you want to generate stylesheets dynamically, and also during development.

Why would you prefer LESS over SCSS, or vice versa? I honestly cannot give a compelling reason for one over the other. They're so similar and have such advantages over plain-vanilla CSS that it doesn't matter which you choose, so long as you go with one of them. Before Bootstrap, I would have put my money on SCSS becoming the de facto standard, but Bootstrap might have given LESS a new lease on life, and might even help it overtake SCSS.

Conclusion

Given the similarity between LESS and SCSS, I'm not surprised a framework emerged that is based on LESS. But the framework that was created (Bootstrap, written by two engineers at Twitter and released under the Apache license) has turned out to be a huge hit among developers and designers alike. Bootstrap not only includes Blueprint's grid, but also a large number of other conveniences and stylings that make it easy to have good-looking navigation bars, tables, forms and even widgets for dynamic, client-side applications. Bootstrap even is capable of modifying the size and shape of menus depending on screen size, making it a "reactive" framework appropriate for mobile devices as much as desktop computers.

Some people even are getting tired of Twitter Bootstrap, because it has become so popular and makes it easy to create designs that look like many other Bootstrap sites. But for someone like me, who just wants to get a simple site up and running, and for it to be relatively nice-looking, I've found Bootstrap to be a huge help.

In my next article, I'll actually look at Bootstrap—what it provides, how to install and use it, and even how to use it along with Ruby on Rails, which (as I mentioned above) comes with support for SCSS, rather than LESS. Regardless of which of these systems and frameworks you use though, I strongly encourage you to try them and incorporate them into your own work. An experienced designer presumably will know how to integrate these technologies, while design-challenged programmers will welcome the chance to create a nice-looking site on their own.■

Reuven M. Lerner is a longtime Web developer, consultant and trainer. He is also finishing a PhD in learning sciences at Northwestern University. His latest project, SaveMyWebApp.com, went live this spring. Reuven lives with his wife and children in Modi'in, Israel. You can reach him at reuven@lerner.co.il.

Resources

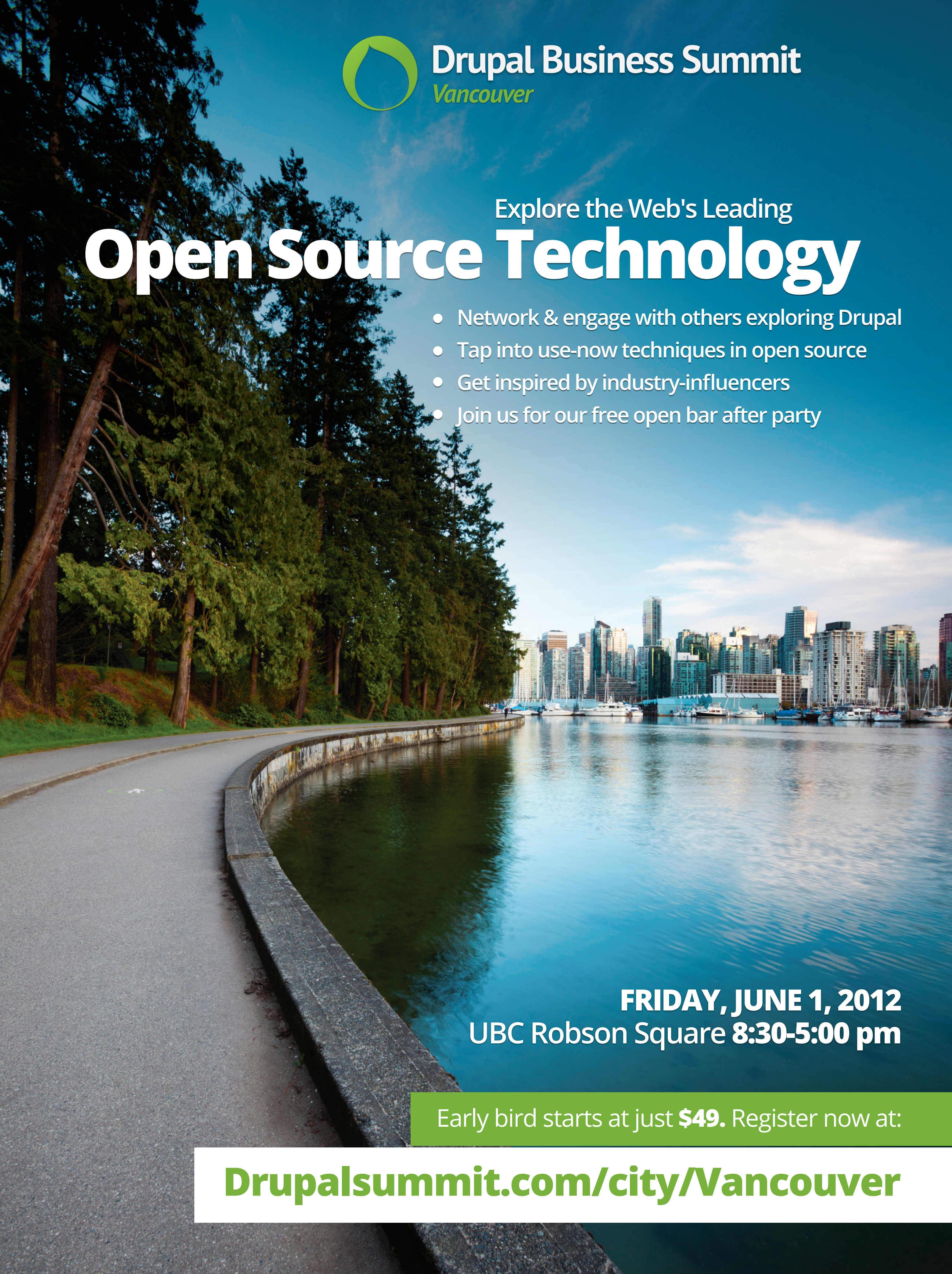
If you are interested in CSS, it might be useful to read up on the subject. The W3C's standard is at **http://www.w3.org/Style/CSS**. You also might be interested in *HTML & CSS: The Good Parts*, a book written by Ben Henick and published by O'Reilly that tries to point readers in the direction of useful and appropriate techniques. Another good O'Reilly book on the subject of CSS is the *CSS Cookbook* (3rd edition) by Christopher Schmitt.

Blueprint has not been updated since May 2011, but it still works and is well documented. You can read about it and download it from **http://blueprintcss.org**.

Sass/SCSS is documented at **http://sass-lang.com**. The site includes many recipes for common things that people want to do. Compass, the framework built on Sass/SCSS, is at **http://compass-style.org**. Note that many on-line tutorials and screencasts about Compass use the old Sass syntax and, thus, might be a bit hard to understand if you know only the new syntax.

Two books about Sass and Compass are *Sass and Compass in Action* by Wynn Netherland, Nathan Weizenbaum and Christopher Eppstein, published by Manning; and *Pragmatic Guide to Sass*, written by Sass creator Hampton Catlin, published by the Pragmatic Programmers.

Finally, if you are interested in comparing the finer points of Sass and LESS, a nice chart and introduction is at **https://gist.github.com/674726**.



Drupal Business Summit
Vancouver

Explore the Web's Leading
Open Source Technology

- Network & engage with others exploring Drupal
- Tap into use-now techniques in open source
- Get inspired by industry-influencers
- Join us for our free open bar after party

FRIDAY, JUNE 1, 2012
UBC Robson Square **8:30-5:00 pm**

Early bird starts at just **$49.** Register now at:

Drupalsummit.com/city/Vancouver

DAVE TAYLOR

Calculating Word Point Values

Dave continues improving the *Scrabble* and *Words With Friends* script.

My last article ended by wrapping up the word finder utility for *Scrabble* and *Words With Friends*, a fun and complicated project that ended up requiring about 65 lines of Bourne Shell. That's not too long, but for a shell script, actually, it is rather long, and it was one of the scripts we've written that most begged to be coded in Perl or another more string-friendly language. Still, we persevered, right?

In this article, let's wrap things up by looking at word values based on the individual letter values in both *Scrabble* and *Words With Friends*.

To start, here's how the point chart looks for *Words With Friends*:

```
A=1   D=2   G=3   J=10   M=4   P=4   S=1   V=5   Y=3
B=4   E=1   H=3   K=5    N=2   Q=10  T=1   W=4   Z=10
C=4   F=4   I=1   L=2    O=1   R=1   U=2   X=8
```

This contrasts with the somewhat lower point value of letters in the game *Scrabble*:

```
A=1   D=2   G=2   J=8   M=3   P=3   S=1   V=4   Y=4
B=3   E=1   H=4   K=5   N=1   Q=10  T=1   W=4   Z=10
C=3   F=4   I=1   L=1   O=1   R=1   U=1   X=8
```

If you look closely, you'll see that just about every word is going to be worth less in *Scrabble* than in *Words With Friends*—interesting. I always assumed that the two used the same basic letter-for-letter point values, so that calculating point values in one game taught you how to calculate points for the other—not so!

Calculating Point Values

There are a lot of ways to translate a sequence of letters into their individual point values and calculate the sum of those values. Indeed, an array comes to mind immediately, but the problem is that you still have to write the code to step through the individual letters, and if you're doing that, why not use sed for the letter → numeric value substitution instead?

It turns out that's a fast and elegant

Now that you have a bunch of numeric values, however, what do you do with them?

solution, and I've coded it also to compare easily what letters have a given point value in wwf (aka *Words With Friends*) and s (aka *Scrabble*).

Here's how I coded the one-point letters in each:

```
wwf1="s/[asetior]/ 1 /g
s1="s/[asentiloru]/ 1 /g";
```

As you can clearly see, *Scrabble* has a lot more one-point letters than *Words With Friends* does. Again, I had no idea until I started this analysis.

With sed expressions coded up this way, it's easy to turn a word like "cat" into "4 1 1". Here's the full set of substitutions:

```
wwf1="s/[asetior]/ 1 /g"
wwf2="s/[dlnu]/ 2 /g"
wwf3="s/[ghy]/ 3 /g"
wwf4="s/[bcfmpw]/ 4 /g"
wwf5="s/[kv]/ 5 /g"
wwf8="s/[x]/ 8 /g"
wwf10="s/[jqz]/ 10 /g"

s1="s/[asentiloru]/ 1 /g"
s2="s/[d]/ 2 /g"
s3="s/[mpbc]/ 3 /g"
s4="s/[vyhwf]/ 4 /g"
s5="s/[k]/ 5 /g"
s8="s/[jx]/ 8 /g"
s10="s/[qz]/ 10 /g"
```

Quite honestly, entering all that data is the hardest part of creating the word-point-value script. It's tedious work, so you'll be smart to copy and paste.

Now that you have a bunch of numeric values, however, what do you do with them? It turns out that's easy too:

```
sed 's/  / + /g'
```

Think about the original substitution, and you'll see what I'm doing: the first letter substitutes to <space> letter <space>, as does every subsequent letter. Therefore, individual spaces denote the very beginning and end of the word, while double spaces are only between digits. Replace those double spaces with a +, and that sequence of digits translates into a simple mathematical formula:

```
4 + 1 + 1
```

To solve simple math, you can use `$(( ))` as a shell notation, or you can call the built-in function `expr`—same basic result.

Before we get there, however, here's how I'll use that sequence of sed substitutions in the script:

```
wwfexpr=$(echo $1 | sed "$wwf1;$wwf2;$wwf3;$wwf4;$wwf5;$wwf8;$wwf10"
 ➥| sed 's// + /g')
sexpr=$(echo $1 | sed "$s1;$s2;$s3;$s4;$s5;$s8;$s10" |
 ➥sed 's/  / + /g')
```

You can see I've tucked the double-space-to-plus-sign substitution into the same subshell invocation too—short and neat. In fact, those two lines are the heart of the script. There's only one line left actually, and it both calculates the actual values and shows the result:

```
echo "\"$1\" has a base point value of $(expr $wwfexpr) in WwF
 ➥and $(expr $sexpr) in Scrabble"
```

This is what I really like about programming with the power of the entire Linux shell at your fingertips: this script that calculates point values for a given word in both *Scrabble* and *Words With Friends* is actually only three lines long, if you don't count the variables we set up at the beginning.

Now, let's run it to see what kind of values we see:

```
$ sh wordvalue.sh calculate
"calculate" has a base point value of 18 in WwF and 13 in Scrabble
```

Let me explain the `sh script.sh` notation briefly, in case you haven't see this approach before. A classic way that hackers Trojan Horse a Linux or UNIX system is to drop a shell script like vi or ls into somewhere like the /tmp directory. It's not a problem, unless your PATH looks like this:

```
.:/bin:/usr/bin:/usr/local/bin
```

in which case you can unwittingly run the invasive script and possibly create a setuid root copy of the shell for the bad guys to exploit at their later convenience. Security's a bit far afield for this particular column, but suffice it to say that for security reasons, I never have "." in my PATH.

Therefore, I could use `./script` to invoke the script in my current directory if it's marked as executable, but since I have so many scripts lying around, I find it even safer to not mark them as executable until I'm 100% sure they're done and tested. Instead, `sh script` works just as well, although it spawns a subshell for execution.

Now you know. And, here are more examples:

```
$ sh wordvalue.sh word
"word" has a base point value of 8 in WwF and 8 in Scrabble
$ sh wordvalue.sh linux
```

```
"linux" has a base point value of 15 in WwF and 12 in Scrabble

$ sh wordvalue.sh journal
"journal" has a base point value of 19 in WwF and 14 in Scrabble
```

At this point, I'll leave it as an exercise for you, the reader, to figure out how to graft this functionality onto the script we wrote in the previous few articles that calculated possible words from a set of letters.

The additional bonus task is to be able to analyze the board so you can figure out how to cover DL, TL, DW and TL squares, as available (that stands for double letter, triple letter, double word and triple word, in case you're not a hard-core word-gamer). Beware though, it's considerably more difficult, because now you have to figure out how to enter the current state of the board—definitely extra credit! ■

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 www.DaveTaylorOnline.com.

Powerful: Rhino

Rhino M6500/E6510
- Dell Precision M6500 w/ Core i7 Quad (8 core)
- Dell Latitude E6510 w/ 2.53-2.8 GHz Core i5/i7
- Up to 17" WUXGA LCD w/ X@1920x1200
- NVidia Quadro FX 3800M
- 250-750 GB hard drive
- Up to 32 GB RAM (1333 MHz)
- DVD±RW or Blu-ray
- 802.11a/b/g/n
- Starts at $1385

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

Tablet: Raven

Raven X201 Tablet
- ThinkPad X201 tablet by Lenovo
- 12.1" WXGA w/ X@1280x800
- 2.0-2.13 GHz Core i7
- Up to 8 GB RAM
- 250-500 GB hard drive / 160 GB SSD
- Pen/stylus input to screen
- Dynamic screen rotation
- Starts at $1940

Rugged: Tarantula

Tarantula CF-31
- Panasonic Toughbook CF-31
- Fully rugged MIL-SPEC-810G tested: drops, dust, moisture & more
- 13.1" XGA TouchScreen
- 2.4-2.53 GHz Core i5
- Up to 8 GB RAM
- 160-750 GB hard drive / 256 GB SSD
- Call for quote

EmperorLinux
...where Linux & laptops converge

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

Model specifications and availability may vary.

KYLE RANKIN

The Sysadmin's Toolbox: iftop

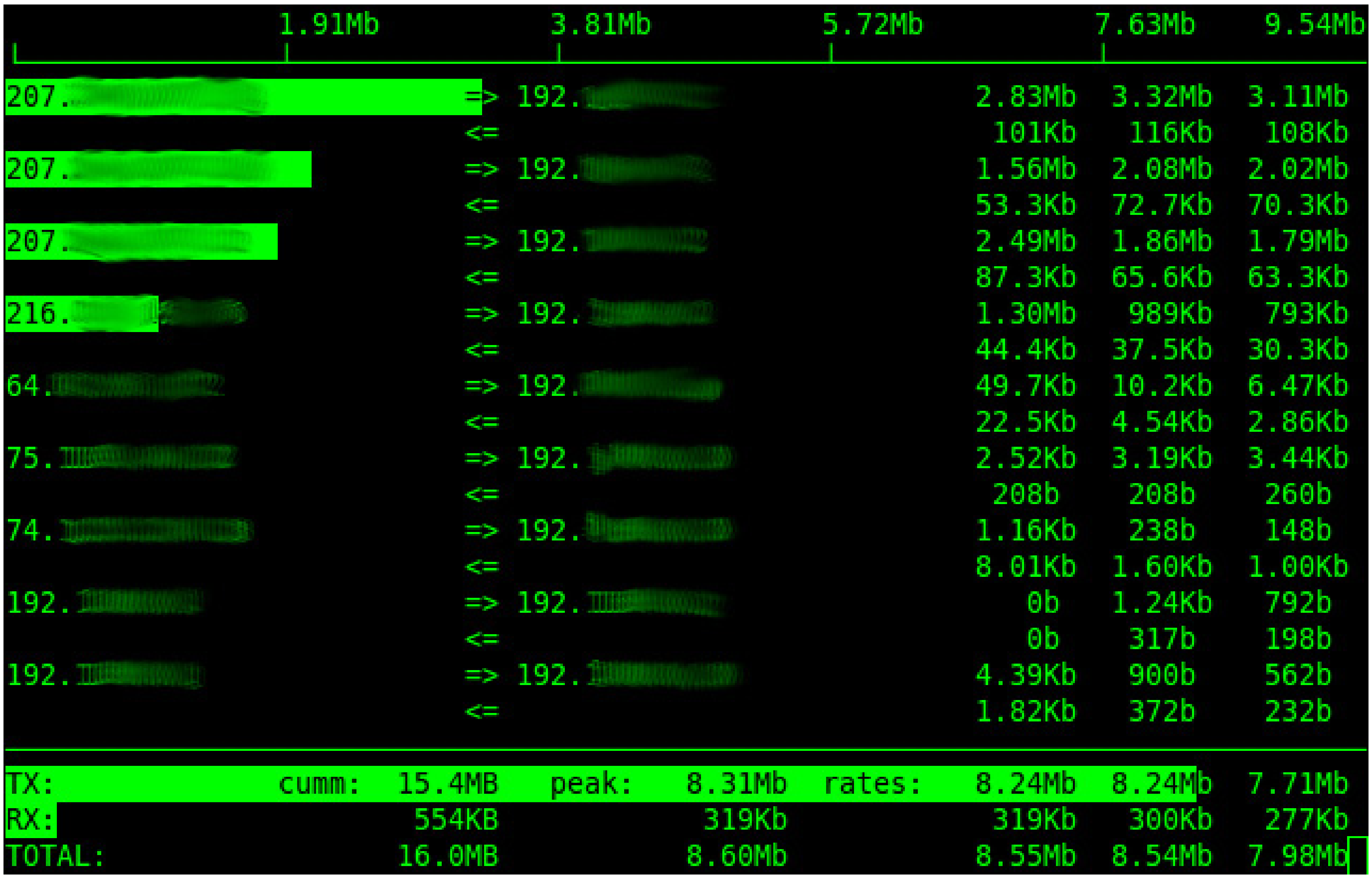
Who's using up all the bandwidth, and what are they doing? Use iftop to find out.

Longtime system administrators often take tools for granted that they've used for years and assume everyone else has heard of them. Of course, new sysadmins join the field every day, and even seasoned sysadmins don't all use the same tools. With that in mind, I decided to write a few columns where I highlight some common-but-easy-to-overlook tools that make life as a sysadmin (and really, any Linux user) easier. My last article covered sar, a tool you can use to collect and view system metrics over time. This time, I discuss a program that's handy for viewing real-time network performance data: iftop.

Anyone who's had to use a network at a conference has experienced what happens when there just isn't enough network bandwidth to go around. While you are trying to check your e-mail, other people are streaming movies and TV shows, downloading distribution install disks, using p2p networks, upgrading their distributions or watching cat videos on YouTube. Although it's certainly frustrating to try to use one of those networks, imagine how frustrating it would be to be the admin in charge of that network. Whether you run a conference network, a local office network or even a Web server at your house, it can be really nice to know what is using up all of your bandwidth.

Whether you run a conference network, a local office network or even a Web server at your house, it can be really nice to know what is using up all of your bandwidth.

iftop is a Linux command-line program designed to give you live statistics about what network connections use the most bandwidth in a nice graphical form. As you may realize from the name, iftop borrows



```
                1.91Mb          3.81Mb          5.72Mb          7.63Mb    9.54Mb
└───────────────┴───────────────┴───────────────┴───────────────┴──────────────
207.[redacted]                => 192.[redacted]              2.83Mb  3.32Mb  3.11Mb
                              <=                                101Kb   116Kb   108Kb
207.[redacted]                => 192.[redacted]              1.56Mb  2.08Mb  2.02Mb
                              <=                             53.3Kb  72.7Kb  70.3Kb
207.[redacted]                => 192.[redacted]              2.49Mb  1.86Mb  1.79Mb
                              <=                             87.3Kb  65.6Kb  63.3Kb
216.[redacted]                => 192.[redacted]              1.30Mb   989Kb   793Kb
                              <=                             44.4Kb  37.5Kb  30.3Kb
64.[redacted]                 => 192.[redacted]              49.7Kb  10.2Kb  6.47Kb
                              <=                             22.5Kb  4.54Kb  2.86Kb
75.[redacted]                 => 192.[redacted]              2.52Kb  3.19Kb  3.44Kb
                              <=                               208b    208b    260b
74.[redacted]                 => 192.[redacted]              1.16Kb    238b    148b
                              <=                             8.01Kb  1.60Kb  1.00Kb
192.[redacted]                => 192.[redacted]                  0b  1.24Kb    792b
                              <=                                 0b    317b    198b
192.[redacted]                => 192.[redacted]              4.39Kb    900b    562b
                              <=                             1.82Kb    372b    232b
───────────────────────────────────────────────────────────────────────────────
TX:             cumm:  15.4MB   peak:   8.31Mb  rates:   8.24Mb  8.24Mb  7.71Mb
RX:                     554KB            319Kb            319Kb   300Kb   277Kb
TOTAL:                 16.0MB           8.60Mb           8.55Mb  8.54Mb  7.98Mb
```

Figure 1. iftop output—the IPs have been smudged to protect the innocent.

a lot of ideas from the always-useful load troubleshooting tool top. Like top, iftop updates automatically every few seconds, and like top, by default, it sorts the output you see by what's using the most resources. Where top is concerned with processes and how much CPU and RAM they use, iftop is concerned with network connections and how much upload and download bandwidth they use.

Even though iftop is packaged for both Red Hat- and Debian-based distributions, it's probably not installed by default, so you will need to install the package of the same name. In the case of Red Hat-based distributions, you might have to pull it down from a third-party repository. Once it's installed, the simplest way to get started is just to run iftop as the root user. iftop will locate the first interface it can use and start listening in on the traffic and display output similar to what you see in Figure 1. To close the program, press q to quit just like with top.

At the very top of the screen is a scale that goes along with the bar graph iftop might display with each connection. The next rows of output correspond to

For me, the really great thing about iftop is that it's a relatively simple command-line tool.

each network connection between a pair of hosts. In between the two hosts are arrows that let you know the direction the traffic is flowing. The final three columns provide average bandwidth for each connection during the last 2, 10 and 40 seconds, respectively. So for instance, the very top connection in Figure 1 has averaged around 2.83Mb during the last 2 seconds, 3.32Mb during the last 10 seconds and 3.11Mb during the last 40 seconds. Underneath all the transmit and receive columns at the bottom of the screen are a series of statistics for overall transmitted and received traffic (TX and RX, respectively) including 2-, 10- and 40-second averages for both those and, finally, the totals for the interface.

Note: if you have a server with multiple interfaces, you may want iftop to monitor a different interface from the default. Just add `-i` followed by the interface to monitor when you launch iftop. For instance, to monitor eth2, I would type `iftop -i eth2`.

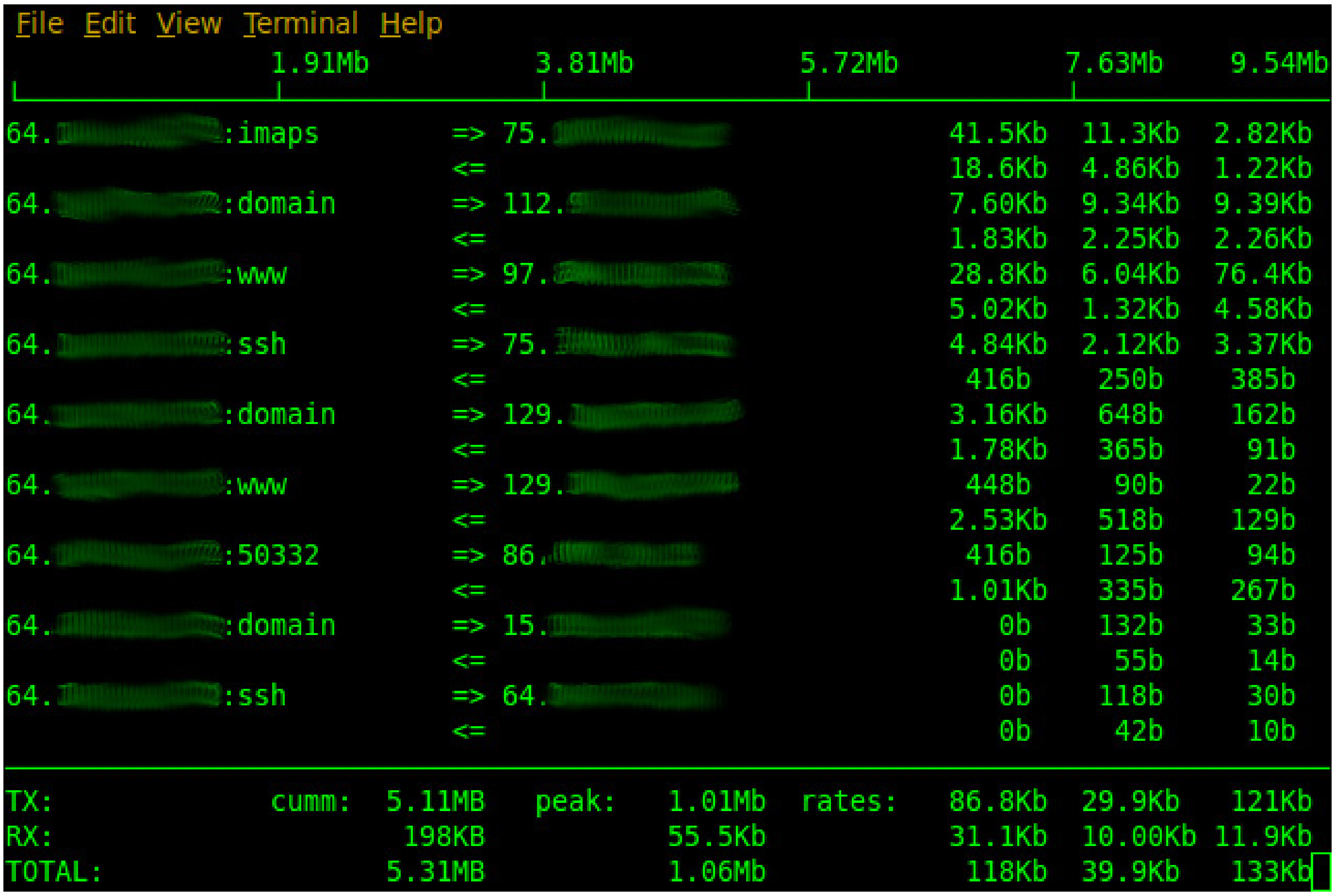
Disable DNS Lookups

By default, when you run iftop, it will try to translate all of the IP addresses into hostnames. Sometimes this can be useful if you are diagnosing issues on a local network; however, like with a lot of other network diagnostics tools, resolving all of those IPs can slow down the program and also may contribute to the traffic you see in the output. The solution is to run iftop with the `-n` argument, so it just shows you IP addresses for everything (you always can run a DNS query against an IP you are interested in, in another window). Alternatively, if you already have iftop running, you can press n to disable DNS lookups.

Show Port Data

When you run iftop on a server that might serve multiple purposes, it can be handy to know whether all of that upstream traffic is accessing your Web server, your mail server or something else. Alternatively, if you are trying to figure out what's using up all of your download bandwidth, it can be handy to see whether the top connections are Web connections or some rsync job you have running. To figure all of this out, iftop allows you to toggle the port display on and off. Press the p key while iftop is running, and it will display the ports used for both the source and destination IP for all traffic.

The one big downside to showing both the source and destination ports used for a connection is that you'll find in many cases you are concerned only



```
File  Edit  View  Terminal  Help
                1.91Mb          3.81Mb          5.72Mb          7.63Mb     9.54Mb
└───────────────┴───────────────┴───────────────┴───────────────┴──────────────────
64.[redacted]:imaps         => 75.[redacted]              41.5Kb  11.3Kb  2.82Kb
                            <=                            18.6Kb  4.86Kb  1.22Kb
64.[redacted]:domain        => 112.[redacted]             7.60Kb  9.34Kb  9.39Kb
                            <=                            1.83Kb  2.25Kb  2.26Kb
64.[redacted]:www           => 97.[redacted]              28.8Kb  6.04Kb  76.4Kb
                            <=                            5.02Kb  1.32Kb  4.58Kb
64.[redacted]:ssh           => 75.[redacted]              4.84Kb  2.12Kb  3.37Kb
                            <=                             416b    250b    385b
64.[redacted]:domain        => 129.[redacted]             3.16Kb   648b    162b
                            <=                            1.78Kb   365b     91b
64.[redacted]:www           => 129.[redacted]              448b     90b     22b
                            <=                            2.53Kb   518b    129b
64.[redacted]:50332         => 86.[redacted]               416b    125b     94b
                            <=                            1.01Kb   335b    267b
64.[redacted]:domain        => 15.[redacted]                 0b    132b     33b
                            <=                               0b     55b     14b
64.[redacted]:ssh           => 64.[redacted]                 0b    118b     30b
                            <=                               0b     42b     10b
───────────────────────────────────────────────────────────────────────────────────
TX:            cumm:  5.11MB   peak:   1.01Mb  rates:   86.8Kb  29.9Kb   121Kb
RX:                    198KB           55.5Kb           31.1Kb  10.00Kb 11.9Kb
TOTAL:                5.31MB           1.06Mb            118Kb  39.9Kb   133Kb
```

Figure 2. iftop with only the source ports displayed.

with one or the other. For instance, if you are running a Web server, you may notice that a lot of traffic is going to your Web port (labeled www in iftop), but all of the ports used by IPs accessing your Web server use all sorts of high ports. In that case, you can press either S or D to toggle the display of either source or destination ports, respectively. Figure 2 shows an example of iftop output where I've chosen to display only the source ports.

For me, the really great thing about iftop is that it's a relatively simple command-line tool. It's true that a number of other programs exist that can provide fancy Web-based graphs of your network traffic, and I think those are great for trending network data just like they are for trending system load and other metrics. What I like about iftop is the same thing I like about top—when there's a problem, you can get instant real-time data about your system that updates as the situation progresses.■

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.

SHAWN POWERS

LTSP, Part III: Servers Unite!

What's better than a lab full of thin clients? Ten labs full of thin clients! This article shows how to scale LTSP. If you think more is better, read on!

My last two articles have focused on Linux thin clients. I've covered how to set them up, how to administer them and even how best to tweak your server to meet your needs. This article finishes the series by describing how to scale LTSP in large environments. There are a few different methods, and each has its advantages and disadvantages.

The Ants Go Marching One by One

One of the options for a large LTSP roll-out is simply not to scale at all. This may seem like a cop-out, but since LTSP works so well in a classroom (or similar) environment, simply adding a second network card to your workstation class computers is a simple way to serve 4–5 thin clients per classroom. In fact, if your network infrastructure is old and can't support the large bandwidth requirements thin clients demand, this type of setup is perfect. The high bandwidth is managed by a cheap desktop switch, and the only traffic on your main building network is for Internet and file sharing.

When you're setting up your network in this way, it's important to realize the thin clients in one classroom won't be able to see the thin clients in another classroom. Since every classroom has its own server, every server is setting up its own NAT for the thin clients to live. If you plan to do something fancy like sharing a printer connected to a thin client, keep in mind only the server will be visible to the rest of the network. (You still can get around this limitation by sharing the thin-client-connected printer via the classroom server. Linux is incredibly flexible!)

If a classroom-server-type setup sounds ideal for you, setting it up is as simple as configuring the classroom workstation (often the "teacher station") as an LTSP server. It still will function perfectly fine as a workstation, but because it

Because multiple DHCP servers on a network can be messy, I recommend using a separate computer for DHCP and disabling the service on each LTSP service.

will be running the LTSP services in the background, it will share its resources with the thin clients connected to the second Ethernet card. Depending on the speed of the workstation, and whether your thin clients are running local apps, a mid-range workstation can support anywhere from 3–8 computers. Remember, running Firefox and Flash as local apps will really take a huge burden off the teacher's workstation.

Single NIC Servers, or the "Herding Cats" Method

Oddly enough, the single NIC setup is the toughest to set up, the most difficult to load balance, and the hardest to update, but it's the system I use personally. I suspect it's because this used to be the only way to scale, and I'm comfortable, so I'm reluctant to change. I know, it's a pitiful excuse, but a truthful one. Basically, in this scenario, you have any number of servers running LTSP in your server closet, using only a single network interface to connect to the network. One of the servers, or more commonly a separate DHCP server, tells the clients which server to connect to. I usually chop up the network by location or purpose, and point similar computers at a common server.

Because multiple DHCP servers on a network can be messy, I recommend using a separate computer for DHCP and disabling the service on each LTSP service. You can copy the dhcpd.conf file from one of the LTSP servers and tweak it to serve your needs. You can leave a global directive to point all servers to a single LTSP server, and then manually add your "chunks" of clients one by one. Here's a snippet of my dhcpd.conf file, specifying a group of clients to boot to a specific server:

```
# LTSP Server 1 clients
group   {
        next-server 192.168.1.200; #Server 1's IP address
        filename "/ltsp/i386/pxelinux.0";
        # Hosts
        host client1 { hardware ethernet 00:90:c2:cd:85:60;
        ➥fixed-address 192.168.1.101; }
        host client2   { hardware ethernet 00:16:cb:bc:ad:37;
        ➥fixed-address 192.168.1.102; }
        }
```

Remember to base your dhcpd.conf file

on a file from an LTSP server, so you get all the pertinent thin-client options in the global section of the conf file.

In the above section of the conf file, you'll notice the `next-server` statement. That tells the thin client where to go for its TFTP boot file. The `filename` directive tells the client what boot file to download from the TFTP server. It is possible to host the PXE boot files on a central server, and then point the thin clients to another server for the rest of the boot process, but it gets complicated really quickly. I find it's much easier to hand the thin client off to the LTSP server right inside the dhcpd.conf file. It means each LTSP server must be running TFTPD, but since each server is already waiting for thin clients to connect, it's not a big deal.

If you do the DHCP configuration/trickery above, configuring each LTSP server is pretty much the same as the standard server model. You still need to create the chroot, and you still need to run `ltsp-update-image` anytime you make a change to the chroot, and any changes you make must be done to each and every LTSP server on your network. It isn't an efficient way to manage a network, but it's conceptually very simple and easy to troubleshoot when something goes wrong. The big downside is replicating changes to every server and updating each chroot. If you're thinking there must be a better way, well, you're right!

One Server to Rule Them All

A few years ago, some of the brilliant folks working on the LTSP Project (Stéphane Graber was the main hero here) realized that NBD was so efficient at serving the chroot image, it really wouldn't be a bottleneck to use a single NBD chroot image and spread the actual workload of running applications across a cluster of servers. As with most great ideas, this introduced a bunch of problems, but for the most part, these problems have been addressed, and LTSP Cluster is a viable, easily scalable way to implement a large LTSP install.

There is a Web site for LTSP Cluster (**http://www.ltsp-cluster.org**), but to be honest, it's not the most up-to-date site from which to learn. The best walk-through I've been able to find is on the Ubuntu site: **https://help.ubuntu.com/community/UbuntuLTSP/LTSP-Cluster**. Following the directions on that site will get you a fully functional cluster just waiting to be tweaked. For the purpose of this article, I'll explain what happens in a cluster environment, so hopefully the concept makes sense. Because cluster support is arguably the least-developed aspect of LTSP, you might decide it's not worth the effort.

When running LTSP Cluster, you must dedicate a server (or virtual machine) as the NBD server for the entire network. You also can create a highly available scenario with fancy DHCP/DNS magic, but for most scenarios, a single NBD server is the way to go. The chroot environment is very similar to the traditional LTSP model, but when running the `ltsp-build-client` script, you have to add the `--ltsp-cluster` flag so the clustering options are built in.

Once the NBD image is created, separate servers, referred to as "application servers", are added to the cluster configuration. When a client boots up, the NBD server queries the attached application servers, determines which server is least taxed and assigns the thin client to that server. From that point on, until reboot, the client runs its applications via SSH from the application server it was assigned. It still is possible (and recommended) to use local apps in order to utilize the power of the thin-client device.

The main configuration difference with LTSP Cluster is that the lts.conf file has been replaced with a Postgres database and a Web interface. Once you get the hang of using the slightly confusing interface, you'll find it is a powerful way to organize configurations based on individual client machines (Figure 1). I also created a short video a couple years ago, demonstrating how to add nodes in the Web interface, because it can be a bit daunting at first glance: **http://youtu.be/7QdYW-NT_sw**.

LTSP Cluster is truly the most advanced way to scale your thin-client environment. Because it does add a couple layers of complexity to your rollout, I recommend that before you implement Cluster, you at least are familiar with how LTSP works with its traditional installation method. In my experience, when LTSP Cluster works, it works very well. When something goes wrong, troubleshooting is tougher than with a traditional setup. As with most things, your mileage may vary.

Thin Shmin, We Like Our Clients Fat!

Finally, there is arguably the least-server-intensive model for implementing thin clients, and that is not to use thin clients. Yes, yes, my mastery of the obvious is astounding, but seriously, LTSP supports a

Open Frame PPC

New - PPC-E10

- 10" 1024x600 TFT LED Backlit LCD
- Fanless ARM9 400Mhz Processor
- Analog Resistive Touchscreen
- Up to 1 GB Flash & 256 MB RAM
- 10/100 Base-T Ethernet
- 3 RS232 & 1 RS232/422/485 Port
- 1 USB 2.0 (High Speed) Host port
- 1 USB 2.0 (High Speed) OTG port
- 2 Micro SD Flash Card Sockets
- SPI & I2C ports
- I2S Audio Interface w/ Line-in/out
- Operating Voltage of 12 to 26 Vdc
- Optional 2D Accelerated Video & Decoder
- Pricing starts at $595 for Qty 1

2.6 KERNEL

The PPC-E10 Compact Panel PC comes ready to run with the Operating System installed on Flash Disk. Apply power and watch either the Linux X Windows or the Windows CE User Interface appear on the vivid 10" color LCD. Interact with the PPC-E10 using the responsive integrated touch-screen. Everything works out of the box, allowing you to concentrate on your application, rather than building and configuring device drivers. Just Write-It and Run-It. For additional information please contact EMAC.

www.emacinc.com/panel_pc/ppc_e10.htm

Since 1985 OVER 27 YEARS OF SINGLE BOARD SOLUTIONS

EMAC, inc. EQUIPMENT MONITOR AND CONTROL

Phone: (618) 529-4525 • Fax: (618) 457-0110 • Web: www.emacinc.com

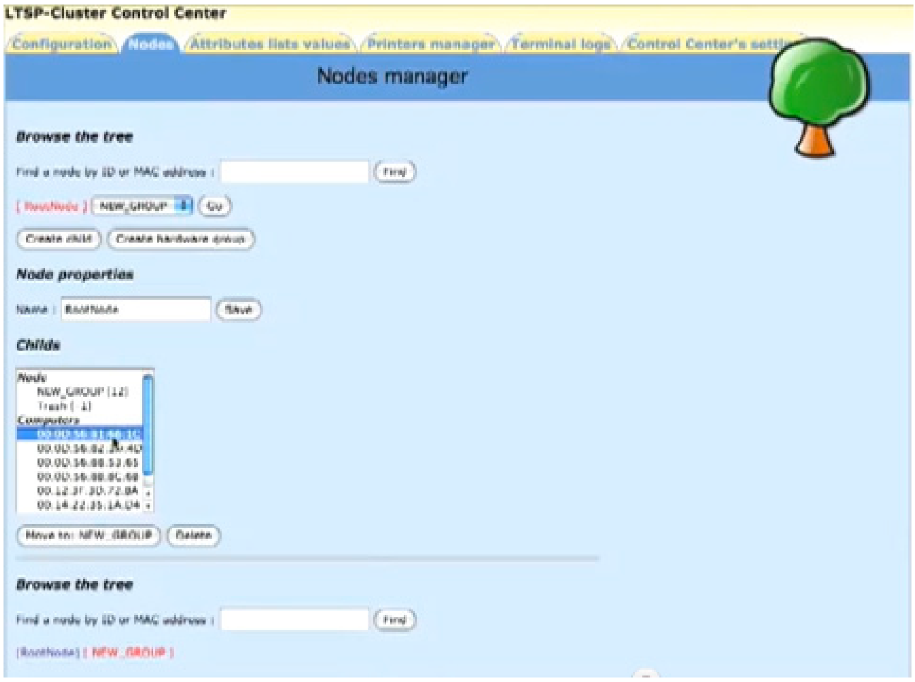


Figure 1. The Web interface is confusing, but useful.

"fat-client" model. Basically, rather than depending on the LTSP server for running applications, the NBD chroot image contains a complete Linux install, so every application, even down to the window manager, is run from the client. In this model, all the server is responsible for is serving out the NBD image. To enable the fat-client mode, you need to add the `--fat-client` flag when running `ltsp-build-client` to create your chroot.

As with every scenario, this has its shortcomings. Because absolutely everything is run from the client, the fat-client model requires powerful machines. In fact, the specs for a fat client are similar to that of a standalone computer, with the exception of a hard drive. The fat clients require a substantial network connection to the server, and unless you have a very fast network infrastructure, you might be better off with actual standalone computers.

If you have a bunch of workstations without hard drives, or if the maintenance time saved by having a single chroot environment for all your workstations is worth it, fat clients can be incredibly useful. I've noticed, however, that even top-end Atom-based thin-client devices tend to bog down in fat-client mode. If you're going to depend on your clients to run the complete OS, it's important to test them first to make sure they're up to snuff.

Because you may have a combination of new and old client machines, it is also possible to mix and match the fat-client and thin-client model from the same server. It means creating two NBD images, but assigning the proper image is as simple as specifying a new filename in your dhcpd.conf file. This is a nice way to take advantage of a few really powerful workstations without making your slower hardware obsolete.

All Things Are Possible, Not All Things Are Wise

My last few articles have torn the LTSP system apart and looked at its juicy insides. Perhaps that's a gross

Table 1. Scaling Models, the Pros and Cons

MODEL	DESCRIPTION	BEST FOR	ADVANTAGES	DISADVANTAGES
Classroom Server	Workstation class machines with extra Ethernet card server, 3–8 clients per classroom.	Environments with poor network infrastructure and/or lack of big servers.	Minimal server costs, utilizes existing workstations as mini-servers, keeps thin-client bandwidth off main network.	Classroom thin clients isolated from each other, potential for double-NAT problems, more servers means more configurations to manage.
Single Ethernet	Centralized servers have a single Ethernet port, with a separate DHCP server allocating clients to servers.	Large network installations with sufficient network infrastructure.	Troubleshooting is simple, clients can be reassigned by modifying dhcpd.conf file, configuration is similar to standalone model.	Manual load balancing is cumbersome and often incorrectly done, multiple servers to configure means repeating the same admin tasks over and over, no automatic failover if server goes down.
LTSP Cluster	Centralized NBD server load balances over any number of application servers.	Large installations with sufficient network infrastructure and experienced LTSP administrator.	Clustering allows for central configuration of all clients and single chroot for quick updates.	Web interface slightly cumbersome, layers of complexity make troubleshooting more difficult, NBD server failure can take down entire network.
LTSP Fat Client	Powerful client machines load entire operating system over NBD, running all applications locally.	Powerful client machines with sufficient network infrastructure.	Client machines handle all computing, so server requirements are minimal.	Requires quite powerful client machines and solid network infrastructure. Standalone hard drive installs should be considered.

metaphor, but hopefully, it's a little more clear how powerful thin clients can be for your organization. My favorite part is how many different ways LTSP can be implemented. If you're just starting the process, a few classroom labs are the perfect way to see if LTSP will work for you. In fact, by recycling older workstations and using them as thin clients, the cost of implementation is often zero. It just takes a few adventurous users willing to learn a new way to compute.

As the system administrator, you have several ways to set up your LTSP system. Table 1 gives some pros and cons of each scaling method discussed here, but if you're just starting, I recommend going with a single server and a handful of clients. Once you get to the point of scaling, you'll have a much better idea of what fits into your environment best. If you have questions along the way, feel free to drop me an e-mail or ask the professionals in the IRC #ltsp channel on Freenode.■

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.

Logic Supply's LGX Extended Temperature AU115 Atom System

The key selling points for Logic Supply's LGX Extended Temperature AU115 Atom System include the fanless system's enhanced vibration and thermal tolerances and its very compact footprint. The AU115 system, says Logic Supply, offers reliable computing in an operating temperature environment of –20°C to 60°C and is easily mountable, making it ideal for deployment in remote monitoring, outdoor kiosks, data acquisition and networking applications. With no fans or vent openings, the AU115 is protected from exposure to the dust, dirt and grime typically found on factory floors or in mobile environments. The device also conforms to the iMIL-STD-810F standards for vibration and shock, and it can be configured with a solid-state device for storage to extend the lifetime of the system and ensure data integrity.
http://www.logicsupply.com/au115

Chris
Assembly Team Manager
JT01 Intel® Atom™ Fanless System with Ubuntu Linux
Fully assembled, plug-and-play system with HDD, RAM, Wi-fi, and OS.
Low-profile, power efficient, and quiet.
Genuine expertise.
LOGIC SUPPLY®
www.logicsupply.com/linux

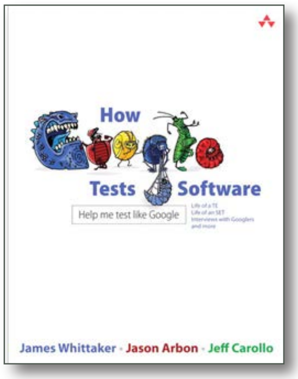
AdaCore's GNAT Pro, CodePeer

AdaCore recently announced two major product upgrades—GNAT Pro 7.0 and CodePeer 2.1—both of which support the upcoming Ada 2012 language standard. GNAT Pro is AdaCore's full-featured, multilanguage, flagship development environment complete with libraries, bindings and a range of supplementary tools. Besides the Ada 2012 support, GNAT Pro's key new features include the new GNATtest automatic test framework and numerous enhancements to several existing tools. Meanwhile, CodePeer is AdaCore's advanced static analysis tool that helps developers detect potential runtime and logic errors in Ada programs. By mathematically analyzing every line of software and considering every possible input and every path through the program, CodePeer can be used early in the development life cycle to identify defects when they are much less costly to repair.
http://www.adacore.com

Roderick W. Smith's *Linux Essentials* (Sybex)

It's a positive sign for our revolution to see the collection of basic Linux books continuing to expand and diversify. *LJ* writer Roderick W. Smith's new title *Linux Essentials*, published by Sybex, adds a unique twist by adding full-color images to the mix of resources that aid readers in understanding the essentials of Linux. *Linux Essentials* features a learn-by-doing approach that clearly identifies the concepts to be covered and explores them in a hands-on tutorial format. Topics include installation, desktop configuration, management of files and filesystems, remote administration, security and more. Fedora is the distribution discussed in the book. *Linux Essentials* is a great way to share the gift of Linux with our new, less-technical friends and family members.
http://www.sybex.com

James A. Whittaker, Jason Arbon and Jeff Carollo's *How Google Tests Software* (Addison-Wesley)

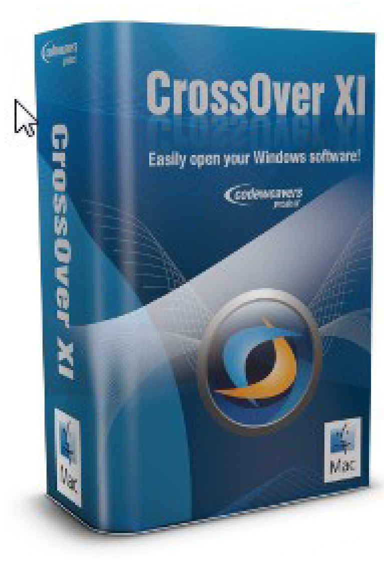

How Google Tests Software
Help me test like Google
James Whittaker · Jason Arbon · Jeff Carollo

Things done at "Google scale" are mind boggling. Hundreds of millions of lines of code distributed across millions of source files. Billions of build actions prompting millions of automated tests daily. Hundreds of thousands of browser instances daily. Apps released continuously into production for massive worldwide use. Folks, software testing doesn't get any tougher than this. To do all of this right, Google is pioneering the future of testing and automation. Enter the new book *How Google Tests Software* from author trio James A. Whittaker, Jason Arbon and Jeff Carollo, which seeks to offer mere mortals the chance to learn from the Google experience. The authors distill wisdom from legendary Google Testing leader James Whittaker and other top Google experts to reveal exactly how the Internet giant tests software and offer new best practices anyone can use. Some of these practices include techniques for analyzing risk and planning tests; implementing exploratory, black-box, white-box and acceptance testing; testing "Docs & Mocks"; and much more.
http://www.informit.com

The Portland Group's PGI 2012 Compilers and Development Tools

PGI®

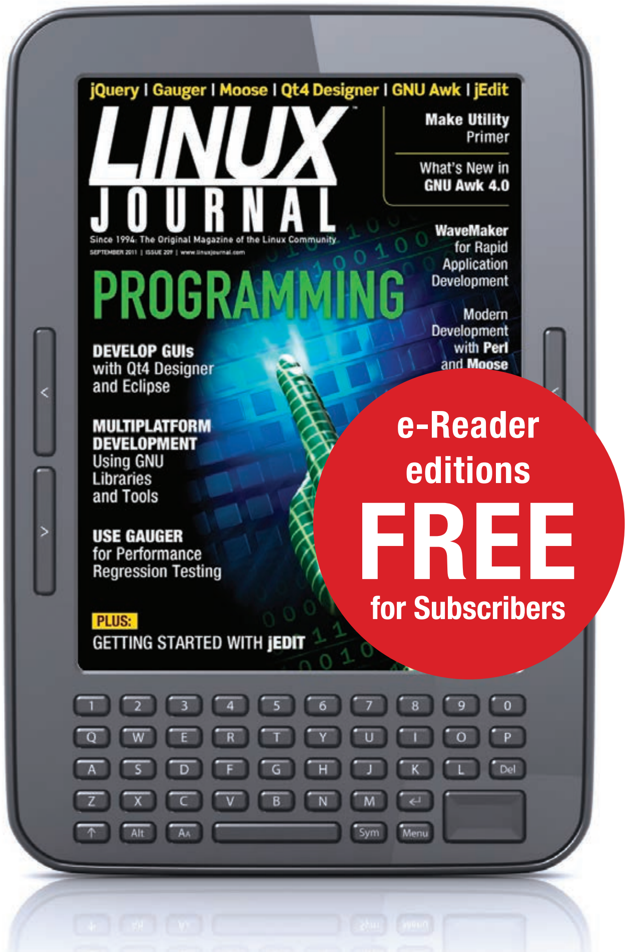
The Portland Group says that its new PGI 2012 line of parallelizing compilers and development tools helps developers leverage the huge performance potential of GPUs better than ever before. PGI 2012, which supports Linux, OS X and Windows, is the Portland Group's first general release to include support for the OpenACC directive-based programming model for NVIDIA CUDA-enabled GPUs. The updated PGI Accelerator FORTRAN and C compilers are targeted at scientists and engineers who are not full-time programmers, freeing developers to focus on optimizing their algorithms. The 2012 release is also the company's first to include the fully feature-enabled PGI CUDA C/C++ compiler for multicore x64 CPUs from Intel and AMD. Finally, PGI 2012 includes a number of performance and feature enhancements for multicore x64 processor-based HPC systems.
http://www.pgroup.com



CodeWeavers' CrossOver XI

The only thing not simple about CrossOver XI is "how we pronounce the name of the product", said CodeWeavers' ever irreverent CEO, Jeremy White, recently about his company's new complete package for running Windows-based apps on Linux or Mac. The boss is unsure whether its customers will pronounce the name "CrossOver Ex-Eye", or "CrossOver Eleven", "CrossOver Exy", or his own favorite, "CrossOver Zigh". Regardless of what you call it, CrossOver XI combines all previous versions of CodeWeavers' CrossOver software into one simple download, giving "everyone in our universe immediate access to thousands of Windows applications otherwise unavailable without a Microsoft license", added White. Applications, which number in the thousands, range from Wizard 101, to Microsoft Office 2010 to nostalgia-oozing WordPerfect 3.1.

http://www.codeweavers.com

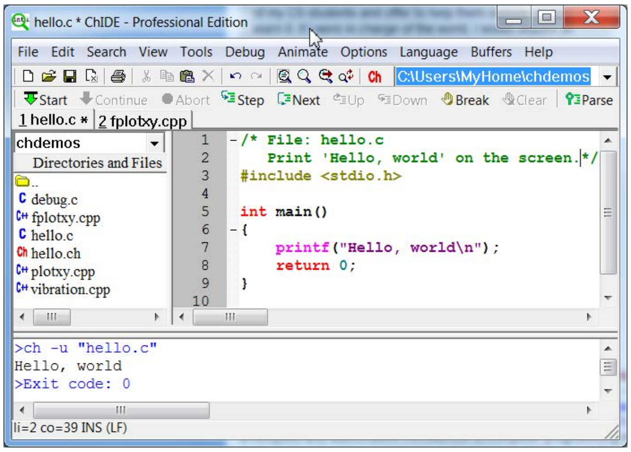

jQuery | Gauger | Moose | Qt4 Designer | GNU Awk | jEdit
LINUX JOURNAL
PROGRAMMING
DEVELOP GUIs with Qt4 Designer and Eclipse
MULTIPLATFORM DEVELOPMENT Using GNU Libraries and Tools
USE GAUGER for Performance Regression Testing
PLUS: GETTING STARTED WITH jEDIT
Make Utility Primer
What's New in GNU Awk 4.0
WaveMaker for Rapid Application Development
Modern Development with Perl and Moose
e-Reader editions FREE for Subscribers

Inductive Automation's Ignition

Deploy the new Ignition 7.4, says maker Inductive Automation, and you'll feel as if you had an extra pair of hands helping you knock out your HMI/SCADA/MES development projects. Inductive Automation bills the Java/Web-based Ignition as the "simplest and most innovative software for HMI, SCADA and MES applications in the automation industry". Ignition connects to nearly any major SQL database or operating system and features a built-in OPC-UA server with drivers for the most common PLC brands. New in v7.4 are innovations, such as dynamic component templates, user-defined types, scripting with Python 2.5 and project templates. All of the above, says Inductive Automation, carve out repetitive development tasks, leaving more time for innovating, improving testing and securing projects.

http://www.inductiveautomation.com

SoftIntegration's Ch 7.0 and Embedded Ch


hello.c * ChIDE - Professional Edition
File Edit Search View Tools Debug Animate Options Language Buffers Help
C:\Users\MyHome\chdemos
Start Continue Abort Step Next Up Down Break Clear Parse
1 hello.c * 2 fplotxy.cpp
chdemos
Directories and Files
debug.c
fplotxy.cpp
hello.c
hello.ch
plotxy.cpp
vibration.cpp
li=2 co=39 INS (LF)

```
/* File: hello.c
   Print 'Hello, world' on the screen.*/
#include <stdio.h>

int main()
{
    printf("Hello, world\n");
    return 0;
}
```

```
>ch -u "hello.c"
Hello, world
>Exit code: 0
```

SoftIntegration notes that an instructor using its Ch C/C++ interpreter, now in version 7.0, has called it "as close to a perfect teaching environment" he has ever seen. A programmer customer said using Ch is "the fastest way to write and deploy embeddable C/C++ scripting programs across platforms". Ch 7.0 and Embedded Ch 7.0 are an embeddable C/C++ interpreter for cross-platform scripting, 2-D/3-D plotting, numerical computing, shell programming and embedded scripting. The 7.0 release adds new features to make Ch especially appealing for secondary school and college students to learn computing and math. These include many new math functions and features for plotting, QuickAnimation, ChIDE and Embedded Ch. Embedded Ch allows users to embed Ch into an C/C++ application, such as game, semiconductor ATE, SoC, CAM, IC, PCB, RF, MEMS or LED as a scripting engine. Commercial and academic licenses are available.

http://www.softintegration.com

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

2012

USENIX Federated Conferences Week

June 12–15, 2012 Boston, MA

www.usenix.org/fcw12

Networked Systems for Developing Regions
Storage Provenance
Systems Research Cloud Computing
Networked Systems for Developing Regions
Web Application Development
Provenance Systems Research
Cloud Computing Storage

Back for 2012, USENIX is combining established conferences and workshops into a week of research, trends, and community interaction. Events include:

///// USENIX ATC '12

2012 USENIX Annual Technical Conference
Wednesday–Friday, June 13–15 www.usenix.org/atc12

WebApps '12 /////

3rd USENIX Conference on Web Application Development
Wednesday, June 13 www.usenix.org/webapps12

///// HotCloud '12

4th USENIX Workshop on Hot Topics in Cloud Computing
Tuesday–Wednesday, June 12–13 www.usenix.org/hotcloud12

HotStorage '12 /////

4th USENIX Workshop on Hot Topics in Storage and File Systems
Wednesday–Thursday, June 13–14 www.usenix.org/hotstorage12

///// TaPP '12

4th USENIX Workshop on the Theory and Practice of Provenance
Thursday–Friday, June 14–15 www.usenix.org/tapp12

NSDR '12 /////

6th USENIX/ACM Workshop on Networked Systems for Developing Regions
Friday, June 15 www.usenix.org/nsdr12

///// Why Attend

You'll leave the week with insights into the future of many hot topics. Your access to leading researchers and industry experts will not only provide answers to your most burning questions, but will also help you build lasting connections across multiple disciplines. Take full advantage of this opportunity and learn from the top researchers, practitioners, and authors all in one place, at one time.

EARLY BIRD DISCOUNT
REGISTER by
MAY 21 &
SAVE

usenix ASSOCIATION

www.usenix.org

Stay Connected...

http://www.usenix.org/facebook
http://www.usenix.org/linkedin
http://twitter.com/usenix
http://blogs.usenix.org

HARDWARE

ZaReason's Valta X79

ZaReason combines Sandy Bridge CPUs and screaming-fast graphics cards to make a serious powerhouse computer.

SHAWN POWERS

About a month ago, I was contacted by Earl Malmrose of ZaReason, who wanted to know if I'd like to review ZaReason's new Linux-based desktop computer, built around the new Intel 6-Core processor and quad channel memory. I told him I'd be thrilled to review it, and asked if he'd also include a snappy ATI video card so I really could push the system to the limit using one of my favorite side hobbies, namely cryptocurrencies.

I start with a review of the system itself and finish with a bit of fun—I run the numbers and see what sort of CPU and GPU-hashing power I can get from it. Whether you think cryptocurrencies are a brilliant take on alternative economics or a dumb idea that wastes electricity, I can assure you no one knows how to overclock hardware quite like a Bitcoin miner. (I don't actually overclock this system, since I'm sure ZaReason would like it back in full working order, but I push it to the max with stock settings.)

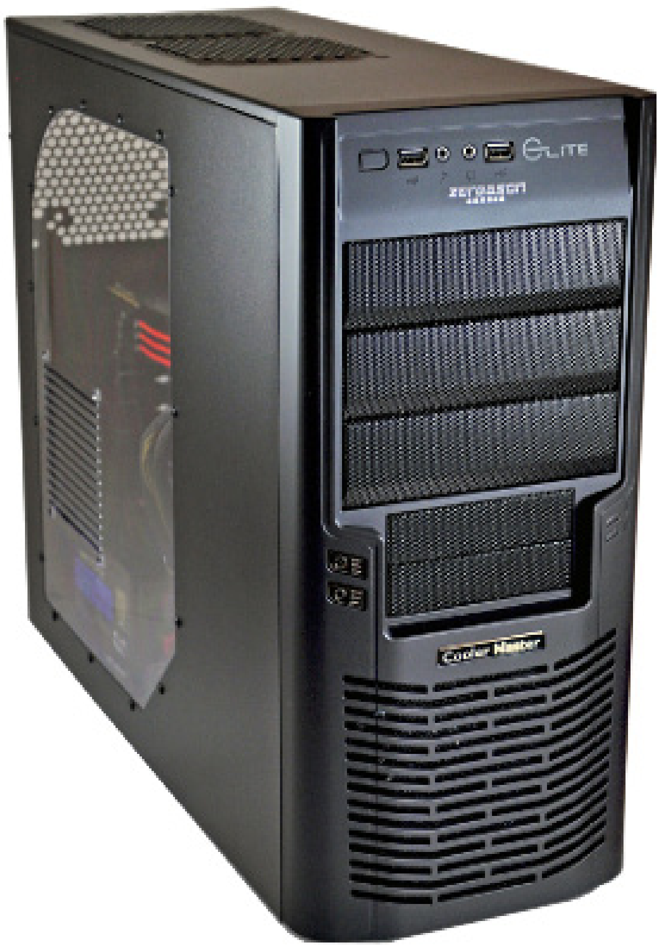


Figure 1. The Valta X79 has ZaReason branding, but is otherwise sporting an unmodified CoolerMaster case (image from http://www.zareason.com).

The Construction

Generally, if you want a case that a regular human can change parts on, you build your own machine. That's more of a guideline than a rule, but it seems most big computer companies like to make their cases as proprietary as possible. Sure, you might be able to fit your computer system into a size 4 shoebox, but good luck if you ever want to upgrade your hardware. ZaReason, on the other hand, includes a standard CoolerMaster brand mid-side tower case. I use the term "mid-size" rather sheepishly, as it's the quite large "G-Lite 430 Black" (Figure 1).

ZaReason has nicely branded the case with its company name, but otherwise left it the standard CoolerMaster unit—refreshing, no? To be fair, the case itself is a bit more flexible than I'd like, but it doesn't feel cheap by any means. Part of its flexibility is due to the massive amount of ventilation it boasts (Figure 2). Because the X79 is meant to be a powerhouse, the cooling consideration is greatly appreciated. In fact, it comes with an additional case fan installed on the side to help keep the beast within on the cool side.

On the front panel, the case sports a power button and reset button, as one would expect. It also has the following:

- Two USB 2.0 ports.

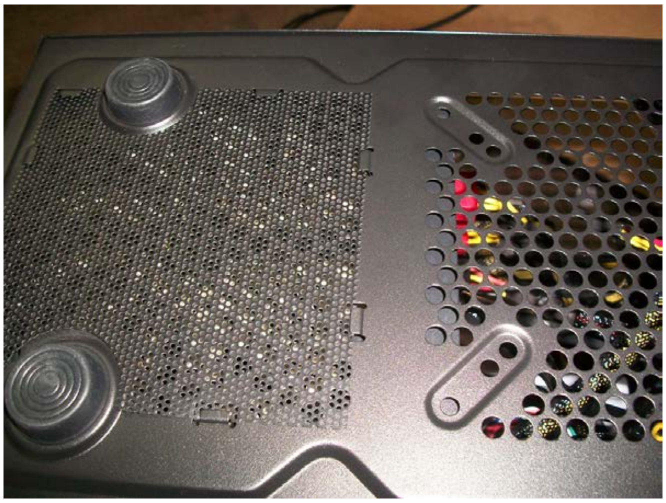


Figure 2. This ventilation is appropriate, but it does make the case feel a bit flimsy.

- Headphone and microphone jacks (analog).

The back of the case provides quite a generous set of connections (Figure 3), including:

- Six USB 2.0 ports.
- Two USB 3.0 ports (clearly marked).
- One PS/2 port.
- Gigabit Ethernet port.
- 8 channel, 7.1 analog jacks.
- Coaxial SPDIF-out.
- SPDIF-optical out.
- One 1394 port (400Mbit).

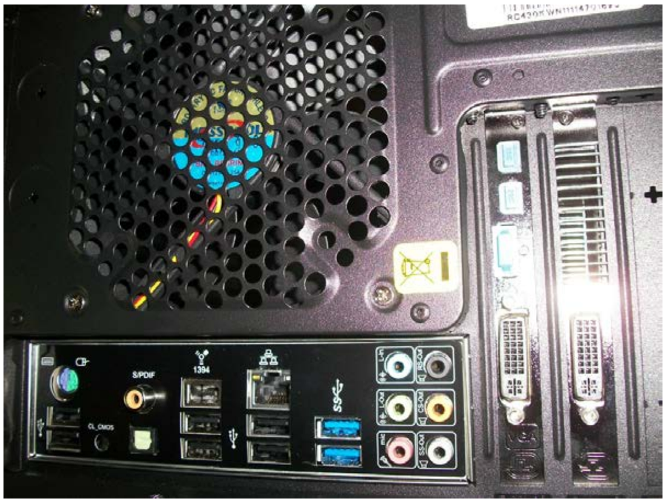


Figure 3. The ports are easy to access, and the external CMOS reset button is very convenient.

- External CMOS reset button.

Of course, all those ports really tell the story of the motherboard more than the case, so I cover that next.

The Mother (Board)

The specifications on the Valta X79 don't specify a brand name for the motherboard, so I won't focus on the name printed on the board, but rather the features it boasts. First off, in order to handle the 3930K second-generation Sandy Bridge i7 CPU, the motherboard has the Intel X79 chipset and LGA-2011 socket. (More on the CPU in a bit.) Adding to the CPU itself, the X79 has:

- Three PCI-E 3.0 x16 slots.
- Four PCI-E x1 slots.
- Four DDR3 Quad-Channel RAM slots.
- 16GB DDR3 RAM @1600MHz (4x4GB).

Another pleasant surprise was the power supply included with the Valta X79. Because this build included a sizable GPU along with the zippy CPU, ZaReason installed an 850 watt PSU. Although the wattage is about what I'd expect, I was happy to see the power supply was an 80 Plus Gold-rated unit. Often when building a machine for brute horsepower, little thought is given to efficiency. Thankfully, that's not the case here. 80 Plus Gold certification means the power supply is 87% efficient at 20% load, 90% efficient at 50% load, and 87% efficient at 100% load. As it happens, when this system is running full bore with both the GPU and CPU maxed out, it's using almost exactly 425 watts. Whether it was a coincidence or not, it means the PSU was sized exactly right for maximum efficiency. Well played, ZaReason.

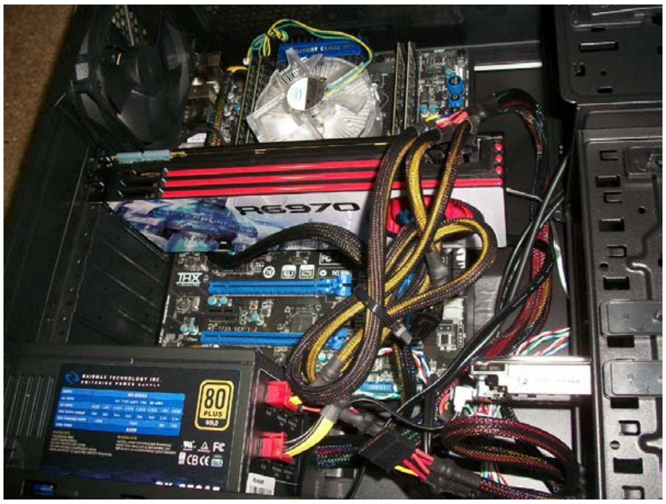


Figure 4. The interior is roomy and cabling neat, allowing for maximum airflow.

Perhaps it's raw CPU horsepower, or perhaps it's a combination of the CPU plus the quad channel RAM. The bottom line is this CPU is fast!

When 3-D Isn't Enough

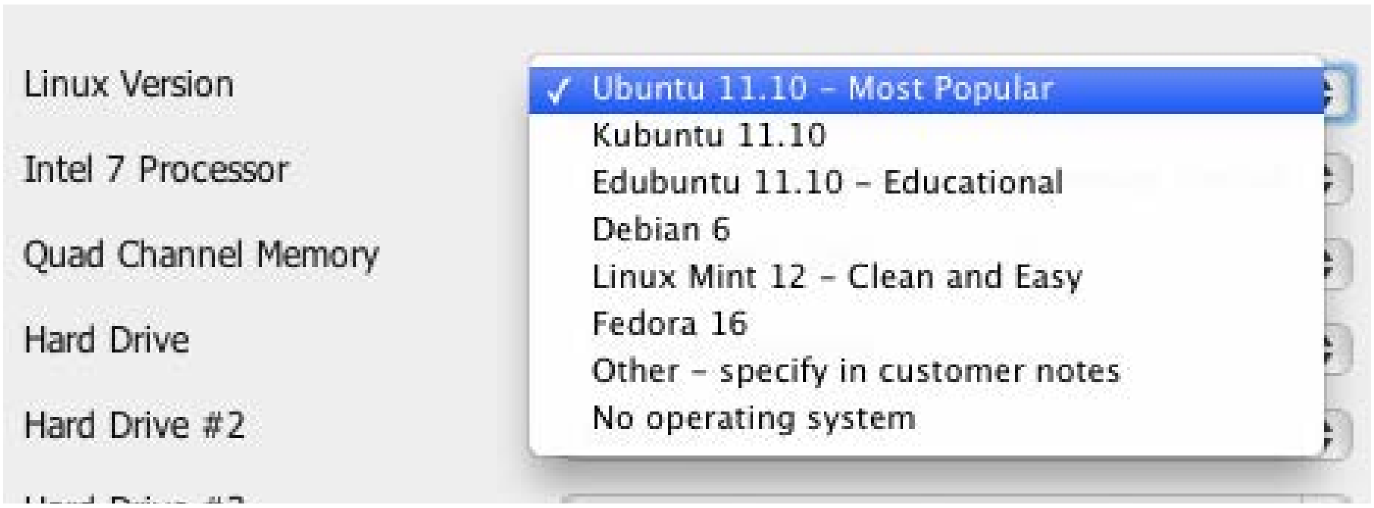
The graphics card shipped with my unit specifically was chosen with Bitcoin mining in mind. Thanks to an architecture difference between NVIDIA and ATI GPUs, the higher number of Arithmetic Logic Units (ALUs) in the ATI cards means they perform better watt for watt and dollar for dollar when it comes to the pure brute force needed for Bitcoin mining and password cracking. NVIDIA GPUs aren't subpar, they're just different. They tend to have a more-advanced onboard memory system, making them ideal for different sorts of mathematical calculations. The differences are actually rather fascinating, but because I may be alone in my fascination, I'll just leave it at that. ZaReason happily will ship either an ATI or an NVIDIA graphics card. This review unit has the ATI Radeon HD 6970.

Second-Generation i7

If the ATI 6970 is a Cadillac amongst video cards, the 6-core, second-generation 3930K i7 CPU running at 3.2GHz stock in the Valta X79 is a Ferrari amongst processors. I put the CPU through its paces mining a CPU-based cryptocurrency (Litecoin to be specific), but everything CPU-intensive was just lightning-fast.

It's often difficult to find a day-to-day task that actually puts a modern CPU to the test. Thankfully, as Linux users, we still compile quite a bit of our software, and compilation takes CPU. I took it upon myself to compile a few programs. Granted, none of the programs I compiled were enormous, but I think it's telling that when I finally typed `make`, I thought there was a dependency that `configure` missed. I got the command prompt back so quickly, that I didn't even consider the possibility the compilation had completed!

I realize I make it sound like a magical CPU, and you're thinking how adorable that Shawn guy is for being impressed by a fast CPU. The thing is, my personal PC is an AMD 6-core monster overclocked to almost 4GHz. I've compiled the same programs, and I've never been fooled into thinking there was an error when really it just finished that fast. Perhaps it's raw CPU horsepower, or perhaps it's a combination of the CPU plus the quad channel RAM. The bottom line is this CPU is fast!



Linux Version	✓ Ubuntu 11.10 – Most Popular
Intel 7 Processor	Kubuntu 11.10
Quad Channel Memory	Edubuntu 11.10 – Educational
Hard Drive	Debian 6
Hard Drive #2	Linux Mint 12 – Clean and Easy
	Fedora 16
	Other – specify in customer notes
	No operating system

Figure 5. No Windows option here (image from http://www.zareason.com).

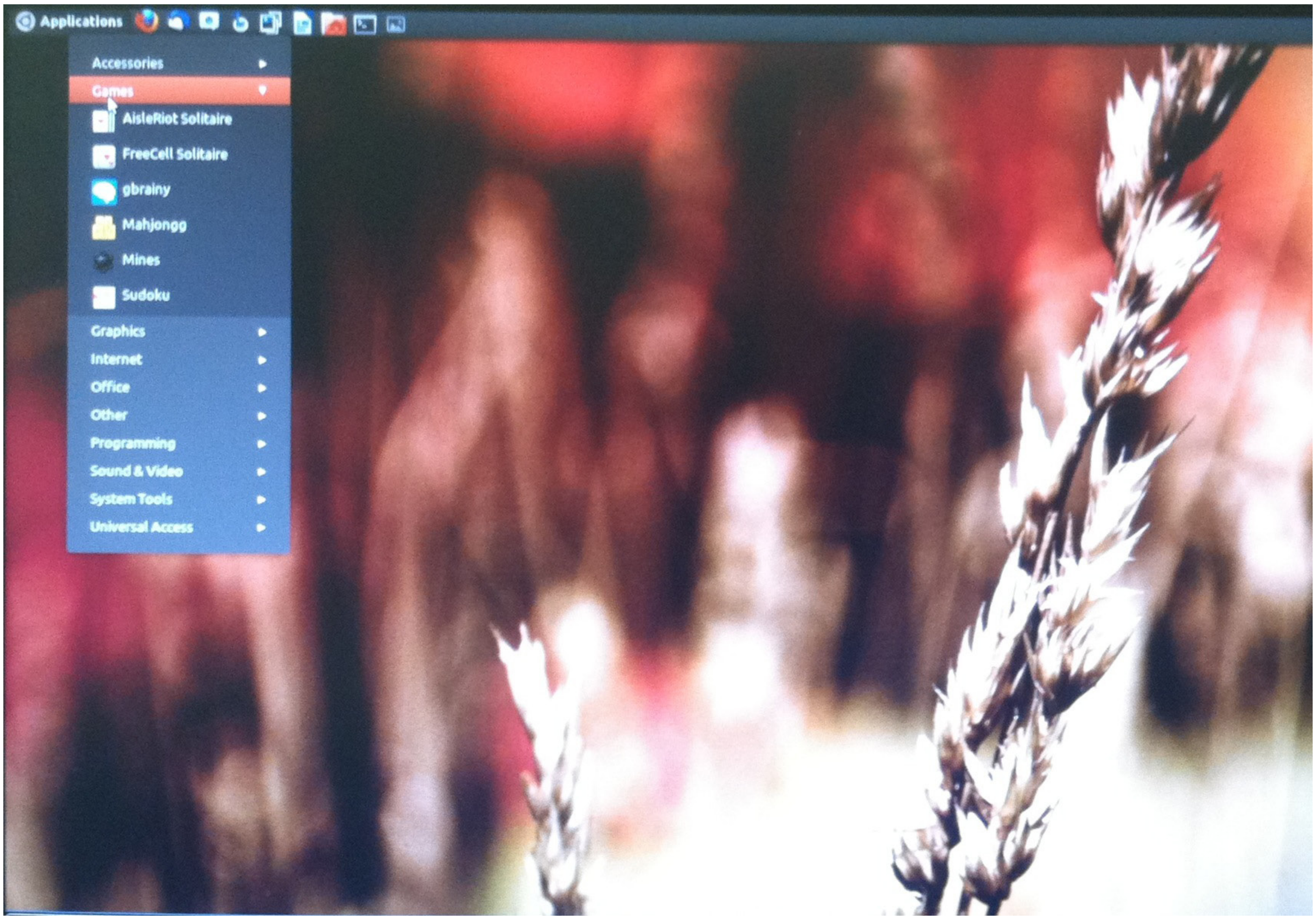


Figure 6. Look Ma, no dock!

Software

Linux. 'Nuff said (Figure 5).

Seriously though, ZaReason is a company that sells Linux computers. When configuring your system, you get to choose from the most common distributions. By name, they offer all the 'buntus, Debian, Fedora, Linux Mint, no operating system and my favorite, "tell us what kind of Linux you want."

Earl, being a *Linux Journal* reader himself, knew my feelings regarding Ubuntu's Unity interface. For the review unit, ZaReason installed Ubuntu 11.10, but instead of the default Unity interface, they installed GNOME 3 with extensions to look much like my beloved GNOME 2 (Figure 6). You may think the review unit got special treatment, but Earl actually told me this interface is what they ship to many customers, especially those uncomfortable with Unity.

We've Secretly Replaced Our Benchmarking Tools with Cryptocurrency!

It's easy to look up benchmarks on specific CPUs and GPUs. Because ZaReason sent me this unit after reading my article on cryptocurrency, it seems only fair to benchmark it using that method. Before I delve into hashrates and profits/losses, however, let me define a few terms for those not familiar with cryptocurrencies:

- **Mining:** cryptocurrency is something I've written about in past months, but basically, it's a form of virtual currency. That currency is created and secured by a large network of folks donating processing power to cryptographically verify transactions. In return for that donated processing power (aka "mining"), virtual coinage is produced and distributed to the miners. This both introduces the currency fairly into the economy and gives people an incentive for donating their processing power.

- **Hashrate:** this is the speed at which

JT01 Intel® Atom™ Fanless System with Ubuntu
Fully assembled, plug-and-play system with HDD, RAM, Wi-fi, and OS.
Low-profile, power efficient, and quiet.
Assembled & tested Ubuntu Linux compatible systems...

Genuine expertise. LOGIC SUPPLY

www.logicsupply.com/linux

© 2012 Logic Supply, Inc. All products and company names listed are trademarks or trade names of their respective companies.

The second-generation i7 processor, with its new AVX instruction set, is the fastest Litecoin miner I've ever seen.

miners can verify transactions, or more precisely, the speed at which they can solve the math problems that verify transactions. The faster the processor, the higher the hashrate.

- **Bitcoin:** the most widely accepted and traded cryptocurrency. The algorithm used to verify transactions is SHA256 and is most efficiently mined with GPUs, specifically AMD/ATI-brand GPUs.

- **Litecoin:** created to be the silver to Bitcoin's gold, Litecoin uses the Scrypt algorithm with the intent of being more efficient to mine with a CPU. Scrypt is more memory-intensive than SHA256, so CPUs have an easier time verifying the transactions.

Another important thing to understand about cryptocurrency is that the speculation market for individual currencies is extremely volatile. Bitcoins have traded for as little as fractions of a penny each, all the way to more than $30 each. Litecoins peaked at about a nickel, and at the time of this writing, they are selling for about a half-cent each. Cryptocurrencies are fun to play with, but I certainly don't recommend investing more than you can afford to lose.

Torture Testing, for Profit!

Let me start with the graphics card. First, the AMD 6970 is an incredibly fast gaming card. I know that's not what I'm reviewing, but it's worth noting I couldn't get this thing to drop a frame no matter how hard I tried.

When it comes to Bitcoin mining, the 6970 is no slouch either. Running at stock clockrates and voltages, I was able to mine Bitcoins at around 375 million hashes per second, or MH/s. Using my Kill-A-Watt meter, I measured the difference in wattage at about 225 watts versus the idle GPU. Using the profitability calculator at **http://allchains.info**, given the current Bitcoin difficulty and trading price, it turns out I can mine about $1.10 worth of Bitcoins every day. Keep in mind, however, that my electricity costs about $0.11 per kilowatt hour, so while I might make $1.10 in Bitcoins, it costs me $0.59 in electricity. At current rates (end of March 2012), that means this computer will make about $0.51 per day in profits, along with quite a bit of noise and heat.

The second-generation i7 processor, with its new AVX instruction set, is the fastest Litecoin miner I've ever seen. Running all six cores (12 with hyperthreading), the Core i7-3930K uses right around 200 watts according to my Kill-A-Watt. That's actually a little lower than some benchmarks I've seen, but nonetheless, it's what I recorded. The impressive part is that the CPU was able to sustain about 72 thousand hashes per second (KH/s), which is more than twice as fast as my AMD 6-core CPU can muster. Using the same profitability calculator from allchains.info, this CPU can mine around 42 Litecoins a day. Since the price of Litecoins is so low, that equates to only around $0.23. Unfortunately, at 200 watts, it costs around $0.53 per day to mine, so if I were to mine Litecoins today, I'd lose around $0.30 a day.

That is where the speculation comes in. Just a few short weeks ago, Litecoins were selling for 3–4 cents each, which means it was profitable to mine. Depending on your cost for electricity, perhaps it never will be profitable to mine Bitcoins or Litecoins. Again, I stress not to spend more money than you can afford to lose. Still, if you're into cryptocurrencies, it's a neat way to stress-test a computer!

Conclusion

The Valta X79 is a screaming-fast computer. To be honest, if I were going to build a powerhouse computer from scratch, it's the exact type of system I would build. I think that's what I like best about it. ZaReason has taken the essence of building your own computer, using standard parts, and done the testing to make sure everything works well together. Then ZaReason assembles it, configures it and, most important, supports it. If you've ever wanted to build a custom system, but wished you didn't have to give up a warranty and tech support, you'll love this Linux beast from ZaReason. The price for the unit as reviewed is $2,396. To configure and price your own custom Valta X79, head over to **http://www.zareason.com**. ■

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.

Resources

ZaReason: **http://www.zareason.com**

Bitcoin: **http://www.bitcoin.org**

Litecoin: **http://www.litecoin.org**

Market Price and Profit Calculator: **http://www.allchains.info**

Object-Oriented Programming with Lua

In spite of being a multiparadigm programming language, Lua provides a nontrivial object-oriented programming model that may prove cumbersome to programmers who want to apply OO software engineering practices to development. This article presents a reusable mechanism through which you can implement an object-oriented model using Lua's built-in constructs.

ALEJANDRO SEGOVIA

Lua is a dynamic scripting language and has been developed at the PUC-Rio University in Brazil since 1993. In its current 5.2 version, it is lightweight and portable, and it has become very successful for both open-source and commercial software projects, particularly in the field of video games, where it is being used by companies like EA.

Lua also is becoming increasingly important for mobile device software development and recently has been chosen by Wikipedia as its new scripting language. I first became interested in Lua when I learned about how easy it was to embed into C and C++ applications.

But, enough introduction—let's start creating some objects!

A Basic Object-Oriented Model

The first thing you should know is that Lua does not provide any special constructs for declaring object classes. Indeed, all Lua knows about are "tables". Programming in Lua in an object-oriented fashion boils down to being able to create an object model on top of Lua tables.

Fear not, however. Here I walk-though a model that can be used (hopefully) for all your object-oriented needs. This model is based on the one presented in the book *Programming in Lua*, which I highly recommend reading.

Let's start by supposing you want to create a mechanism through which you can represent 2-D Point objects. In Lua, you can define a generic Point table that looks like this:

```
Point = {x=0, y=0}
```

Think of this as the template for your point objects. All of your points will have x and y attributes.

Now, you can't do much with this construct as it stands. You need a way to "create" new points. In order to do so, let's add to the Point table a new attribute called, well, "new". It will allow you to create new points:

```
Point.new = function(o)
    if o then
        return o
    else
        return {x=0, y=0}
    end
end
```

The reason to add a branch in the code is that in case the caller doesn't supply any parameters, the function will just create a default point set to (0,0).

That wasn't very difficult! You now have a Constructor and can create new points like so:

```
p1 = Point.new({x=10, y=10})
```

However, for people coming from other programming languages, this function declaration might look a little strange and even may be hard to remember. Fortunately, Lua provides some syntactic sugar that allows you to rewrite this constructor in a more familiar fashion.

Take a look at the following snippet. The semantic value is exactly the same as before:

```
function Point.new(self, o)
        local p = o or {x=0, y=0}
        return p
end
```

It's much nicer, right? Let's stick to this style for the rest of this article.

And, that's it as far as creating or "instantiating" new points goes. Now, let's add a method to the Point class.

Let's provide the points the ability to pretty-print themselves to the standard output.

First, give Points a `print` method:

```
function Point.print(p)
    print("("..p.x..","..p.y..")")
end
```

Now, you can print points like so:

```
p1 = Point.new({x=10, y=10})
Point.print(p1) --prints: "(10,10)"
```

In a similar fashion, you can add a range of methods like translating, scaling and rotating points. The only limitation is that you always will have to call these methods as shown above: prefixing every invocation with the class name and explicitly supplying the point on which to operate.

It is natural to ask, "can you improve this syntax?" It turns out, you can by using Lua's colon operator (:).

The colon operator can be used in a statement like `p1:print()`, and it will be expanded to `p1.print(p1)` automatically. Just like there are two semantically equivalent options for adding methods to your classes, these two expressions are semantically equivalent as well.

Now, consider this: if you could have Lua associate whatever name is *at the left* of the colon operator to the object class, you would be able to simulate message passing to objects. This would allow using the following syntax with your objects:

```
p1:print() --prints p1
p2:print() --prints p2
```

Thus far, however, you can't associate p1 and p2 with Point. When `p1:print()` is expanded to `p1.print(p1)`, the Lua interpreter will print an error message stating that `"p1" has no attribute called "print"`.

Who knows about "print"? Well, "Point" does. What you need to do is tell Lua that when it fails to find a given attribute or method in "p1", it has to continue searching in "Point". This association can be declared using the concept of metatables.

Assuming "p1" is a point like before, this snippet will set everything up so the Lua interpreter continues searching for attributes and methods missing in "p1" in your Point class:

```
setmetatable(p1, Point)
Point.__index = Point
```

After this little change, the following code will start to behave the way you want:

```
p1.print(p1) --prints p1
```

But more important, its semantically

Although Lua doesn't provide an out-of-the-box object model, it turns out that implementing Polymorphism in Lua is very easy due to the dynamic nature of the language.

equivalent sibling will work too:

```
p1:print() --prints p1
```

What sadly will not work is trying to do this on "p2", because "p2" has no metatable. Therefore, you must set its metatable to "Point" as well. And, not just for "p2", but also for every single Point instance you create.

Although manually setting the metatable for every Point instance definitely is an option, it might prove to be cumbersome and error-prone. Furthermore, because this is an operation that has to be performed *on every* point instance, why not do it in the Point constructor? Let's do that.

This is the updated Point constructor. It first will check whether an object has been supplied and will create a default one if it hasn't. Then, it will set the object's metatable to be Point, making unrecognized messages be rerouted to the Point class for every point created:

```
function Point.new(self, o)
    local p = o or {x=0, y=0}
    setmetatable(p, self)
    self.__index = self
    return p
end
```

Note: I am passing the metatable as the first argument of the Constructor (and calling it "self"). This not only will allow you to create new points using the colon operator (as shown in the next example), but it also lets you do some advanced object-oriented tricks that I'll discuss later.

With this updated constructor, the following is now valid:

```
p1 = Point:new({x=10, y=10})
p2 = Point:new({x=20, y=20})
```

That's it as far as declaring, instantiating and using point objects goes. Try running this example using the Lua interpreter, and make sure you understand the concepts. I will build upon them in the next sections to show how to implement Polymorphism and Inheritance.

Polymorphism

In the previous section, I described a mechanism by which you can model a class "Point" by means of Lua tables. In this section, let's build on the previous concepts to support Polymorphism with your objects.

Polymorphism is one of the most powerful concepts in object-oriented programming. It allows the programmer to change the behavior of a system dynamically based on the type of the objects to which you send your messages.

Although Lua doesn't provide an out-of-the-box object model, it turns out that implementing Polymorphism in Lua is very easy due to the dynamic nature of the language.

Suppose you have a global function "printpoints()", which receives a list of points and has them printed to the standard output. Furthermore, suppose you had two different point types in your system: Point, which models 2-D points just like you have been using so far, and Point3D, a new class that represents 3-D points that live in 3-D space.

For this example, let's imagine that the "printpoints()" function will be required to handle mixed lists of 2-D and 3-D objects.

Well, if you can have both point types present in a single list and you have a statically typed programming background, you already might be thinking that you need both points to ascribe to a common interface that allows them to be printed. The term "IPrintable" may even come to mind.

Indeed, what you require is for both points to share a set of messages they respond to. In other words, you need both types to agree to a contract that "certifies" that they can be "printed".

In statically typed programming languages, this contract can be expressed by means of Interfaces. In Lua, however, such a construct is not necessary, as you can use the dynamic foundation of the language to ask, at runtime, whether an object responds to any given message.

In this case, the message would be "print".

This concept of objects conforming to a common Interface that is not declared anywhere explicitly is known as an Implicit Protocol. Both point types will abide to the Implicit Protocol of knowing how to print themselves.

Let's see how this "printpoints()" function could go:

```
function printpoints(points)
    for _, p in ipairs(points) do
        if p.print then
            p:print()
        end
    end
end
```

Here, you receive the list of points to print, and independently of what the list contains, iterate over its elements. For each element, you ask whether it responds to the message "print". If it does, you send it the "print" message.

Let's use the interactive interpreter

Unlike Polymorphism, when modeling type Inheritance, it is not enough to have the objects ascribe to the same Implicit Protocol and rely on the dynamism of Lua.

to see how this function behaves when handling a list consisting of two 2-D Points and one 3-D Point:

```
$ lua
Lua 5.1.4  Copyright (C) 1994-2008 Lua.org, PUC-Rio
> require "point"
> points = { Point:new({x=10,y=10}),
Point:new({x=20,y=20}),
 ➥Point3D:new({x=30,y=30,z=1}) }


--The "points" list contains 2D and 3D points.


> printpoints(points)
(10,10)
(20,20)
(30,30,1) -- This is a 3D point!
```

Note how the points in the list were able to respond to the "print" message, each using its corresponding behavior, meaning 2-D points printed their (x,y) pairs, whereas the single 3-D point correctly printed its (x,y,z) tuple. This is, effectively, Polymorphism in Lua.

If you are feeling uneasy about the concept of Interfaces, remember that the true meaning of Polymorphism is not in implementing Interfaces, but rather in having objects dynamically respond to the messages received according to their type. Dynamically typed programming languages are, therefore, an ideal fit for Polymorphism.

Inheritance

The last major concept common to object-oriented programming that I'm going to tackle here is Inheritance. How can you add an Inheritance mechanism to the object-oriented model I've been discussing?

Unlike Polymorphism, when modeling type Inheritance, it is not enough to have the objects ascribe to the same Implicit Protocol and rely on the dynamism of Lua. In order to implement Inheritance, you need to describe a relationship effectively between the types of the base data type (the base class) and the derived data type (the derived class).

What you need to do is be able to "extend" an object type with a newer, "more concrete" type that adds specific logic.

The way to achieve this using Lua

might seem strange at first, but if you bear with me, you will see it naturally matches the concepts used here thus far.

To continue with this example, let's add to the system the ability to represent 3-D points in homogeneous coordinates. In homogeneous coordinates, each 3-D point is represented using four values (x,y,z,w).

When converting from Cartesian coordinates to homogeneous coordinates, you just need to set the w value to 1. To convert from homogeneous coordinates back to Cartesian coordinates, you need to divide all components of the point by the w value, therefore taking (x,y,z) back to Cartesian coordinates.

The 3-D point (1,1,1) in homogeneous coordinates would be (1,1,1,1), as well as (2,2,2,2), (3,3,3,3) and so on. If you are not convinced, try dividing by w in each case and see what happens.

Points in 4-D homogeneous coordinates *are* 3-D points, meaning, Points in homogeneous coordinates (PointH) have an "is a" relationship with class Point3D. This is an Inheritance relationship you can represent in the object model.

Assuming class Point3D exists and is similar to Point, you start by stating that PointH (4-D homogeneous coordinate points) are 3-D Points:

```
PointH = Point3D:new({x=0, y=0, z=0, w=1})
```

What I did here, conceptually, was to declare PointH to be a new class that "Inherits" all the behavior from Point3D. In particular, its metatable will point to Point3D.

What's interesting is the fact that when you create a new PointH instance called "ph", by using the PointH:new() method, Point3D's constructor will be called, but with the "self" object pointing to (table) PointH instead. Therefore, the "ph" instance will have its metatable pointing to PointH and not Point3D.

This achieves a "chain" of metatables. PointH instances will have PointH as their metatable, whereas class PointH will have Point3D as its metatable. This chain of metatables allows the Lua interpreter to conduct the following searches automatically when a message is sent to a PointH instance:

- 1) First, search in the instance itself.
- 1.1) If the method is present, call it.
- 1.2) Otherwise, search in the PointH class.
- 1.2.1) If the method is present, call it.
- 1.2.2) Otherwise, search Point3D class.
- 1.2.2.1) If present, call it.
- 1.2.2.2) Otherwise, fail.

In this example, if you send "ph" a "print" message, "ph" doesn't know "print", so it delegates the message to its metatable: PointH. PointH doesn't know how to "print" either, so it delegates the message to its own metatable: Point3D. Point3D knows how to "print", so its method is used.

Now, suppose you taught PointH how to interpret the "print" message by defining the function `PointH.print(p)`. In that case, PointH's print would have been used instead of continuing the search in Point3D.

What this means is that your Inheritance model effectively supports "overriding" a base class' methods.

This is how PointH's complete implementation would look:

```
PointH = Point3D:new({x=0, y=0, z=0, w=1})


function PointH.print(p)
    print("("..p.x..","..p.y..","..p.z..","..p.w..")")
end
```

Here, Point3D's print is overridden with PointH's, meaning that when calling print on a PointH instance, the w value will be printed along the (x,y,z) tuple.

Conclusion

This article described one of the mechanisms through which object-oriented programming can be implemented in the Lua programming language. Polymorphism and Inheritance, two of the main pillars upon which object-oriented programming is based, was discussed as well.

Hopefully, you will be able to use these ideas, or build on top of them, to bring your own object-oriented software designs to Lua.■

Alejandro Segovia Azapian is a computer engineer from Uruguay. He specializes in computer graphics, and he has conducted several undergraduate courses at the Uruguayan Catholic University on the topic. At the time of this writing, he is working as a software engineer for Autodesk, Inc.

Resources

Programming in Lua: **http://www.lua.org/pil/index.html**

Lua: **http://www.lua.org**

"Why 'Lua' is on everybody's lips, and when to expect MediaWiki 1.19": **http://en.wikipedia.org/wiki/Wikipedia:Wikipedia_Signpost/2012-01-30/Technology_report**

Lua (Programming Language)—Applications (Wikipedia): **http://en.wikipedia.org/wiki/Lua_%28programming_language%29#Applications**

Lua-Scripted Video Games (Wikipedia): **http://en.wikipedia.org/wiki/Category:Lua-scripted_video_games**

Android Programming with APP INVENTOR

Drag and drop your way to Android programming.

AMIT SAHA

MIT App Inventor, re-released as a beta service (as of March 5, 2012) by the MIT Center for Mobile Learning after taking over the project from Google, is a visual programming language for developing applications for the Android mobile computing platform. It is based on the concept of blocks, and applications are designed by fitting together blocks of code snippets. This may sound like a very childish way of programming, especially for seasoned readers of *Linux Journal*. But then again, App Inventor will tickle the child programmer in you and make you chuckle at the ease with which you can develop applications for your Android device. In this article, I describe how to use the camera on the Android device, develop e-mail and text-messaging-based applications and also show how to use location sensors to retrieve your current geographical location. Let's get started.

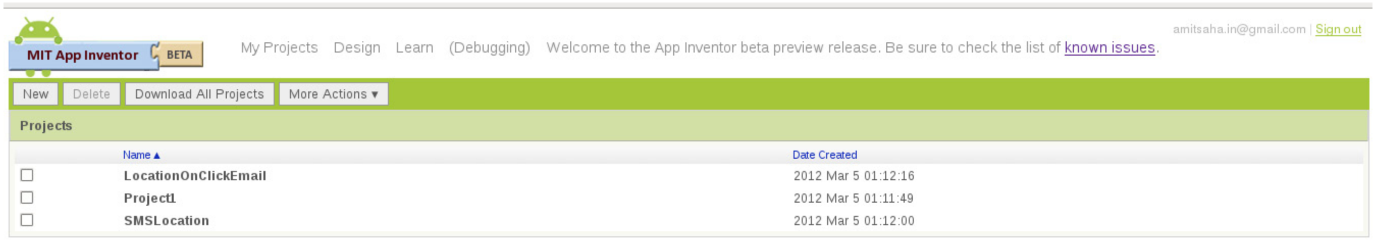
GETTING STARTED

App Inventor has minimum setup requirements and is completely browser-based. You need a working Java installation on your system, as it uses Java Web Start for its functioning. Point your browser to **http://appinventor.mit.edu**, and once you sign in with your Google account, you should see a screen as shown in Figure 1. This is called the Projects Page where you can see your existing projects and create new ones.

Now, let's develop and deploy an Android application using App Inventor and in the process learn the basic development-deployment cycle. Create a New Project using the New Project button, and enter a name for your project, say "Project1". Now you should see the Designer window for your project. The Designer window is composed of four sub-components. The Palette on the leftmost side of the window is the placeholder for all the available components for your project. The Viewer is where the application will be designed by placing together various components (this is where you design the user interface for your application). The Components show the currently used components in your project, and the Properties column is where you assign the properties of the components.

First, let me briefly explain the notion of components. An App Inventor project is made up of building blocks called components, such as a text label to display text, a text box to take user inputs, a camera component to click pictures and so on. Currently, you will see a few categories of components—basic components, such as those for user input and display of text to more specialized components, such as those for displaying media and animations, and components acting as an interface to the device sensors. A complete reference for all the components is available at **http://appinventor.mit.edu/learn/reference/index.html**. Components have associated behavior, methods and properties. Some of the properties can be set; whereas others can be only read.

In this first project, let's use the following components: Camera, Button and Image. The code usually shows it better, but briefly here is what you're going to do: clicking the button starts the camera on your device, which you use to click a picture, which then is



Welcome to the App Inventor beta preview release. Be sure to check the list of known issues.

Projects

	Name ▲	Date Created
☐	LocationOnClickEmail	2012 Mar 5 01:12:16
☐	Project1	2012 Mar 5 01:11:49
☐	SMSLocation	2012 Mar 5 01:12:00

Figure 1. App Inventor's Projects Page

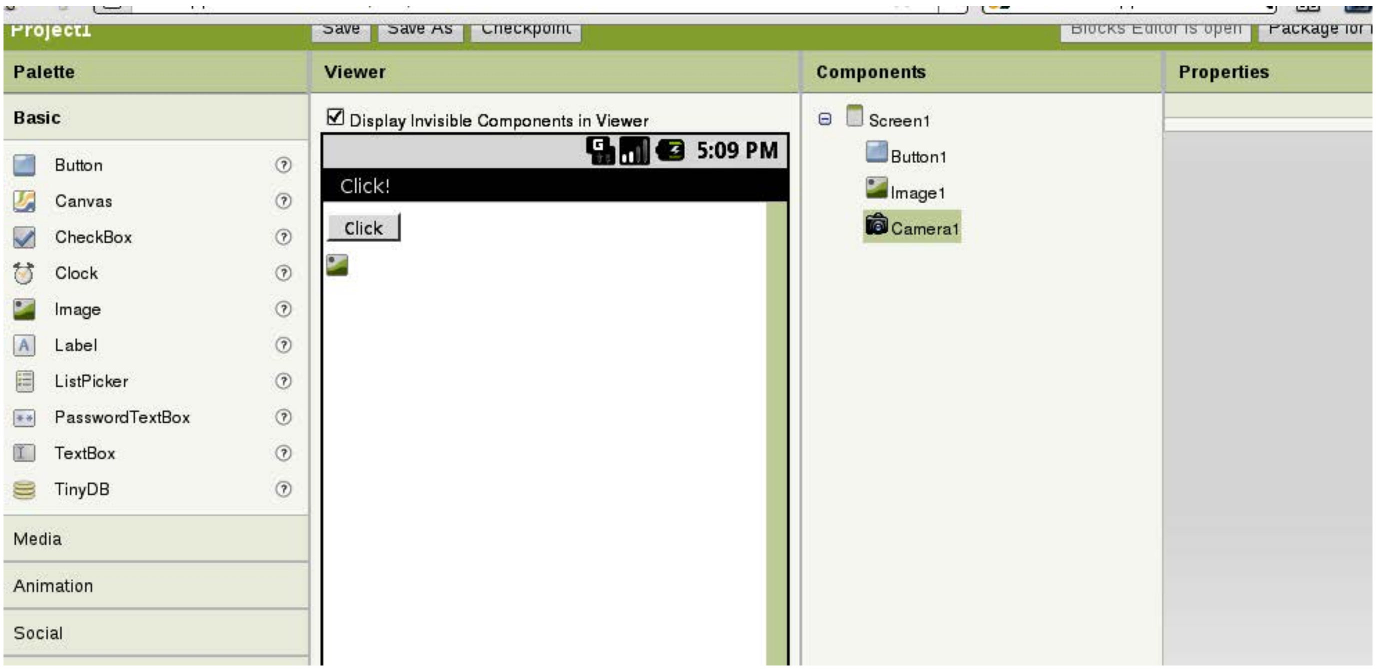


Figure 2. User Interface for Project1

displayed using the Image component. Here are the steps:

1. Drag a Camera component from the palette to the Viewer. It should show up under Non-visible components below the Viewer. By default, it will be named as Camera1, which you can, of course, change to something else.

2. Drag a Button to the Viewer, and from the Properties, change its Text to "Click".

3. Drag an Image component onto the Viewer.

4. You can play around with the Screen properties to set things like title, background color and orientation. For the purpose of this project, set the Title to "Click!".

That completes the design of the user interface (Figure 2). Next, let's program the components using Blocks.

Open the Blocks Editor, which should start downloading the JAR file for the editor. It will ask you for the location of the App Inventor setup commands if you have not installed them in the standard location under /usr/google. The Blocks Editor for the current project will look like Figure 3. Going back to the description for this project, the goal is to activate the device camera when the button is clicked. This is done with the code block "When Button1.click", which you dragged from the

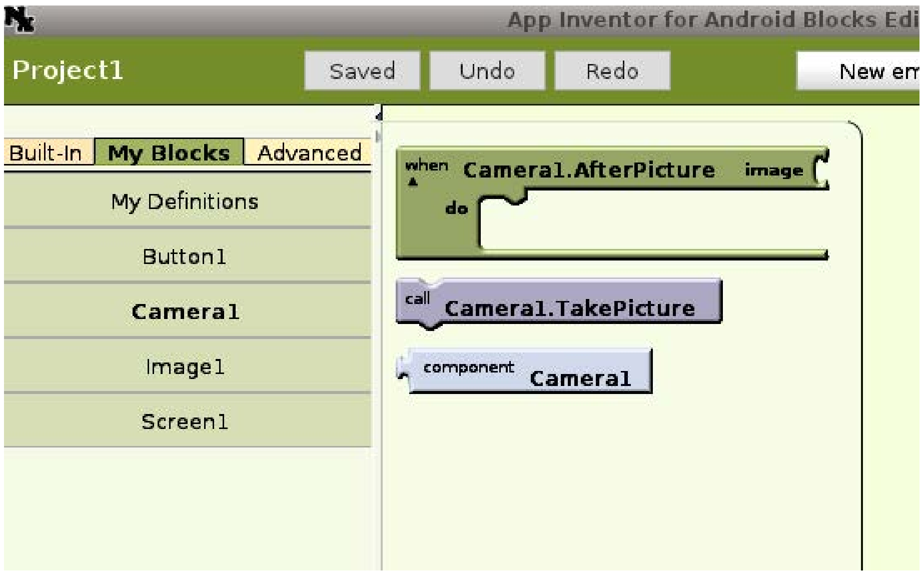


Figure 3. Blocks Editor for Project1

Blocks pane on the left. When the button is clicked, you want the device's camera to be activated, so drag the "call Camera1.TakePicture" block inside the previous block. Once the picture is taken, you will want it to be displayed using the Image component. So, insert the block "when Camera1.AfterPicture" into the editor, and then set the "Image1.Picture" to the location of the saved image.

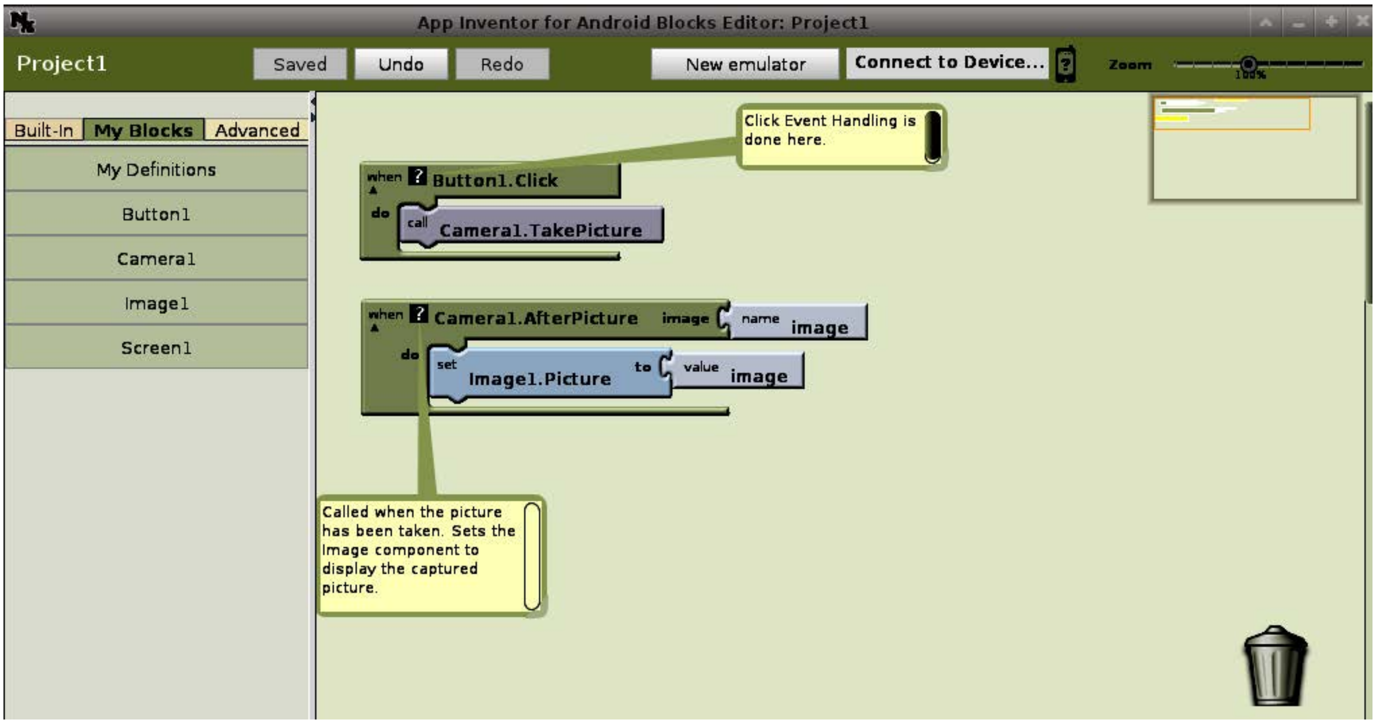
Now that you have designed the user interface and programmed the application's logic, you're ready to test it. Go back to the Designer window, and on the right, click on Package for Phone→Download to this Computer. That should initiate the download of the Android package (.apk file) for your project. Now, transfer this file to your Android device, and install it. Then, try it out.

A PEEK UNDER THE HOOD

Now you have designed and deployed your first Android application, and you have used components (the camera component and the image components), assigned them behavior and set properties. If you are familiar with the idea of event-driven programming, you already will have realized that App Inventor is an event-driven programming framework. The event can be the user clicking a button or the reception of a text message. For example, when the button is clicked, an event is said to have occurred, and in response to this event, the camera is activated. Again, when the camera finishes capturing a picture and saving it, the response code uses the image location to display it using an image component.

Earlier, I mentioned that components have associated behavior, methods and properties. You can find these for a component by clicking the component in the Blocks Editor. For example, Figure 4 shows the method available for the Camera component (Camera1.TakePicture) and the behavior (Camera1.AfterPicture).

Besides the blocks associated with components, more fundamental


App Inventor for Android Blocks Editor: Project1
Project1
Saved
Undo
Redo
New emulator
Connect to Device...
Zoom
Built-In
My Blocks
Advanced
My Definitions
Button1
Camera1
Image1
Screen1
when Button1.Click
do call Camera1.TakePicture
when Camera1.AfterPicture image name image
do set Image1.Picture to value image

Click Event Handling is done here.

Called when the picture has been taken. Sets the Image component to display the captured picture.

Figure 4. Blocks Available for the Camera Component

programming blocks are available: Math blocks, Logic blocks, Control blocks and others. (I'll demonstrate using a few of these in one of the projects later in this article.)

Now that you have a basic idea of developing applications using App Inventor, let's look under the hood a bit, starting from the source. Download the source code for "Project1" by going to the Projects Page and selecting Project1 and clicking on More Actions→Download Source. That should start downloading the sources in a zip file. When you unzip the file, you will have two directories: src and youngandroidproject. Under the src directory, you will have a subdirectory called appinventor, which houses the subdirectories, and then ai_droidery/Project1 (note that "droidery" is my Google user name). In this directory, you will see the source files Screen1.blk, Screen1.scm and Screen1.yail. Screen1.blk is an XML-based representation of the visual blocks that was created earlier; Screen1.yail is an intermediate language based on the Scheme language used by App Inventor, which is then fed to Kawa to create the Android package for installation on Android devices. The Screen1.scm file is a JSON representation of the components used in the project with details about the components, such as the version information. If you are keen to understand how App Inventor really works, you also may want to check out App Inventor's source code (see Resources).

SENSORS, TRUE TO THEIR NAMES, ARE THE EYES AND EARS OF YOUR ANDROID DEVICE. THEY ALLOW YOUR DEVICE TO SENSE THE WORLD AROUND IT.

Next, let's move on to becoming familiar with sensors and some of the other components available in App Inventor.

SENSING THE WORLD USING SENSORS

Sensors, true to their names, are the eyes and ears of your Android device. They allow your device to sense the world around it. For example, the location sensor on your device keeps track of your current location information using your mobile and Wi-Fi signal information and GPS data. Other sensors on your Android device include proximity sensors and motion sensors. In this section, let's use the location sensor on your Android device to write two simple applications that can be used on their own or as a starting point for something more useful

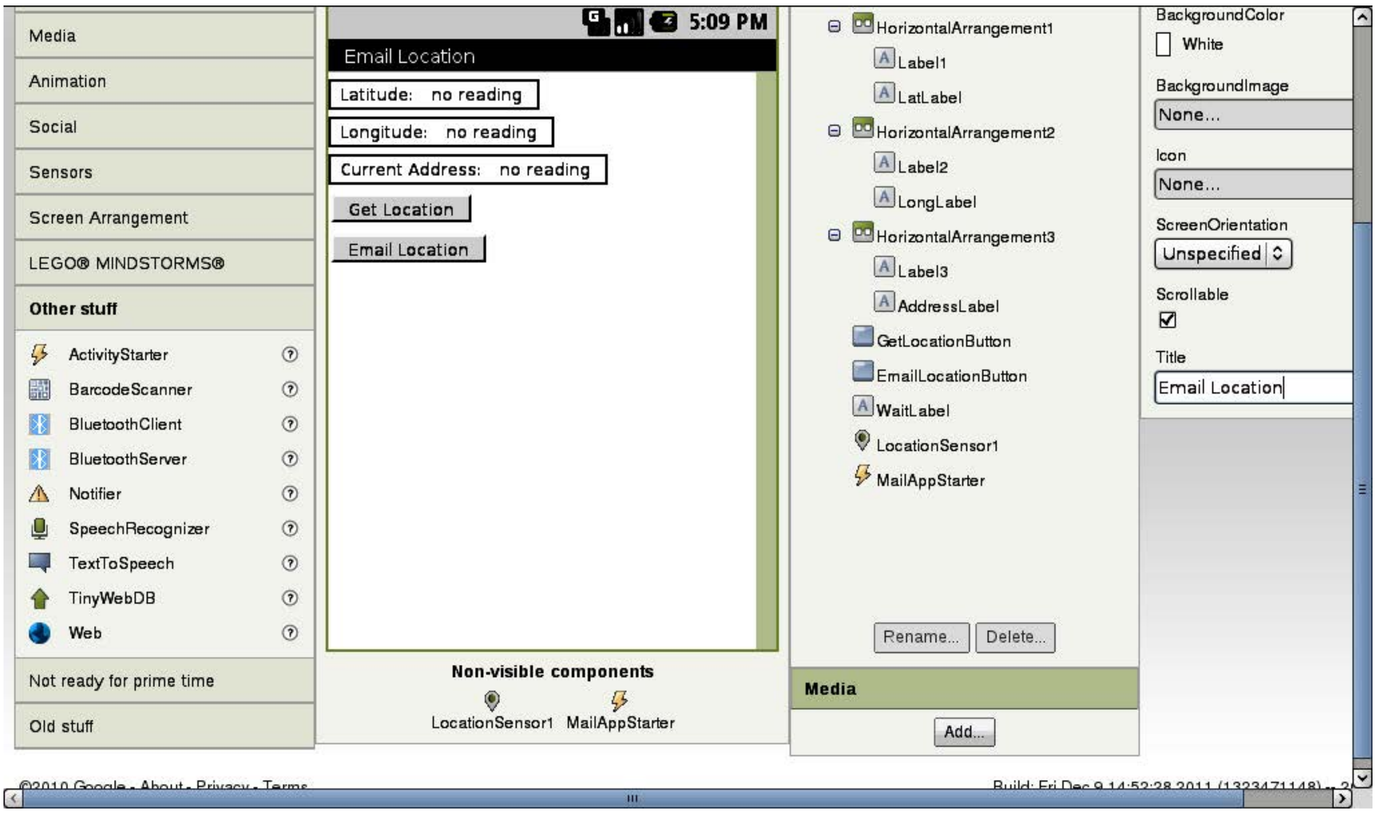
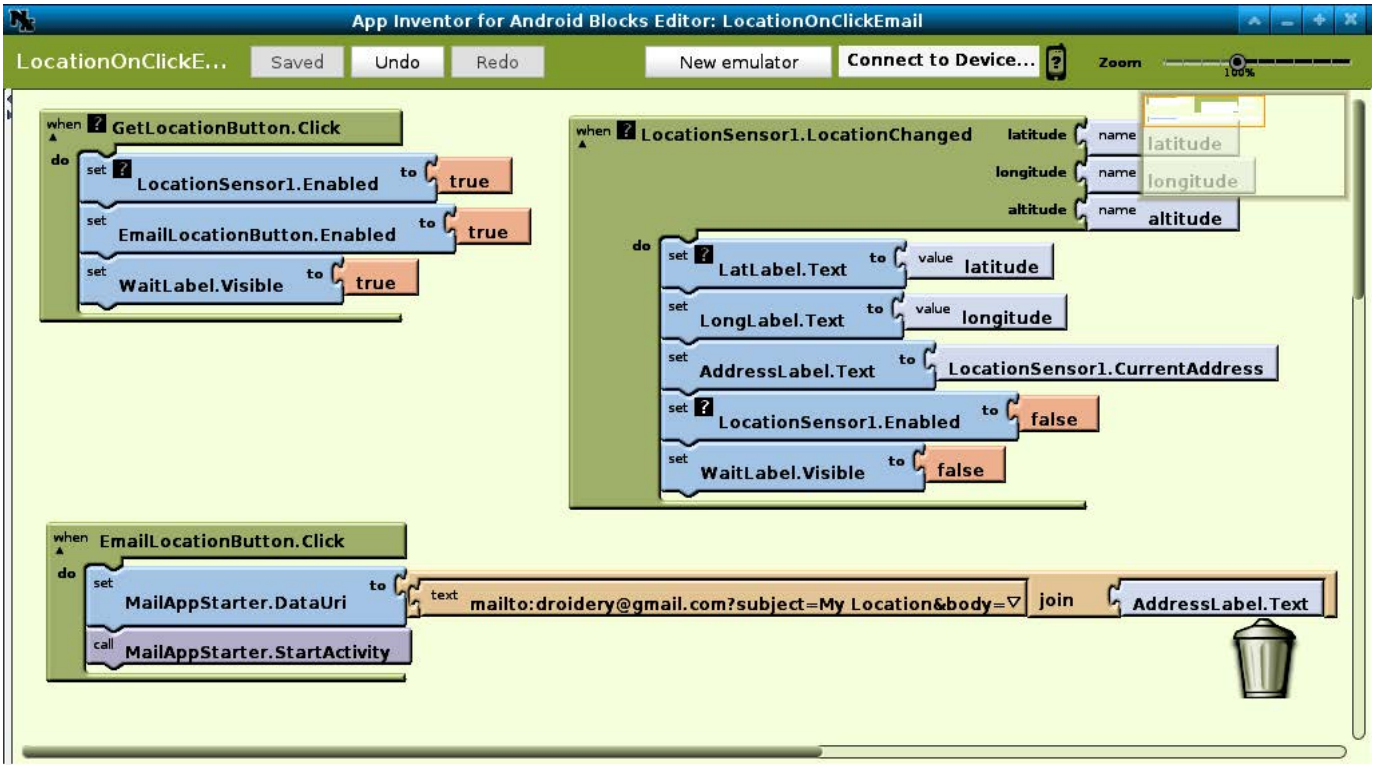


Figure 5. User Interface for the LocationOnClickEmail Project



```
when GetLocationButton.Click
do
  set LocationSensor1.Enabled to true
  set EmailLocationButton.Enabled to true
  set WaitLabel.Visible to true

when LocationSensor1.LocationChanged  latitude name latitude
                                      longitude name longitude
                                      altitude name altitude
do
  set LatLabel.Text to value latitude
  set LongLabel.Text to value longitude
  set AddressLabel.Text to LocationSensor1.CurrentAddress
  set LocationSensor1.Enabled to false
  set WaitLabel.Visible to false

when EmailLocationButton.Click
do
  set MailAppStarter.DataUri to text mailto:droidery@gmail.com?subject=My Location&body= join AddressLabel.Text
  call MailAppStarter.StartActivity
```

Figure 6. Final Blocks for the LocationOnClickEmail Project

and customized. In the process, you'll learn to make use of a couple more App Inventor components.

E-MAIL YOUR CURRENT LOCATION

Consider a not-so-fictional scenario when you might want to tell your friend exactly where you are at the moment so that she can drive down to meet you. Or, you simply may be lost. Either way, the Location Sensor can help. Let's call this project "LocationOnClickEmail". The user interface for this project looks like the one shown in Figure 5. Besides the basic components, such as text labels and buttons, add the LocationSensor component (found under the Sensors category) and an ActivityStarter component (found under the Other Stuff category). The ActivityStarter component, which has been named "MailAppStarter" will be used to start the e-mail application on the Android device. For details on the ActivityStarter Component, refer to **http://appinventor.mit.edu/learn/reference/other/activitystarter.html**.

Now you need to add the project logic using the Blocks Editor as before. Figure 6 shows the final state of the Blocks Editor for this project.

The application logic can be divided into two steps—obtaining the location using the Location Sensor when the Get Location button is clicked. This is done in the "when GetLocationButton.Click" code block. When this button is clicked, the Location Sensor is enabled. Once the Location Sensor has been able to obtain the location information, it invokes the "when LocationSensor1.LocationChanged" method where the text labels are updated with the location data. Next, when the Email Location button is clicked, the MailAppStarter component's DataUri property is set to start the mailing application. Here, the recipient is set to "droidery@gmail.com", the subject to "My Location" and the body of the message to the obtained address. The recipient and subject can be changed in the mailer application on the device.

That completes the current project. For more details on using the Location Sensor and the App Starter components, refer to the App Inventor Reference (see Resources).

TEXT-MESSAGING-BASED LOCATION SENSOR APPLICATION

In the last application, you initiated the location sending event. What if you want to design an application that will run as a service, such that when it receives a request via text message, it sends your current location to the sender? Even with privacy being such a sensitive issue in today's connected world, such an application can be useful if you want to make sure your not-so-grown-up kid isn't lost, for example. In addition to the Location Sensor component, you will become familiar with the Texting component in this application.

Here is the idea: on receipt of a text message with "location" in its body, the application replies with the current

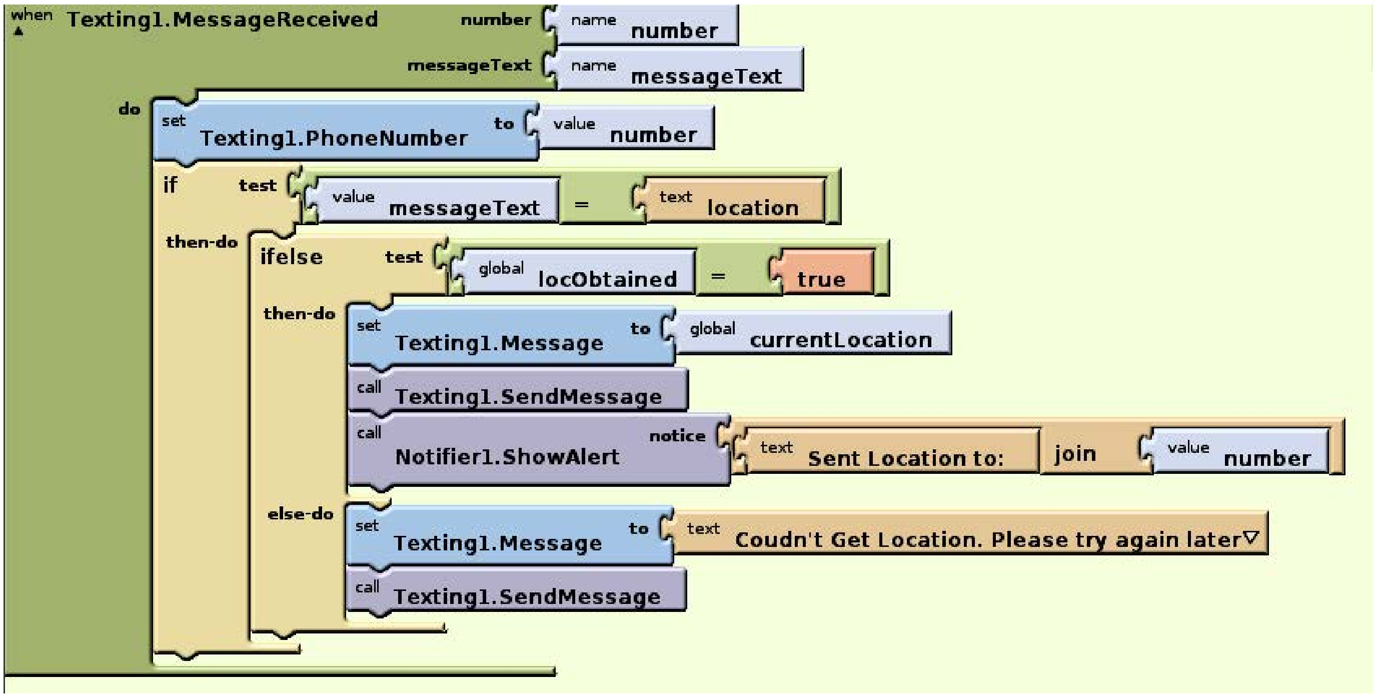

```
when Texting1.MessageReceived   number      name number
                                messageText name messageText
  do  set Texting1.PhoneNumber to value number
      if test  value messageText = text location
      then-do
        ifelse test  global locObtained = true
        then-do
          set Texting1.Message to global currentLocation
          call Texting1.SendMessage
          call Notifier1.ShowAlert notice  text Sent Location to: join value number
        else-do
          set Texting1.Message to text Coudn't Get Location. Please try again later
          call Texting1.SendMessage
```

Figure 7. Action Taken When a Text Message Is Received

location as a text message. The actions taken upon receipt of a text message are shown in Figure 7. This is the core logic for the application. In the "Texting1.MessageRecieved" procedure, the "number" and "messageText" are available as arguments. If the "messageText" is "location", then check whether the location has been obtained. If yes, then construct a reply using the address and send the text; otherwise, send an error message back as the reply.

The complete application, along with others, can be downloaded from **https://bitbucket.org/amitksaha/articles_code/src**. You can upload the source (.zip) archives to App Inventor directly and try out the applications after packaging them. In this article, I have strictly concentrated on using an Android device for testing the applications. For basic uses, you also can use the emulator that is available in App Inventor and also use a live development methodology where you can install the application directly to your device. See the App Inventor Web site to try these out. I tested these applications on my Samsung Galaxy-SII running Android 2.3, but I hope there won't be any issues with running them on other devices running Android 2.2 and higher.

LOOKING AHEAD

I started this article with the intention of having some fun programming for the Android platform, and I hope it has been so thus far. If you're interested in looking into App Inventor further, the first things that you might want to check out, apart from extending the projects to something more fun and useful, are the various other components. Of special note is the Data Store component that allows you to store data on the device, the Web components for interacting with remote Web content, other Sensor components and Media components.

App Inventor is fun, but you might feel that although it's good as a starting point, you would prefer a more traditional programming language as you become more familiar with Android development. Instead of completely throwing your App Inventor project away, consider using the App Inventor Java Bridge to use your App Inventor components while you write Android applications using the more traditional way of programming in Java.

If you feel the need to run your own App Inventor service, the MIT Center for Mobile Learning has made available the App Inventor JARs to enable you to host your own service (see Resources).

If you want to keep exploring App Inventor itself, two excellent books are available: David Wolber, Hal Abelson, Ellen Spertus and Liz Looney's *App Inventor: Create your own Android apps* (O'Reilly) and Jason Tyler's *App Inventor for Android: Build Your Own*

Apps—No Experience Required! (Wiley).

If you enjoyed App Inventor, you might want to look at some other tools for programming your Android device visually, such as DroidDraw and Corona. And if you want to program Android visually on the device itself, check out Catroid.■

Amit Saha is currently a PhD research student in the area of Evolutionary Algorithms and Optimization. Like his random echoes show (http://echorand.me), he has been writing on myriad Linux and open-source technologies for the past five years. Overall, he loves playing around with a bit of this and a bit of that. He welcomes comments on this article and beyond at amitsaha.in@gmail.com.

Resources

MIT App Inventor: **http://appinventor.mit.edu**

Setting Up Your Computer: **http://appinventor.mit.edu/learn/setup/index.html**

App Inventor Reference: **http://appinventor.mit.edu/learn/reference/index.html**

APK File Format: **http://en.wikipedia.org/wiki/APK_%28file_format%29**

Under the Hood of App Inventor:
http://googleresearch.blogspot.com/2009/08/under-hood-of-app-inventor-for-android.html

App Inventor Open-Source Project: **http://code.google.com/p/app-inventor-releases**

Activity Starter Component: **http://appinventor.mit.edu/learn/reference/other/activitystarter.html**

App Inventor Java Bridge Project: **http://groups.google.com/group/app-inventor-instructors/browse_thread/thread/10a64e64b7886afb**

Running Your Own App Inventor Service: **http://appinventoredu.mit.edu/developers-blogs/andrew/2011/nov/running-your-own-app-inventor-service**

App Inventor Course: **http://sites.google.com/site/appinventorcourse**

"Android App Development" presentation by Peter McNeil: **http://www.cjugaustralia.org/September+2011**

Catroid Project: **http://code.google.com/p/catroid**

Code for This Article (available in the appinventor_article subdirectory):
https://bitbucket.org/amitksaha/articles_code/src

A PURE DATA MISCELLANY

An introduction to Pd—a graphic patching environment designed for audio production and processing, with extensibility for the addition of other media-related features.

DAVE PHILLIPS

After reviewing recent trends in the world of SuperCollider3 for LinuxJournal.com, I decided to investigate activity in the world of Pure Data, better known to its users and abusers as Pd. However, instead of offering up yet another tutorial on this or that aspect of Pd, I've opted for a nonlinear account of my experience with the system.

WHAT IS PURE DATA?

Pd often is described as an example of a "visual programming language", a definition that is at least broad enough to avoid inaccuracy. Personally, I define Pd as a graphic patching environment designed for audio production and processing, with extensibility for the addition of other media-related features, such as tools for processing images, text and video. Pd's graphic patching capabilities consist of a small collection of basic units—objects, messages, numbers, symbols and comments—that are placed on a canvas-like work area and graphically connected (patched) into signal processing networks (Figure 1), a process similar to constructing a sound-producing arrangement of modules on an old-school modular patching synthesizer. Of course, Pd is a little more modern and a lot more flexible. Its basic units are not predefined synthesis or audio processing primitives. They can be those things (and much more), but you need to define them as such, and you need to know what definitions and values are allowable for each type of unit. Figure 1 illustrates

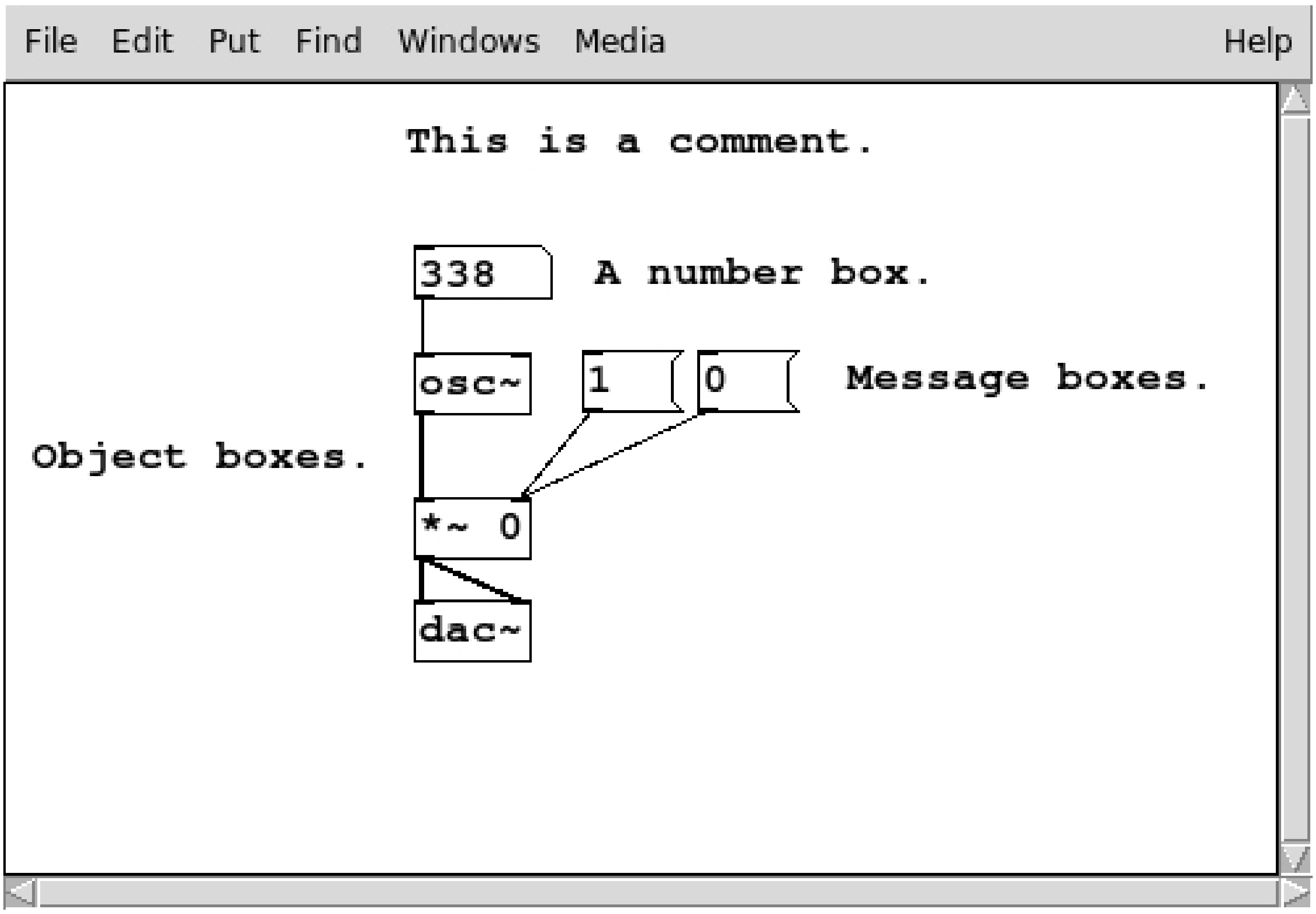


Figure 1. A Simple Pd Patch

a small Pd patch with four basic units and some typical values in place.

This patch creates the sound of a sine wave at 338Hz. The waveform and its amplitude are defaults for the osc~ object. The frequency value is supplied by the number box. The dac~ audio output object receives sound data from the osc~ object and can be switched on or off by clicking on the message boxes. The GUI clearly indicates the connections between the patch's components and the signal's path through them. The patch is complete, but no amplitude envelope is present. When the Compute Audio button is checked, the patch immediately will play at the default amplitude and continue until switched off (the message box with the 0 value).

The example is here merely to present a picture of Pd's GUI. I could go on to add an envelope generator and an on/off switch, and I could proceed to inform my readers about details like hot and cold inlets, the order of the processing graph, how to add GUI elements and myriad other things you can learn by studying Pd's extensive documentation.

Incidentally, many users hear about Pd with some reference to the famous Max/MSP program commercially available from Cycling 74. The reference is understandable—Miller Puckette, Pd's author, is also the author of the original Max/MSP. I'm not sure why he would create an open-source alternative to his (very successful) commercial software, but I'm comfortable with his decision. In fact, Max/MSP and Pd are not identical programs. The developers at Cycling 74 have created an environment related to and similar to Pd in many ways, but it's not a for-sale version of Pd, and Pd is not a free version of Max/MSP.

THE FLAVORS MENU

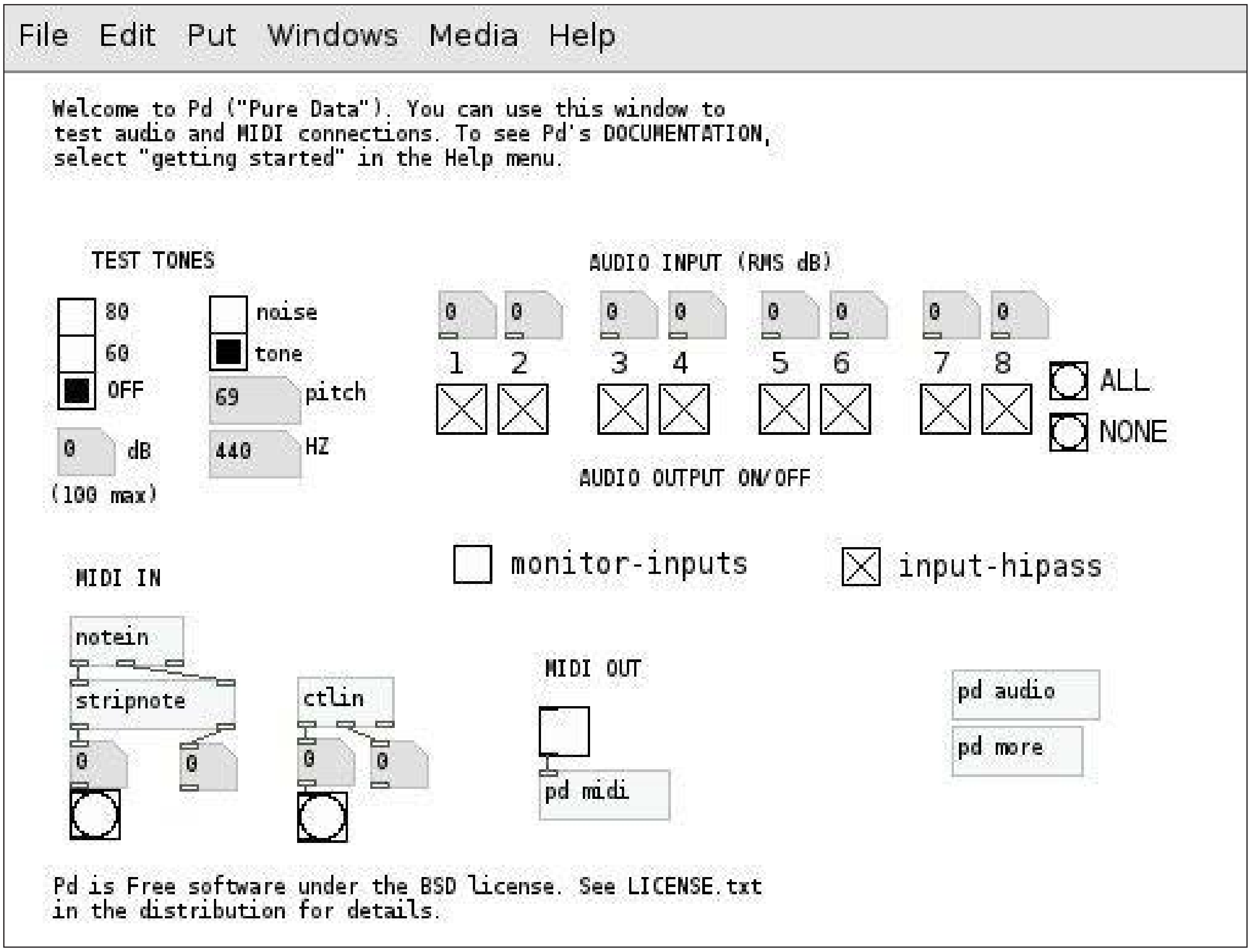
Pd comes in a variety of flavors, all based on the "vanilla" Pd release maintained by Miller Puckette. The other flavors differ primarily in their packaged extensions and their GUI enhancements, but some deeper changes may be made as well, typically to accommodate performance requirements. Notable extensions to Pd include GEM (OpenGL graphics), iemlib (library of GUI, math, filters and other DSP code), zexy (a package of "objects that provide functionality missing in plain Pd") and the PDP/PiDiP/GridFlow packages (more tools for manipulating images and video). Incidentally, the original vanilla GUI is built on Tk, a widget set commonly associated with the Tcl scripting language. The unmodified interface is certainly usable, but both the Pd-extended and Pd-L2Ork Projects have contributed improvements to its appearance and usability.

The package repositories for many mainstream distributions include packages for Pd in either the vanilla or Pd-extended flavors. The Pd-L2Ork Project maintains packages for various Linux distributions, and of course, all versions of Pd are available in source code packages. If you want to try Pd in an optimized risk-free environment, check out the Puredyne distribution. Puredyne is a live Linux system designed for multimedia artists and musicians who want a fast, light system that's also loaded with some of the best audio/video programs available in the free software universe.

For my re-introduction to Pd, I decided to use the packages available from the L2Ork site maintained by Professor Ivica Bukvic at Virginia Tech. This flavor is based on the Pd-extended package, with customizations for the hardware used by the L2Ork performance group. Pd-L2Ork includes almost all the extra packages I needed for this article. The Tech group's flavor works well, but I encountered some significant problems when I tried to run examples from those extra libraries (see below).

GETTING INTO IT

Regardless of which Pd package you



Welcome to Pd ("Pure Data"). You can use this window to test audio and MIDI connections. To see Pd's DOCUMENTATION, select "getting started" in the Help menu.

Pd is Free software under the BSD license. See LICENSE.txt in the distribution for details.

Figure 2. Pd's Audio/MIDI Test Suite

choose, your first step into its world should be taken in the Media menu. First, configure your audio and MIDI devices—JACK audio and ALSA MIDI are recommended for Linux users—then select the Test Audio And MIDI item from the menu. This selection runs the testtone.pd patch, a suite of tests to verify your audio/MIDI configuration (Figure 2). When you're satisfied with its results, start opening and running the example files found in /usr/lib/pd/doc/ or /usr/local/lib/pd/doc/.

LEARNING TOOLS

Searching on Google and Google Videos provides plenty of tutorials and presentations on Pd (see Resources for some of my favorites); however, Pd's online documentation is one of the system's great strengths. Cool and useful example files abound in /usr/lib/pd/doc/ (or /usr/local/lib/pd/doc/), complete and ready to run. They include a wonderful variety of synthesis techniques, effects and dynamics processors, GUI methods and so forth. I find the synthesis examples of special interest. Many techniques are presented that are not so easily accessed in other systems—for example, SSB modulation and four different FM synths. Users are encouraged to use the example instruments as the basis for their own instrument designs and compositions, a matter accomplished with simple cut-and-paste operations. Editing patches is equally simple, with all edits done directly on the patch canvas. Pd patches can be represented as plain-text files, but they rarely are edited directly and more often are used for exchanging patches between users. The GUI is where the editing action takes place in Pd.

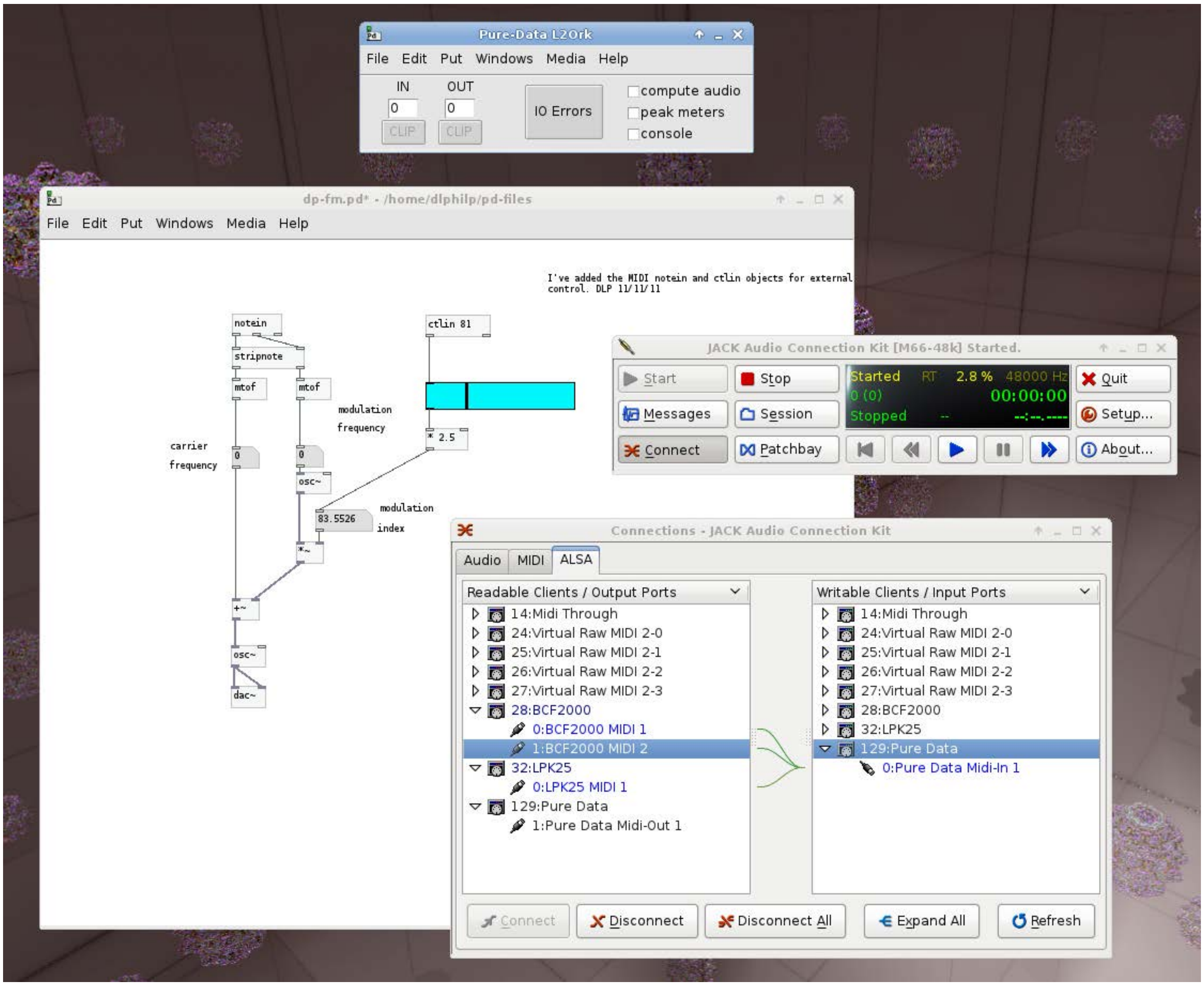
By the way, when you don't want to break your work flow to look up a module's definition, simply right-click over any box to summon a pop-up menu that includes a selection for the object's Help window. The Help file describes the object's functions and provides a summary of its parameters. New users should note that the Help file itself is an active patch, so be careful about changing its default values.

Given Pd's longevity, it's no surprise to find a rich set of third-party resources for its study and expansion. If the program's own documentation isn't enough, you can find a great FLOSS Pd manual on the Net, and other Pd-oriented books abound (see Resources). I recommend especially *Designing Sound*, by Andy Farnell. Apart from its great significance to users of Pd, *Designing Sound* is a most engaging and deeply fascinating book. Ostensibly targeted toward game sound designers, this book is a treasure for anyone who works with digital audio. It covers a vast range of subject material, all relevant to his topic at large, much of which is potentially difficult to impart to a non-specialist reader. However, Mr Farnell is a master of exposition and explanation, and I'm absorbed by whatever I read in his book. *Designing Sound* has a particular appeal for Pd people—every example in the book is presented in Pd, and although the author states it is not a Pd tutorial, beginners may find his presentation especially enjoyable.

MAKING CONNECTIONS

Pd loves external connections. Indeed, it's all just pure data to Pd—if you can get the data into Pd, the program will make every attempt to process it. That's good news for me, because I want to use it for processing input from a variety of external devices.

For external MIDI control over my patches, I have a Behringer BCF2000 control surface and an Akai LPK25 2-octave keyboard, both of which were instantly recognized by my Arch system and by QJackCtl. I used QJackCtl's ALSA connections panel to hook the hardware



I've added the MIDI notein and ctlin objects for external control. DLP 11/11/11

Readable Clients / Output Ports
- 14:Midi Through
- 24:Virtual Raw MIDI 2-0
- 25:Virtual Raw MIDI 2-1
- 26:Virtual Raw MIDI 2-2
- 27:Virtual Raw MIDI 2-3
- 28:BCF2000
  - 0:BCF2000 MIDI 1
  - 1:BCF2000 MIDI 2
- 32:LPK25
  - 0:LPK25 MIDI 1
- 129:Pure Data
  - 1:Pure Data Midi-Out 1

Writable Clients / Input Ports
- 14:Midi Through
- 24:Virtual Raw MIDI 2-0
- 25:Virtual Raw MIDI 2-1
- 26:Virtual Raw MIDI 2-2
- 27:Virtual Raw MIDI 2-3
- 28:BCF2000
- 32:LPK25
- 129:Pure Data
  - 0:Pure Data Midi-In 1

Figure 3. Hardware Controller Inputs in a Pd Patch

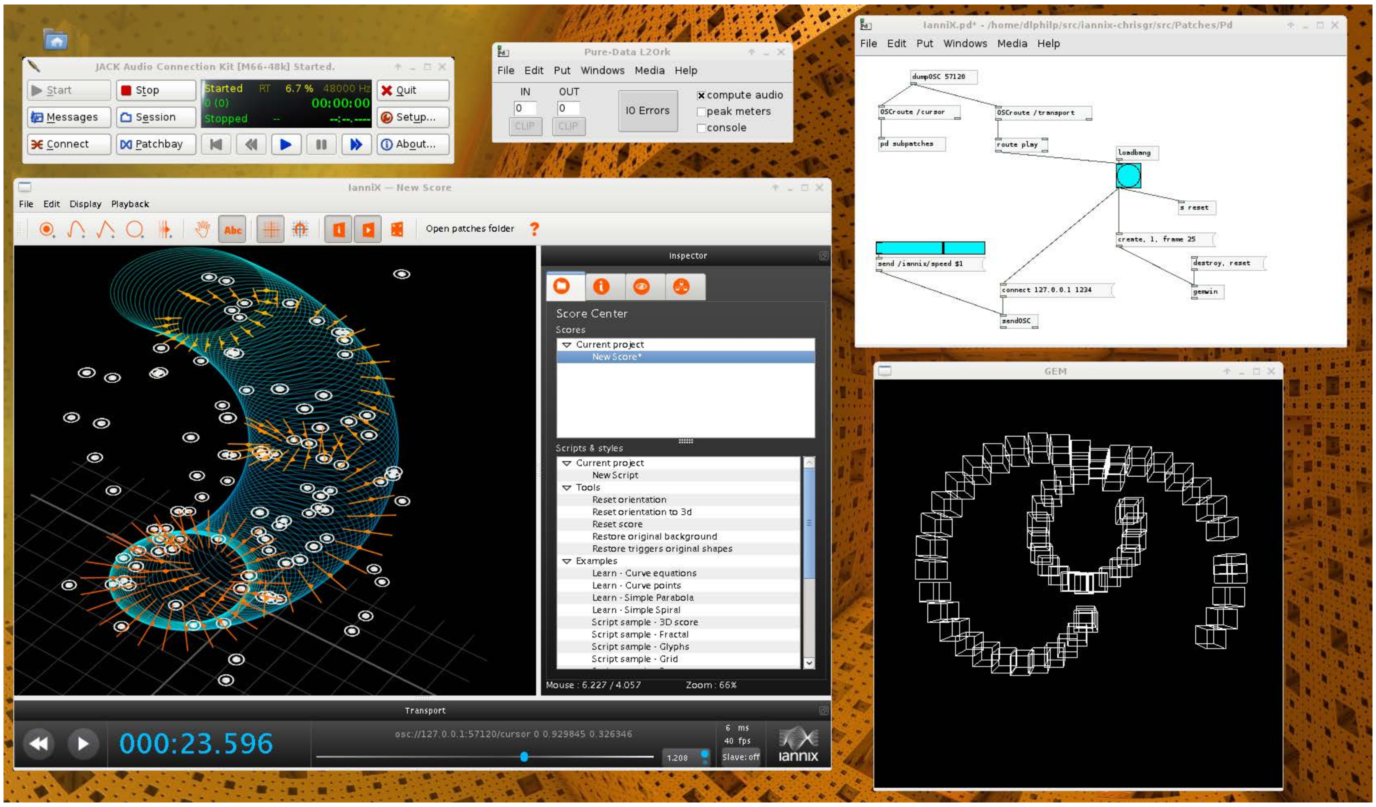

JACK Audio Connection Kit [M66-48k] Started.
Pure-Data L2Ork
IanniX.pd* - /home/dlphilp/src/iannix-chrisgr/src/Patches/Pd
IanniX — New Score
Inspector
Score Center
Transport
000:23.596
GEM
IanniX

Figure 4. Pd Plays with IanniX

into Pd where notein and ctlin objects receive data from my MIDI hardware (Figure 3). Thus, when I move a slider on the BCF, the slider in my Pd patch moves in smooth sync with the hardware controller. I use the LPK to make sudden changes in visual displays, and sometimes I use it simply to play notes into a Pd synthesizer patch.

I tested Pd's OSC support by connecting it to the IanniX OSC sequencer. Thanks to user/developer codex99, I tested a pre-release of IanniX that includes a neat example for working with Pd. The screenshot in Figure 4 shows the connection at work. Pd controls the speed of the sequencer while IanniX controls the animation in the gemwin display window. It's all very cool, and it opens the way to some fascinating possible uses. Perhaps you'd like an OSC-enabled arrangement of Pd + IanniX + Ardour3? No problem, and you might as well throw in connections with Processing and Renoise too, perhaps with a machine per program and everything connected by netsend/netreceive objects.

I also have a Webcam and a miniDV camera that I wanted to hook into Pd. As I've noted already, Pd itself has no video processing capabilities, but many of its externals are dedicated to visual data acquisition and massage. The good news is that the physical connections—USB and FireWire ports and drivers (V4L2 and IEEE1394)—are well supported by those externals. The bad news is that I had to jump through more than a few hoops to make it all work.

VIDEO PROBLEMS AND SOLUTIONS

The packaged version of GEM worked well for everything except patches involving my cameras. Video file playback worked well enough, but I had troubles with live video input. Neither my Webcam nor my miniDV camera were recognized, although both worked fine in the Cheese and Kino programs. A bit of research led me to conclude that my installed version of GEM was borked, so I decided to take the opportunity to upgrade. I built and installed GEM 0.93.3 (with support for its Pd external, of course), and my Webcam came to life in my experiments with the pix_video module. Alas, my miniDV camera remained moribund. Once again, a little research led to the discovery that my Arch system's 3.0 kernel introduced some changes in its support for FireWire devices. So, I ran this command:

```
chmod 666 /dev/fw1
```

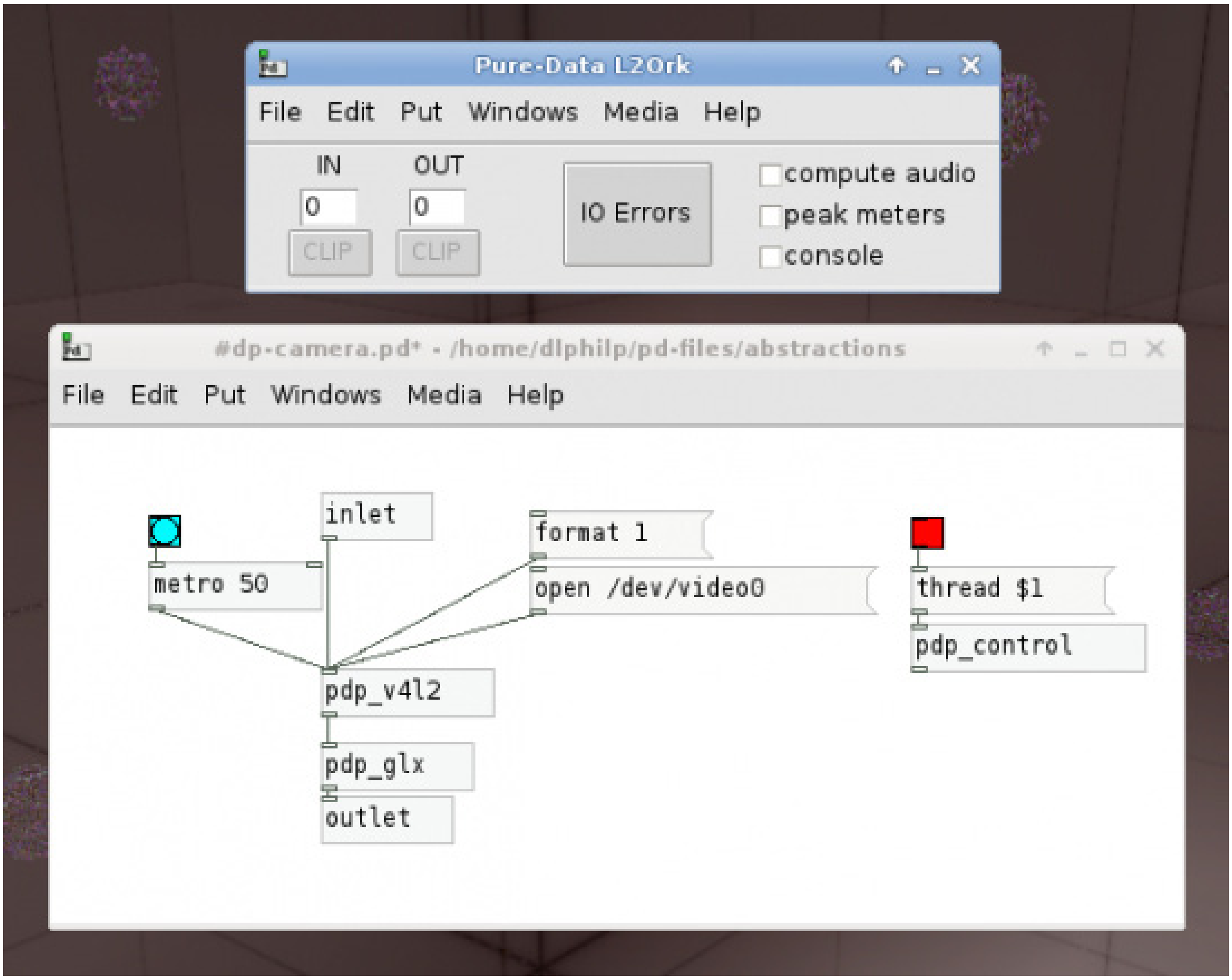
Then, I specified /dev/fw1 as my FireWire device for pix_video. When I selected /dev/fw1 in my patch, the Pd console window complained with this error message:

```
error: Cannot open '/dev/fw1': 25, Inappropriate ioctl for device
```

However, the device worked perfectly in the patch, and now I have video I/O for both devices.

PDP, PiDiP and GridFlow similarly were crippled by problems with the relevant kernel modules. I rebuilt and re-installed all three externals, but I also needed to change my patches to accommodate pd-v4l2 instead of pd-v4l. At last, my test patch opened my Webcam with this series of informative messages:

```
pdp_v4l2: opening /dev/video0
pdp_v4l2: driver info: uvcvideo 1.1.0 / UVC Camera (046d:09a1)
➥@usb-0000:00:0b.1-5
pdp_v4l2 : input 0 : Camera 1
pdp_v4l2: device has 1 inputs
pdp_v4l2: switched to input 0
pdp_v4l2: device supports 0 standards
pdp_v4l2 : format 0 : MJPEG
pdp_v4l2 : format 1 : YUV 4:2:2 (YUYV)
pdp_v4l2: device supports 2 formats
pdp_v4l2 : current format is : MJPG
pdp_v4l2 : setting format : index : 0 : pixel format : MJPG
pdp_v4l2 : capture format : width : 320 : height :240 :
➥bytesperline : 0 : image size : 102400
control 9963776 active (i:0)
control 9963777 active (i:1)
control 9963778 active (i:2)
control 9963788 active (i:12)
control 9963795 active (i:19)
control 9963800 active (i:24)
control 9963802 active (i:26)
control 9963803 active (i:27)
control 9963804 active (i:28)
pdp_v4l2: got 8 buffers type 1 memory 1
```


Pure-Data L2Ork
File Edit Put Windows Media Help
IN
OUT
0
0
CLIP
CLIP
IO Errors
compute audio
peak meters
console
#dp-camera.pd* - /home/dlphilp/pd-files/abstractions
File Edit Put Windows Media Help

```
inlet
metro 50
format 1
open /dev/video0
thread $1
pdp_control
pdp_v4l2
pdp_glx
outlet
```

Figure 5. The #dp-camera Abstraction

```
pdp_v4l2 : mapped 8 buffers
pdp_v4l2 : queued 8 buffers
pdp_v4l2 : device initialized
pdp_v4l2 : created thread : 2412164864
pdp_v4l2 : capture started
```

Everything looked good up to this point, but then the console printed this line:

```
pdp_v4l2: unsupported color model: 1196444237
```

My patch's rendering window opened, the Webcam's LED was on, but I had no picture until I figured out that the "unsupported color model" resulted from the MJPG format setting. I changed the format value to 1 (for YUYV support), and voilà, my smiling face appeared in the patch's rendering display. Finally, I have support for my Webcam in my Pd video patches. Figure 5 shows the basic camera configuration for my PDP/PiDiP video patches.

Incidentally, I use this patch now as an abstraction in my own versions of the tutorial patches.

Alas, the situation with my miniDV camera is not so good. PDP/PiDiP includes the pdp_ieee1394 object, but it doesn't like /dev/fw1, and my Arch system provides no other appropriate device node (such as /dev/dv1394). I ran Pd as superuser, but still got no joy with the camera. I'll continue to search for a solution, and in the meanwhile, I can use GEM when I want DV support.

IT TAKES TWO

In one of my experiments, I created a patch that joined a video processor with an audio synthesizer, both of which elements were controlled by a single slider. The patch worked, but the video playback seriously interfered with the audio continuity. Fortunately, I found a solution: I separated the patch into two patches, then I ran those patches in two instances of Pd, launching one instance with the -noaudio option for the video processor and starting the other with the -rt flag for the audio synthesis. Both patches include a ctlin object that receives data from a slider on an external MIDI control surface (see above). With this arrangement, the graphics are updated smoothly, and the audio playback glitches are gone. This solution is not as elegant as the single-patch design, but it's definitely the better performer on my aged single-core CPU. I learned later that a two-machine solution is available with the netsend/netreceive objects. I haven't tried it yet, but it's on my short list of neat things to do with Pd.

THE PD EVERYWHERE PROJECT

Pd can be decoupled from its default GUI, a feature that has drawn the attention of developers Peter Brinkmann and Peter Kirn. Thanks to their libpd Project, Pd-based software can be run on mobile devices and systems, such as the Android and the iOS. The library can be embedded directly into devices, promising a new generation of Pd-based audio software for the little and not-necessarily-so-little things. You can keep up with libpd's development at the Pd Everywhere site and on Peter Brinkmann's blog (see Resources). It's cool—check it out.

As this article went to press, I received a copy of *Making Musical Apps: Real-Time Audio Synthesis on Android and iOS*, a new book written by Peter Brinkmann. It's a must-have item if you're looking for a way to use Linux to program audio applications for the new mobile devices. The book has also persuaded me that I really need to get an Android.

MORE POWER PD

Pd has its power users, and some of those users have Web sites dedicated to their Pd-related productions and investigations. Frank Barknecht is a true Pd wizard (and keen cyclist); you'll want to look at his footils.org site and check out his work on the RRADical and RjDj Projects. Chris McCormick's WebPd Project is very cool, as is his Squeakyshoecore music (made with Pd, of course), and longtime user/developer Guenter Geiger recently has expanded his Pd-related work into the PDa (Puredata anywhere) Project, an effort to bring Pd's power to Linux-friendly PDA devices.

Dr Albert Graef has lent his considerable talents to Pd in the form of his Pd-Faust software. The project includes a new system for running Faust plugins in Pd, with dynamic reloading and on-the-fly GUI generation. It's available from the Pure Web site (see **http://code.google.com/p/pure-lang/wiki/Addons#pd-faust**) for further details. Dr Graef notes that getting it all up and running may be a bit tedious—it has dependencies not usually found in the average package repos—but the latest release with all needed dependencies is available in the PlanetCCRMA repository at **http://ccrma.stanford.edu/planetccrma/mirror/fedora/linux/planetccrma/14/x86_64/repoview/pd-faust.html**.

These days, it seems everyone wants to get in touch with Pd. I've already noted that the Renoise tracker/DAW and the IanniX OSC sequencer include examples for connecting those worthies to the power of Pd, and even the mighty Csound provides a Csound-to-Pd bridge. These definitely are good days, and good company, for Pd.

By the way, if you're looking for a quick introduction to some of Pd's sonic capabilities, check out some of the offerings on Soundcloud and the contributions at Puredata.info. Soundcloud's Pure Data community has uploaded more than 60 recordings, and you're bound to find something of interest there.

THE WRAP

As a veteran Csounder, I must say I am impressed with Pd's synthesis capabilities. However, I also must confess that I'm most attracted to Pd for its extended services, particularly in the graphics and video realms. I continue to look into the possibilities of the PDP/PiDiP and GridFlow packages, and I have so much more to learn about GEM's capabilities. I'm also interested in using it for poetry and prose text manipulation—Pd is that flexible, as evidenced by its many uses in real-world deployment.■

Dave Phillips is a musician/author/teacher living and working in Findlay, Ohio. He has worked with computers for musical purposes since 1985. He discovered Linux in 1995 and created the http://linux-sound.org Web site shortly after. He has written on audio topics for *Linux Journal* and other publications since 1998 and wrote *The Book Of Linux Music & Sound* (NoStarch Press) in 2000. His extracurricular activities include practicing t'ai-chi, reading Latin poetry, walking a shar-pei named Maximus, and spending as much time as possible with a woman named Ivy.

Resources

"Super Collision At Studio Dave: The New World of SuperCollider3, Part 1" by Dave Phillips: **http://www.linuxjournal.com/content/super-collision-studio-dave-new-world-supercollider3-part-1**

Pure Data: **http://puredata.info**

Max (software): **http://en.wikipedia.org/wiki/Max/MSP**

Cycling74: **http://cycling74.com**

"Vanilla" Pd Release (maintained by Miller Puckette): **http://crca.ucsd.edu/%7Emsp/software.html**

Pd-extended: **http://puredata.info/community/projects/software/pd-extended**

Pd-L2Ork Project: **http://l2ork.music.vt.edu/main/?page_id=56**

Puredyne: **http://puredyne.org**

Programming Electronic Music in Pd by Johannes Kreidler: **http://www.pd-tutorial.com**

Obiwannabe Home: **http://obiwannabe.co.uk**

vreahli's channel (YouTube): **http://www.youtube.com/user/vreahli**

Pure Data: Lesson 01, Hello World! by Dr. Rafael Hernandez (YouTube): **http://www.youtube.com/watch?v=rtgGol-I4gA**

Borgmekanik's Pd Videos (YouTube): **http://www.youtube.com/user/borgmechaniker**

Pd + GEM Creations from Boris.volant and Vincent Prijent (YouTube): **http://www.youtube.com/user/trahisonM/videos**

FLOSS Pure Data Manual: **http://flossmanuals.net/puredata**

pd-graz Group's *bang* Book: **http://pd-graz.mur.at/label/book**

The Theory and Technique of Electronic Music by Miller Puckette: **http://www.amazon.com/Theory-Technique-Electronic-Music/dp/9812700773/ref=sr_1_1?s=books&ie=UTF8&qid=1325192262&sr=1-1**

Composition: Pure Data as a Meta-Compositional Instrument by Michael Barkl: **http://www.amazon.com/Composition-Pure-Data-Meta-Compositional-Instrument/dp/3838316479/ref=sr_1_1?s=books&ie=UTF8&qid=1325192847&sr=1-1**

Designing Sound by Andy Farnell: **http://www.amazon.com/Designing-Sound-Andy-Farnell/dp/0262014416/ref=sr_1_1?s=books&ie=UTF8&qid=1325192902&sr=1-1**

"An Introduction to OSC" by Dave Phillips: **http://www.linuxjournal.com/content/introduction-osc**

IanniX: **http://www.iannix.org/en/index.php**

Ardour: **http://ardour.org**

Processing: **http://processing.org**

Renoise: **http://www.renoise.com**

Cheese: **http://projects.gnome.org/cheese**

Kino Video Editor: **http://www.kinodv.org**

Pd Everywhere: **http://noisepages.com/groups/pd-everywhere**

Peter Brinkmann's Blog—Pure Data, Android Audio, and Random Stuff: **http://nettoyeur.noisepages.com**

Making Musical Apps: Real-Time Audio Synthesis on Android and iOS by Peter Brinkmann: **http://shop.oreilly.com/product/0636920022503.do**

Frank Barknecht: **http://footils.org**

RRADical: **http://puredata.info/community/projects/rradical**

RjDj: **http://rjdj.me/music/Frank%20Barknecht**

Chris McCormick's WebPd Project: **http://mccormick.cx/projects/WebPd**

Chris McCormick's Squeakyshoecore Music: **http://sciencegirlrecords.com/chr15m/squeakyshoecore**

Pd-Faust: **http://docs.pure-lang.googlecode.com/hg/pd-faust.html**

PDa Project: **http://gige.xdv.org/public_html/pda**

Csound-to-Pd Bridge: **http://booki.flossmanuals.net/csound/csound-in-pd**

Offerings on Soundcloud: **http://soundcloud.com/groups/pure-data-exclusive**

Contributions at Puredata.info: **http://puredata.info/community/tracks**

Parallel Programming in C and Python

A fast-track guide to writing parallel programs in C and Python.

AMIT SAHA

Advancements in computing technologies have enabled even "resource-constrained" Netbooks to have four CPU cores. Thus, parallel computing is not about having a large number of computer nodes in a cluster room anymore. To write programs that take advantage of such easily available parallel computing resources, C and Python programmers have libraries at their disposal, such as OpenMP (C) and multiprocessing (Python) for shared memory parallel programming, and OpenMPI (C/Python) for distributed memory parallel programming. And, in case you don't have hardware resources beyond your quad-core personal computer, cloud-computing solutions, such as PiCloud, can be of tremendous use.

For the purposes of this article, I assume you have one computer at your disposal and, thus, focus on OpenMP, multiprocessing and PiCloud. Note that the article's code examples demonstrate parallel programming and may not be the most optimal strategy possible for parallelization. I also have refrained from any benchmarking to show speedups of the parallel programs versus their serial counterparts.

Shared Memory Parallel Programming

Shared memory parallel programming libraries enable writing programs that can take advantage of the multiple CPUs on your computer.

OpenMP (Open specifications for Multi-Processing) is a standard API specification that may be used explicitly to achieve multithreaded, shared memory parallelism. The API is defined for C/C++ and FORTRAN, and it is composed of three primary API components: compiler directives (#pragma in C), runtime libraries and environment variables. An OpenMP program begins life as the master thread until it encounters a parallel block. A team of parallel threads is then created; this operation is called the fork operation. The statements in this parallel block are executed in parallel by these threads. Once the team of threads has finished execution, the master thread is back in control, which is called the Join operation. This model of execution is referred to as the Fork-Join model.

Let's use the GNU implementation of OpenMP, called libgomp. Your distribution's package manager is the easiest way to install this library. To enable the compiler directives that you use in your OpenMP programs, specify `-fopenmp`, and to link to the runtime library, a `-lgomp` has to be added during the compilation of your OpenMP programs.

Listing 1. ompdemo.c

```
#include <stdio.h>
#include <omp.h>

int main (int argc, char **argv)  {

  int nthreads, tid, i;

  /* Get the number of processors */
  printf("Number of processors available::
  ➥%d\n",omp_get_num_procs());

  /* Set the number of threads to the number of processors */
  omp_set_num_threads(omp_get_num_procs());

  /* Fork a team of threads with each thread having a
     private tid variable * */
#pragma omp parallel  private(tid)
  {
    /* Obtain and print thread id */
    tid = omp_get_thread_num();
    printf("Hello World from thread = %d\n", tid);

    /* Only master thread does this */
    if (tid == 0)
      {
        nthreads = omp_get_num_threads();
        printf("Number of threads = %d\n", nthreads);
      }
  }  /* All threads join master thread and terminate */

  return 0;
}
```

Let's start by creating a team of threads (Listing 1). First, use the `omp_get_num_procs()` function to get the number of processors available, which you use to set the number of threads using the `omp_set_num_threads()` function. Next, specify the parallel block of the program using `#pragma omp parallel`. You want each thread to have a private copy of the tid variable, so append this information to the #pragma directive. In the parallel block, let's simply print a hello world message from all of the threads, including the master thread. The master thread has a tid of 0, and it can be used to execute statements specific to it. The thread id of a particular thread is obtained using the `omp_get_thread()` function.

Once you compile and execute this program, the output, depending on the number of processors on your computer, will be similar to the following:

```
$ gcc -o ompdemo ompdemo.c -lgomp -fopenmp
$ ./ompdemo

Number of processors available:: 2
Hello World from thread = 0
Number of threads = 2
Hello World from thread = 1
```

Next, let's consider a code snippet in your program that may look like this:

New: Intel Xeon E5 Based Clusters

Benchmark Your Code on Our Xeon E5 Based Tesla Cluster with:
AMBER, NAMD, GROMACS, LAMMPS, or Your Custom CUDA Codes

Microway MD SimCluster with
8 Tesla M2090 GPUs
8 Intel Xeon E5 CPUs and InfiniBand
2X Improvement over Xeon 5600 Series

Upgrade to New Kepler GPUs Now!

Microway®

NVIDIA TESLA | intel Technology Provider Platinum 2012 | AMD Fusion Partner Premier | GSA

GSA Schedule
Contract Number:
GS-35F-0431N

```
A -> Large array
For every element e in array A
 call function, fun(e)
end for
```

Instead of calling the function, `fun` for every element, `e`, serially, what if you could do it in parallel? OpenMP's work-sharing constructs enable you to do that. Consider the code snippet below:

```
A -> Large array
#pragma omp for schedule(dynamic,chunk)
For every element e in array A
 call function, fun(e)
end for
```

The directive, `#pragma omp for schedule(dynamic,chunk)` specifies that the ensuing for loop will be executed in parallel by the thread team, with the workload specified by the `schedule(dynamic, chunk)` clause. The `chunk` variable specifies the unit of division—that is, the number of iterations of the for loop that will be executed by a thread, and `dynamic` specifies that the chunks will be assigned dynamically to the threads.

Let's use this directive to process a large array, calling a function for each array element (Listing 2). The parallel section of the code begins at the directive `#pragma omp parallel` where we specify that the arrays a and b and the variable `chunk` will be shared

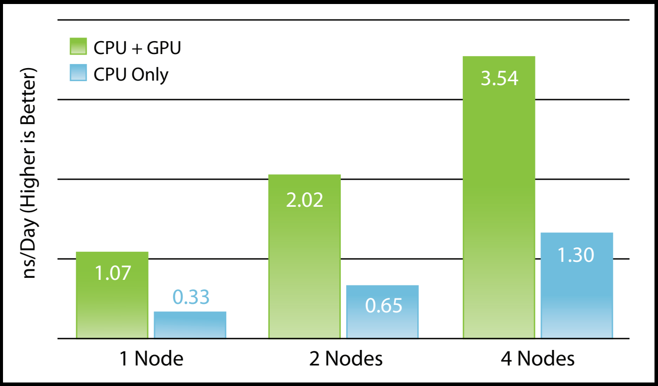
Harness Microway's Proven GPU Expertise

Thousands of GPU cluster nodes installed.
Thousands of WhisperStations delivered.

- **Award Winning BioStack – LS**
- **Award Winning WhisperStation Tesla – PSC with 3D**

Bio-IT World CONFERENCE & EXPO '11 BEST of SHOW WINNER

Life Sciences insight '11 AWARD WINNER for BEST Best New Technology



NAMD F1-ATP Performance Gain

Configure Your WhisperStation or Cluster Today!
www.microway.com/tesla or 508-746-7341

Listing 2. omp_for_eval.c

```
/* Work Sharing construct
https://computing.llnl.gov/tutorials/openMP/#DO

Distributes an array of elements across threads, where each
element is passed as a parameter to a function to be evaluated

*/

#include <omp.h>
#include <stdio.h>
#define N 100000
#define CHUNKSIZE 100
/* dummy function*/
float myfun(float a)
{
  return a*a;
}

int main (int argc, char **argv)
{

  int i, chunk,tid;
  float a[N], b[N];

  for (i=0; i < N; i++)
    a[i] = i * 1.0;

  /* Get the number of processors */
  printf("Number of processors available::
  ➥%d\n",omp_get_num_procs());

  /* Set the chunk size*/
  chunk = CHUNKSIZE;

  /* Set the number of threads to the number of processors*/
  omp_set_num_threads(omp_get_num_procs());

#pragma omp parallel shared(a,b,chunk) private(i)
  {

#pragma omp for schedule(dynamic,chunk)
    for (i=0; i< N; i++)
      {
        b[i] = myfun(a[i]);
      }
  }  /* end of parallel section */

  printf("For evaluation completed, the result
  ➥has been stored in array B\n");

  return 0;
}
```

among the threads. This is required, because all the threads will be accessing the arrays a and b, and `chunk` will be used by the ensuing parallel for loop to determine the unit of division. The variable, i, specifying the loop, will be private to each thread.

The result of the function evaluation will be available at the end of the parallel section in the array b:

```
$ gcc -o omp_for_eval omp_for_eval.c -lgomp -fopenmp
$ ./omp_for_eval
Number of processors available:: 2
For evaluation completed, the result has been stored in array B
```

Listing 3. pi_openmp.c

```
/* Program to compute Pi using Monte Carlo method:
(http://math.fullerton.edu/mathews/n2003/montecarlopimod.html)
 */

#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <string.h>

/* Returns the value of count with niter iterations*/
int part_count(int niter)
{
  int i, count=0;
  float x,y,z;

  for ( i=0; i<niter; i++)
    {
      x = (double)rand()/RAND_MAX;
      y = (double)rand()/RAND_MAX;
      z = x*x+y*y;
      if (z<=1) count++;
    }

  return count;
}


int main(int argc, char* argv)
{
  int niter=0,chunk;
  double x,y;
  int i,count=0; /* # of points in the 1st
                    quadrant of unit circle */
  double z;
  double pi;

  /* Get the number of processors */
  printf("Number of processors available::
➥%d\n",omp_get_num_procs());

  printf("Enter the number of iterations used to
➥estimate pi (multiple of %d please):
➥",omp_get_num_procs());
  scanf("%d",&niter);


  /* Set the number of threads to the number of processors*/
  omp_set_num_threads(omp_get_num_procs());
  chunk = niter/omp_get_num_procs();

#pragma omp parallel shared(chunk) reduction(+:count)
  {
    count = part_count(chunk);
  }

  pi=(double)count/niter*4;
  printf("# of iterations = %d , estimate of pi is
➥%g \n",niter,pi);

  return 0;
}
```

For the next example, let's calculate the value of Pi using a Monte Carlo sampling technique. The core of the algorithm involves generating pairs of random points in the range (0,1) and keeping a count of the number of points that lie in the first quadrant of a unit circle. The number of samples generated have a direct effect on the accuracy of the estimate of the value of Pi. With the help of OpenMP, let's divide the number of samples equally among multiple threads and finally accumulate the result in the master thread to calculate the value of Pi (Listing 3).

In this code, notice the use of the reduction clause, `#pragma omp parallel shared(chunk) reduction(+:count)`. This effectively means to perform the + operation on the private copies of the variable `count`, and store it in the global shared variable `count`. The value of `count` then is used to calculate the value of Pi:

```
$ gcc -o pi_openmp pi_openmp.c -lgomp -fopenmp
$ ./pi_openmp
Number of processors available:: 2
Enter the number of iterations used to estimate pi
➥(multiple of 2 please): 10000
# of iterations = 10000 , estimate of pi is 3.1636
[gene@zion openmp]$ ./pi_openmp
Number of processors available:: 2
Enter the number of iterations used to estimate pi
➥(multiple of 2 please): 1000000
# of iterations = 1000000 , estimate of pi is 3.14222
```

Python's Multiprocessing Module

Python's multiprocessing module effectively side-steps the Global Interpreter Lock that inhibits true multithreaded Python programs. With this module, your Python programs can make use of multiple processors on your computer. Listing 4 is a Python program that uses this module to get the number of CPUs on your computer and creates that number of processes.

A process is created by creating an object of the Process class: `p= Process(target=f, name='Process'+str(i), args=(i,))`.

Listing 4. mpdemo.py

```
'''
Create number of processes using the multiprocessing module
'''

import multiprocessing
from multiprocessing import Process

# dummy function
def f(id):
    #This is a dummy function taking a parameter
    return

if __name__ == '__main__':

    # get the number of CPUs
    np = multiprocessing.cpu_count()
    print 'You have {0:1d} CPUs'.format(np)

    # Create the processes
    p_list=[]
    for i in range(1,np+1):
        p = Process(target=f, name='Process'+str(i), args=(i,))
        p_list.append(p)
        print 'Process:: ', p.name,
        p.start()
        print 'Was assigned PID:: ', p.pid

    # Wait for all the processes to finish
    for p in p_list:
        p.join()
```

Listing 5. mp_queue.py

```
'''
Queue: http://docs.python.org/library/
➥multiprocessing.html#pipes-and-queues

Demonstrates the usage of Queue to share data between processes.
Splits up a large array into chunks and calculates the partial
dot products.
'''

import multiprocessing
from multiprocessing import Process, Queue
from numpy import *

# dot product of the partial data chunks
def add(chunk1,chunk2,product):
    a = chunk1.get()
    b = chunk2.get()

    prod = a*b
    product.put(sum(prod))

if __name__ == '__main__':

    #size of the arrays
    num_el = 100000

    # Create two arrays
    a = linspace(0,100,num_el);
    b = linspace(0,1,num_el);

    # get the number of CPUs and assign it as the number of
    # processes to create
    np = multiprocessing.cpu_count()
    print 'You have {0:1d} CPUs'.format(np)

# chunk size
if num_el%np != 0:
    print "The current chunking mechanism will not work"
    exit
else:
    chunk = num_el/np

# Create the processes
p_list=[]

# Create the Queue which will have the partial products
product=Queue()

for i in range(1,np+1):

    # A pair of queues per process for the two arrays
    aq = Queue()
    bq = Queue()

    # push the chunks into the queue
    aq.put(a[(i-1)*chunk:i*chunk])
    bq.put(b[(i-1)*chunk:i*chunk])

    # create the process
    p = Process(target=add, args=(aq,bq,product))
    p.start()
    p.join()

# collect the individual sums
items=[]
for i in range(product.qsize()):
    items.append(product.get())

# final product: sum of individual products
print "Dot product:: ",sum(items)
```

The `target` argument specifies the callable object to be called by the process's `run()` method, which is invoked by the `start()` method; `name` specifies a custom name for the process; and `args` specifies the argument tuple for the target. Depending on the number of processors on your computer, you should see output similar to the following:

```
$ python mpdemo.py
You have 2 CPUs
Process::  Process1 Was assigned PID::  29269
Process::  Process2 Was assigned PID::  29270
```

This program maintains a list of the created process objects so as to wait for them to finish using the `join()` method. If you attempt to use this example as a starting point to actually do something, you will want a way to send back information from the spawned processes to the master process. There are a few ways to achieve this. The easiest, but not really recommended way, is to use shared state variables. Another way to establish a communication channel between the master and the spawned processes is to use Queues. Listing 5 uses the multiprocessing module to calculate the dot product of a large array by splitting the calculation across multiple processes.

For every process created, a pair of queue objects are created, one each for the two arrays. The whole array is divided into equal chunks, and a process is assigned a chunk to calculate the partial product. The partial products are stored in the queue object `product`. After the processes have finished execution, partial products are retrieved from the queue and summed to get the actual dot product. Here is sample output:

```
$ python mp_queue.py
You have 2 CPUs
Dot product::  3333350.00017
```

Next, let's take a look at Process pools. As the name implies, a pool of processes is created using the `Pool` class, and each member of this process pool is then assigned a task to execute using methods, such as `apply()` or `map()`. Listing 6 demonstrates using Process pools to calculate the value of Pi using the same algorithm as shown in Listing 3.

The pool of processes is created using the statement `pool = Pool(processes=np)`, and each process in this pool is asked to invoke the function `monte_carlo_pi_part` using `count=pool.map(monte_carlo_pi_part, part_count)`, where `part_count` is a list with the number of samples to be generated in each process. The returned value from each process is stored in the list, `count`. Finally, let's sum this list and calculate the estimated value of Pi:

```
$ python pi_mp.py
You have 2 CPUs
Estimated value of Pi::  3.1412128
```

The call to `map()` is a blocking call—that is, the process doesn't terminate until the result has been received. For other applications, the

Listing 6. pi_mp.py

```
'''
Multiprocessing-based code to estimate the value of PI
using Monte Carlo sampling.
Ref: http://math.fullerton.edu/mathews/n2003/montecarlopimod.html
Uses workers:
http://docs.python.org/library/
➥multiprocessing.html#module-multiprocessing.pool
'''

import random
import multiprocessing
from multiprocessing import Pool


#calculate the number of points in the unit circle
#out of n points
def monte_carlo_pi_part(n):

    count = 0
    for i in range(n):
        x=random.random()
        y=random.random()

        # if it is within the unit circle
        if x*x + y*y <= 1:
            count=count+1

    #return
    return count


if __name__=='__main__':

    np = multiprocessing.cpu_count()
    print 'You have {0:1d} CPUs'.format(np)

    # Number of points to use for the Pi estimation
    n = 10000000

    # iterable with a list of points to generate in each worker
    # each worker process gets n/np number of points to
    # calculate Pi from

    part_count=[n/np for i in range(np)]

    #Create the worker pool
    # http://docs.python.org/library/
      ➥multiprocessing.html#module-multiprocessing.pool
    pool = Pool(processes=np)

    # parallel map
    count=pool.map(monte_carlo_pi_part, part_count)

    print "Estimated value of Pi:: ", sum(count)/(n*1.0)*4
```

asynchronous version may be desirable.

The multiprocessing module has other features, such as the ability to execute remote jobs by the use of managers, proxy objects and a number of synchronization primitives. See the module documentation for more information.

Parallel Computing in the Cloud with PiCloud

If you have run out of cores on your workstation and want to make more computing power accessible to your program easily, it's worth looking into PiCloud.

To start using PiCloud, create an

account on the project Web site, and then install the client from the PyPi package index. The Quickstart documentation should help you set up PiCloud on your computer (see Resources).

Let's write a simple function and run it on PiCloud. First, define a function `sort_num()` in the file demo.py as follows:

```
import numpy

def sort_num(num):
    sort_num = numpy.sort(num)
    return sort_num
```

Fire up the Python interpreter and import the function you just defined and the modules cloud and numpy:

```
>>> from demo import *
>>> import cloud
>>> import numpy
```

Next, create an array of 50000 integers with each integer chosen randomly from the range (10, 10000):

```
>>> num=numpy.random.random_integers(10,10000,50000)
```

Now comes the important step, invoking the `cloud.call` method to specify the function that you want to run and its arguments:

```
>>> jid=cloud.call(sort_num,num)
```

This implies that you want the function `sort_num` to be executed with the argument num. This method returns an integer that is an identifier for this particular job. The formal specification of the `cloud.call` method is available at **http://docs.picloud.com/moduledoc.html#cloud.call**:

```
>>> jid
58
```

The returned value of the executed function can be obtained using the function `cloud.result` using the obtained jid:

```
>>> cloud.result(jid)
array([   10,    11,    11, ..., 10000, 10000, 10000])
```

As you can see, the returned array is obtained. Next, consider a function (defined as a file mapdemo.py), which returns the volume of a cylinder when the radius(r) and height(h) is passed to it:

```
import numpy

def vol_cylinder(r,h):
    return numpy.pi*r*r*h
```

If you had one cylinder configuration—that is, one pair of (r,h)—the `cloud.call` method can be used for the purpose. What if you had 100 different configurations to square? You could use 100 `cloud.call` invocations, but there is a more efficient way of doing it: using the `cloud.map` function, which accepts a sequence of parameters (or sequences of parameters):

```
>>> from mapdemo import *
>>> import cloud
>>> import numpy
>>> r=numpy.random.random_sample(100)
>>> h=numpy.random.random_sample(100)
>>> jids=cloud.map(vol_cylinder,r,h)
>>> jids
xrange(359, 459)
```

As you can see, `jids` is a list of 100 elements with values from 359 to 459 (note that the values you have may be different). Here, `cloud.map` has created 100 jobs with the arguments formed from the 100 pairs of the two sequences, r and h. You can pass this list of `jids` directly to `cloud.result` to get the results as a list:

```
>>> result=code.result(jids)
>>> result
[0.35045338986267927, 0.0043144690799585004,
➥0.094018119621969765, 0.93579722612039329,
➥0.0045154147876736109, 0.018836324478609345,
➥0.0027243595262778321, 1.5049675511377265,
➥0.37383416636274164, 0.24435487403102638,
➥0.28248315493701553, 1.2879340600324913,
➥0.68406526971023041, 0.14338739850272786,...
```

In more practical situations, the function computation may not be so trivial, and you may not have a definite idea when all the jobs will be over. In such a case, you can wait until all the jobs have finished to retrieve the results using the function `cloud.join`. The PiCloud documentation uses the map function to calculate the value of Pi (yes, again!) using the `map()` method (see Resources).

Among a host of other interesting features, PiCloud allows your Python application to expose your functions in PiCloud via a REST API, which will allow you to call this function from any other programming language. See the Resources section of this article for an example. PiCloud allows you to set up cron jobs, use different computing resources and use environments to install any native libraries for use in your application.

Distributed Memory Parallel Programming

The past few sections in this article give a taste of parallel programming from a device as resource-constrained (relatively!) as a dual-core desktop. What if you already have a 30-node computing cluster at your disposal? In that case, you would use the tried-and-tested OpenMPI, an implementation of the MPI-2 specification. Another solution for setting up a computing cluster is the Parallel Virtual Machine (PVM). If you don't have multiple computers and still want to try out OpenMPI programming, you simply can run it on a standalone computer or use a virtualization solution, such as VirtualBox, to create a cluster of virtual machines.

Conclusion

My intention with this article was to provide an introduction to parallel programming. Because the hardware requirements are really basic, I mostly looked at shared memory parallel programming. I also covered a more modern way of parallel computing using a cloud-computing service. Another very interesting project, Parallel Python, enables writing shared memory as well as distributed memory parallel programs in Python. I hope the examples in this article help you get started exploring parallel programming from the comfort of your Netbook!

Amit Saha is currently a PhD research student in the area of Evolutionary Algorithms and Optimization. Like his random echoes show (http://echorand.me), he has been writing on myriad Linux and open-source technologies for the past five years. He welcomes comments on this article and beyond at amitsaha.in@gmail.com.

Resources

The code from this article is available at **https://bitbucket.org/amitksaha/articles_code/src/6c95473182a5/parallel_article**.

Parts of the section on PiCloud were adapted from my article on PiCloud, titled “PiCloud: Easy way to the Cloud” published in *Linux For You* (**http://www.lfymag.com**), March 2012.

OpenMP Tutorial: **https://computing.llnl.gov/tutorials/openMP**

GNU OpenMP Implementation: **http://gcc.gnu.org/onlinedocs/libgomp**

Monte Carlo Sampling Technique: **http://math.fullerton.edu/mathews/n2003/montecarlopimod.html**

Process Class: **http://docs.python.org/library/multiprocessing.html#process-and-exceptions**

Shared State Variables: **http://docs.python.org/library/multiprocessing.html#sharing-state-between-processes**

Queues: **http://docs.python.org/library/multiprocessing.html#exchanging-objects-between-processes**

Python Multiprocessing Documentation: **http://docs.python.org/library/multiprocessing.html**

PiCloud Home Page: **http://www.picloud.com**

PiCloud Documentation: **http://docs.picloud.com**

PyPi Package Index: **http://pypi.python.org/pypi/cloud**

PiCloud Quickstart Documentation: **http://docs.picloud.com/quickstart.html**

cloud.map Function: **http://docs.picloud.com/client_basic.html#mapping**

cloud.join Function: **http://docs.picloud.com/moduledoc.html#cloud.join**

Calculating the Value of Pi Using the map() Method: **http://docs.picloud.com/basic_examples.html#calculating-pi**

PiCloud Features: **http://www.picloud.com/product/#features**

Using a C Client to Invoke a Function Published via the REST API: **http://echorand.me/2012/01/27/picloud-and-rest-api-with-c-client**

Running an Evolutionary Algorithm in the Cloud with PiCloud: **http://echorand.me/2012/01/26/picloud-pyevolve-evolutionary-algorithms-in-the-cloud**

OpenMPI Home Page: **http://www.open-mpi.org**

Parallel Programming Tutorial: **https://computing.llnl.gov/tutorials/parallel_com**

Beginner MPI Tutorial: **http://www.mpitutorial.com/beginner-mpi-tutorial**

MPI Python Bindings (mpi4py): **http://mpi4py.scipy.org**

PVM Home Page: **http://www.csm.ornl.gov/pvm**

PVM Book: **http://www.netlib.org/pvm3/book/pvm-book.html**

Parallel Python: **http://www.parallelpython.com**

O'REILLY®

fluent conference

JavaScript & Beyond

Master the Web's Most Important Technologies

MAY 29 – 31, 2012
SAN FRANCISCO, CA

Steve Souders
Performance Evangelist, Google

Brendan Eich
CTO, Mozilla

Ben Galbraith,
VP, Mobile Engineering, Walmart.com

Amy Hoy
Founder, Charm

Explore the Changing Worlds of JavaScript, HTML5, and Beyond

Make sense of the vast explosion of JavaScript and related technologies and take away practical skills you can apply immediately at the **O'Reilly Fluent Conference**.

Conference tracks:

- Ancillary Technologies
- JavaScript in the Browser
- Gaming
- Mobile
- Node.js
- Pure Languages
- Stack Track

Fluent Conference is for JavaScript developers, web developers, web performance engineers, and mobile app developers.

Registration is now open at fluentconf.com

Save 20% with code **LINUX**

Replicate Everything! Highly Available iSCSI Storage with DRBD and Pacemaker

In high-availability clusters, redundancy of data is just as crucial as redundancy of nodes. In the second installment of his series on high availability, Florian Haas explains building rock-solid, block-replicated iSCSI data storage with Pacemaker and DRBD. FLORIAN HAAS

Any high-availability solution is only as good as its data. Historically, high-availability clusters often have single-instance, hardware-based storage solutions—the classic and widespread SAN. Interestingly, this is one of the few remaining strongholds of proprietary solutions, even in environments that are otherwise dominated by open-source technology. It's also often an extremely expensive stronghold.

The first article in this series on Linux high availability covered the basics of the stack, taking storage for granted [see Florian's article "Ahead of the Pack: the Pacemaker High-Availability Stack" in the April 2012 issue]. This installment covers building redundant, automatically replicated, highly available cluster storage that can act as a drop-in replacement for costly and proprietary SAN-based solutions.

Linux iSCSI: a Tale of Four Targets

The highly available storage stack on Linux centers predominantly on iSCSI, an implementation of the SCSI protocol over IP networks. iSCSI, defined in RFC

3720 back in 2004, has become a widely adopted and supported standard for SAN connectivity, largely displacing Fibre Channel-based SANs in many shops. iSCSI, normally TCP-based with optional support for RDMA-capable protocols, such as InfiniBand, could be described as a client/server protocol, but that's confusing, as the iSCSI "client" is very often an application "server" in the conventional sense. Hence, iSCSI sticks to the conventional SCSI terms of "target" (the thing that stores the data physically) and "initiator" (the thing that accesses the data over the wire).

As far as the initiator side is concerned, there's a clearly dominant solution in a Linux environment, and that is the open-iscsi Project. Supported by all major distros and iSCSI vendors, open-iscsi, although not flawless throughout its existence, is now ubiquitous and widely accepted as the reference iSCSI initiator on Linux.

For targets, things are a bit more complicated. No fewer than four open-source iSCSI targets currently are available to the Linux storage specialist.

IET, the iSCSI Enterprise Target, is an iSCSI-only, in-kernel implementation of a target that is available as an out-of-tree kernel module only. IET came out of a proprietary iSCSI target implementation from Netherlands-based Ardis Technologies, forked under GPL terms. It enjoys a remarkable proliferation among vendors of cheap iSCSI storage appliances. Several distributions, including SUSE and Debian/Ubuntu, ship IET. Its development is ongoing, but humming along quietly under most people's radar. Scott Walker and Arne Redlich are currently the project's main developers.

With STGT, ex-IET maintainer Fujita Tomonori intended to write an all-userspace, multiple-protocol replacement for IET. It uses only a tiny, generic stub of in-kernel code that made it into Linux at the 2.6.20 release. Everything else about the target happens in userland. Red Hat was quick to adopt STGT as its default iSCSI target during the Red Hat Enterprise Linux (RHEL) 5 release cycle, and continues to support it through RHEL 6. SUSE also has adopted it for SUSE Linux Enterprise Server (SLES) 11, and it also ships in Debian and Ubuntu. However, active development on STGT seems to have largely ceased, and the project is in maintenance-only mode.

SCST, maintained primarily by Vladislav Bolkhovitin and Bart van Assche, is another IET fork with the goal of fixing "all the problems, corner cases issues and iSCSI standard violations that IET has", as the project boldly states on its Web site. SCST has a large and devoted following, and SCST users primarily laud its performance benefits over the competing target implementations. Contrary to STGT and similar to IET, SCST does a lot of its work in the kernel, and its developers have repeatedly submitted the target for inclusion in the mainline kernel. Thus far, however, these efforts have been unsuccessful—at least partially because a fourth target beat developers to it.

That fourth target is LIO, named after the "linux-iscsi.org" domain that the project owns. Its main developer is Nicholas Bellinger. LIO is a generic, in-kernel target driver, of which the iSCSI target is merely one of many front ends. Of the four targets mentioned here, it is the only one not interrelated with any other, and that's not its only interesting trait. LIO uses an unusual in-kernel configuration approach using ConfigFS, which has produced some interesting flame wars on the linux-kernel mailing list in years past. However, in early 2011, LIO beat out SCST to become the blessed replacement of STGT as the upstream kernel's preferred target subsystem. As upstream kernel releases trickle into distributions slowly, some distros already have incorporated this upstream change, but most have not.

DRBD: Storage Replication for the Masses

Any of the four iSCSI targets just mentioned would enable us to build stable, fully open-source, single-instance iSCSI storage on Linux. But that would violate the three Rs of high availability: Redundancy, Redundancy, Redundancy. Storing all of our data just once isn't good enough. We'll need at least two copies of every block we store to remain available even in the face of a storage node failing.

That's where DRBD comes in. Originally the "Distributed Replicated Block Device", it's a synchronous replication facility in the kernel block layer. DRBD provides a virtual block device type (with the LANANA-registered major number of 147) with unique characteristics: any write I/O it receives passes down into the local block layer and simultaneously into the network stack. Over the wire, data replicates to another node—the DRBD peer—where it again passes through the local I/O subsystem and on to directly attached storage. The synchronous nature of DRBD's replication ensures that any writes to the device complete only when completed on both nodes. As such, its replication is fully transparent to applications.

Obviously, if one of the nodes fails, DRBD automatically switches itself into disconnected mode: rather than persisting in its temporarily pointless quest to replicate, it starts recording addresses of out-of-sync blocks in a persistent bitmap. Once the previously failed node recovers, DRBD re-synchronizes the changed data quickly and efficiently.

DRBD's overall overhead is surprisingly

small. Its impact on read performance is practically nil, as reads are local to the host, without any network layer involved at all. For writes, throughput is practically unaffected by a well-tuned DRBD, although some highly latency-critical applications will be adversely affected by running on DRBD.

DRBD has had a somewhat colorful history with regard to its mainline Linux integration. For the better part of its lifetime, the developers chose to maintain the project as an out-of-tree module. Then in mid-2007, it made its first push toward mainline integration, meeting initial rejection from the kernel community—just like many projects with out-of-tree baggage. The developers then embarked on a massive cleanup process, culminating in a final pull request for the 2.6.32 kernel merge window. Following some last-minute opposition, Linus declined to pull DRBD for just that release, and DRBD finally made it into the kernel for 2.6.33 in February 2010.

Since then, the "official" DRBD codebase and the Linux kernel have again diverged, with the most recent DRBD releases remaining unmerged into the mainline kernel. A re-integration of the two code branches is currently, somewhat conspicuously, absent from Linux kernel mailing-list discussions. That situation has caused interesting disparities regarding the state of vendor support for DRBD. Some distributions, like Debian and Ubuntu, currently ship DRBD kernel code as part of their standard kernel packages. Others, like SLES, ship the out-of-tree kernel module version. Still others, like RHEL, do not ship DRBD at all and rely on outside packaging contributions from CentOS and ELRepo. At any rate, to use DRBD, two components are required at a minimum: the DRBD kernel mode itself and a set of userspace utilities and shell scripts that ship as part of the drbd-utils (drbd8-utils on Debian-like platforms) package. On a system where those packages are available, setting up a replicated data set—in DRBD-speak, a "resource"—amounts to a sequence of these steps:

- Set aside a block device for DRBD's local use. DRBD refers to this as its "backing device". This can be an SCSI LUN, a disk partition, an LVM logical volume or any other Linux block device you can possibly imagine.

- Create a resource definition file in /etc/drbd.d.

- Initialize DRBD's metadata.

- Enable the DRBD resource.

- Kick off the initial full-device synchronization to make sure your data sets are in perfect unison (this step can be skipped if the devices are fresh off the assembly line and known to be identical in content).

- Start using the device as any other block device.

Creating a logical volume or partitioning a disk is certainly no challenge for the intrepid reader. Creating a DRBD resource definition, in contrast, might be. DRBD is known for its myriad configuration options, but it also holds some renown for coming with excellent documentation, and simple baseline resource configurations like the following are easy to obtain from the comprehensive DRBD User's Guide:

```
resource vg1 {
  device    /dev/drbd0;
  disk      /dev/sdc;
  meta-disk internal;
  on mike {
    address   192.168.43.111:7788;
  }
  on nancy {
    address   192.168.43.112:7788;
  }
}
```

This configures a resource named vg1 running the DRBD device /dev/drbd0, which mirrors the SCSI device /dev/sdc between hosts mike and nancy. It replicates on TCP port 7788 between the 192.168.133.111 and 192.168.133.112 IP addresses. `meta-disk internal` simply means that DRBD sticks its own metadata into the last few blocks of /dev/sdc. This doesn't mean /dev/drbd0 will be orders of magnitude smaller than the underlying /dev/sdc—DRBD requires only about 32kB of metadata per replicated Gigabyte.

Once the /dev/sdc devices are available on both nodes mike and nancy, and so is the just-created resource definition file, you can continue with initializing the device:

```
drbdadm create-md vg1
v08 Magic number not found
Writing meta data...
initialising activity log
NOT initialized bitmap
New drbd meta data block
successfully created.
success
```

It is necessary to complete this step on both nodes: the metadata that this command creates is local to each individual DRBD node. Likewise, it's necessary to activate the device on both nodes:

```
drbdadm up vg1
```

And finally, on one node only, it is time to kick off the initial device synchronization with a command that sticks out for its somewhat peculiar use of hyphens and whitespace:

```
drbdadm -- --force primary vg1
```

From this point forward, you may use

the device as any other block device, the ongoing background synchronization notwithstanding (this will sound highly familiar to mdadm users).

For most use cases, the next step would be to create a filesystem on the DRBD device. But because DRBD is just another block device, you can treat it as exactly that, and you're essentially free to do anything you would with a "normal", non-replicated block device. In this case, let's create an LVM Physical Volume (PV) signature and a Volume Group (VG):

```
pvcreate /dev/drbd0
vgcreate vg1 /dev/drbd0
```

Once that volume group exists, you can use it like any other—for example, you can create a 10-Gigabyte Logical Volume from it:

```
lvcreate -L 10G -n lun1 vg1
```

And this logical volume, along with the rest of its VG, is now available on whichever node currently has the underlying DRBD resource in the Primary role. You easily can try this out:

```
mike:# vgchange -a n vg1
mike:# drbdadm secondary vg1
nancy:# drbdadm primary vg1
nancy:# vgchange -a y vg1
```

Thereafter, /dev/vg1/test is available as a block device on nancy. The steps you just manually completed are, of course, a cluster manager's task to handle automatically, as I'll explain in a moment.

In combining DRBD and LVM with a Linux iSCSI target, you can create fully replicated, redundant SAN storage in Linux. You need only a cluster manager to tie everything together.

Pacemaker-Based iSCSI Target High Availability

As for any Pacemaker cluster, you first need to set up the underlying Corosync cluster messaging layer. Let's use a configuration that is substantially identical to that in the first article in this series, where I explained the basics of Corosync in more detail:

```
totem {
  # Enable node authentication & encryption
  secauth: on

  # Redundant ring protocol: none, active, passive.
  rrp_mode: active

  # Redundant communications interfaces
  interface {
    ringnumber: 0
    bindnetaddr: 192.168.0.0
    mcastaddr: 239.255.30.11
    mcastport: 5405
  }
  interface {
    ringnumber: 1
    bindnetaddr: 192.168.43.0
    mcastaddr: 239.255.43.0
    mcastport: 5405
  }
}

amf {
  mode: disabled
}

service {
  # Load Pacemaker
  ver: 1
  name: pacemaker
}

logging {
  fileline: off
  to_stderr: yes
  to_logfile: no
  to_syslog: yes
  syslog_facility: daemon
  debug: off
  timestamp: on
}
```

Attentive readers immediately will realize that this example uses multicast addresses different from the ones in my previous article. This is a generally established Corosync best practice: never reuse cluster communications multicast addresses across multiple distinct clusters sharing one host network. Cluster nodes will not join rogue clusters due to mutual node authentication via the Corosync "authkey" shared secret, but they will multicast cluster communications to hosts that aren't supposed to see them. Allocating a new multicast group to each cluster on the network eliminates this issue.

The actual Pacemaker configuration is one that you can, again, either implement on a step-by-step, interactive basis, or you can import it all in one fell swoop, thanks to the crm shell being fully scriptable:

```
primitive p_drbd_vg1 ocf:linbit:drbd \
  params drbd_resource="vg1" \
  op monitor interval="30" role="Slave" \
  op monitor interval="10" role="Master"
ms ms_drbd_vg1 p_drbd_vg1 \
  meta interleave="true" notify="true"
primitive p_lvm_vg1 ocf:heartbeat:LVM \
  params volgrpname="vg1" \
  op monitor interval="30"
primitive p_target_vg1 ocf:heartbeat:iSCSITarget \
  params implementation="tgt" \
    iqn="iqn.2001-04.com.example:example.vg1" \
    tid="1" \
    additional_parameters="DefaultTime2Retain=60 DefaultTime2Wait=5" \
  op monitor interval="10"
primitive p_lu_vg1_lun1 \
    ocf:heartbeat:iSCSILogicalUnit \
  params lun="1" path="/dev/vg1/lun1" \
    target_iqn="iqn.2001-04.com.example:vg1" \
  op monitor interval="10"
primitive p_ip_vg1 ocf:heartbeat:IPaddr2 \
  params ip="192.168.0.100" cidr_netmask="24" \
  op monitor interval="10"
group g_vg1 \
  p_lvm_vg1 p_target_vg1  p_lu_vg1_lun1 p_ip_vg1
order o_drbd_before_vg1 inf: \
  ms_drbd_vg1:promote g_vg1:start
```

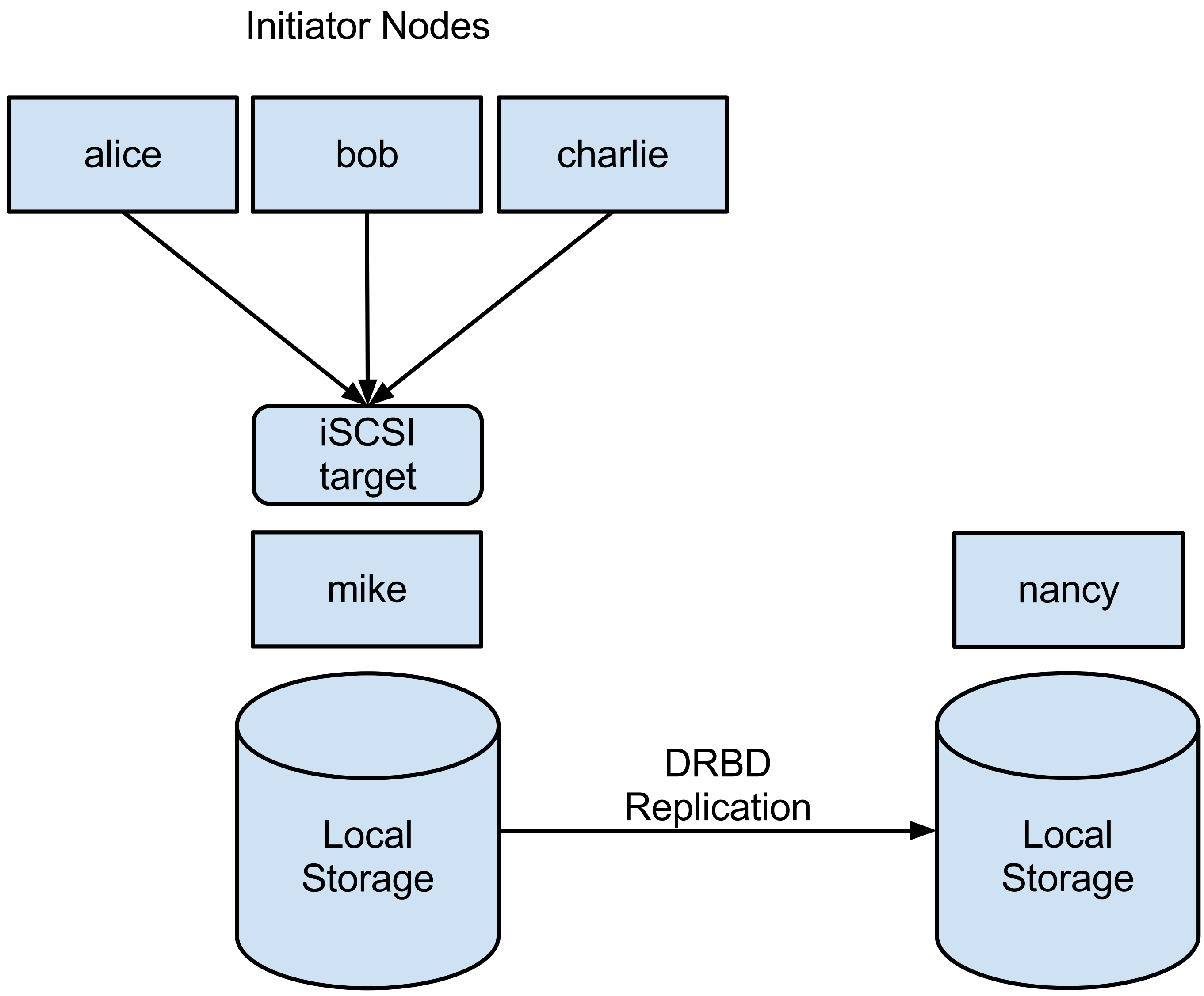


Figure 1. Normal operation: mike hosts iSCSI target; nancy receives synchronous replication stream.

```
colocation c_vg1_on_drbd inf: \
  g_vg1 ms_drbd_vg1:Master
```

This configuration creates an iSCSI target using the STGT implementation—the default on RHEL/CentOS and an available choice on SLES, Debian and Ubuntu—which uses the iSCSI Qualified Name "iqn.2001-04.com.example:vg1". The target contains one Logical Unit with the Logical Unit Number (LUN) 1, and is available through the portal IP address 192.168.0.100 using the default iSCSI TCP port of 3260.

As usual, you can check Pacemaker's status with the crm_mon utility:

```
============
Last updated: Sun Mar  4 21:59:23 2012
Last change: Sun Mar  4 21:58:38 2012 via crm_attribute on mike
```

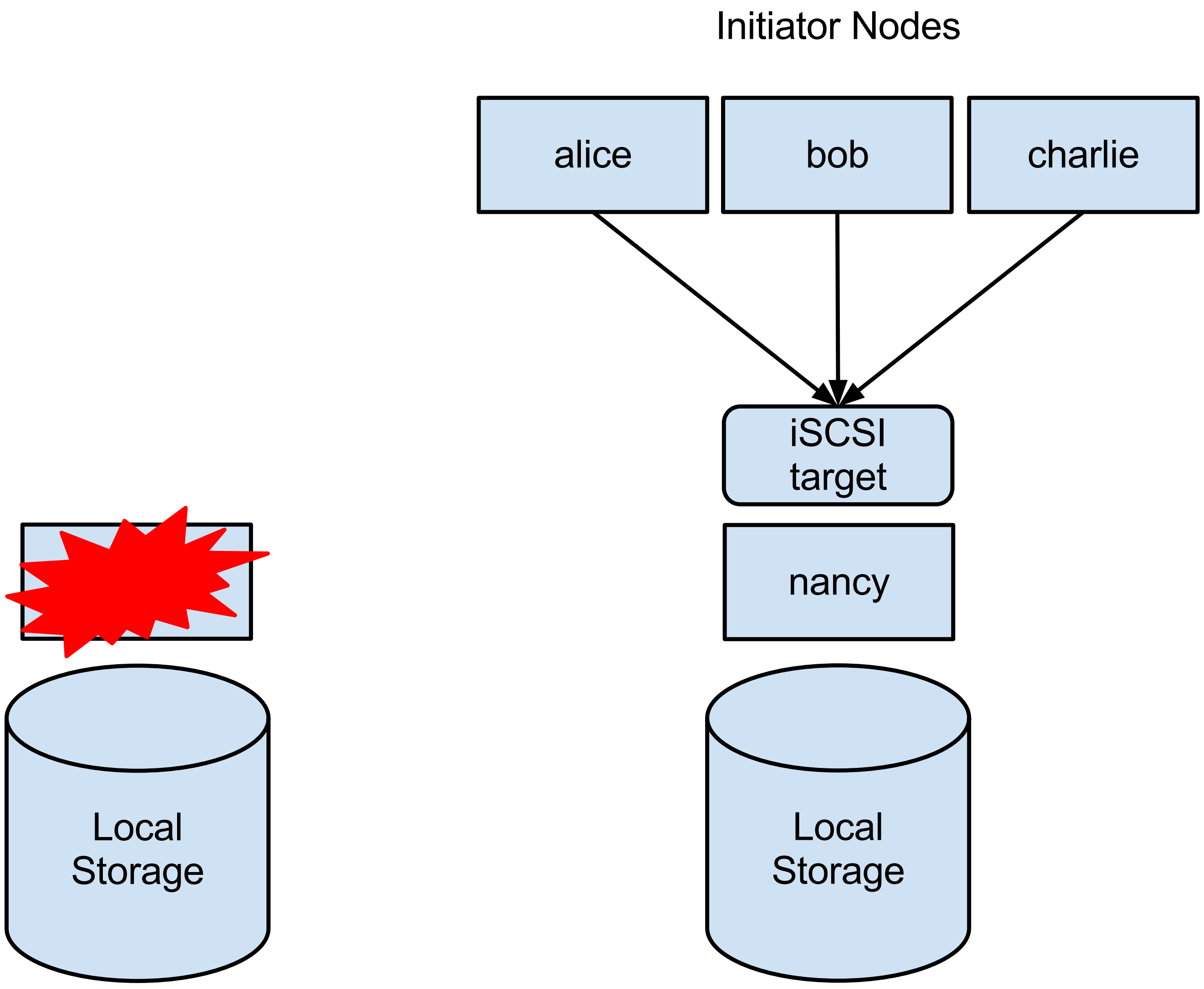


Figure 2. mike has failed; nancy transparently takes over iSCSI target. DRBD replication is suspended.

```
Stack: openais
Current DC: nancy - partition with quorum
Version: 1.1.6-4.el6-89678d4947c5bd466e2f31acd58ea4e1edb854d5
2 Nodes configured, 2 expected votes
6 Resources configured.
============

Online: [ mike nancy ]

 Resource Group: g_vg1
     p_lvm_vg1      (ocf::heartbeat:LVM):  Started mike
     p_target_vg1  (ocf::heartbeat:iSCSITarget):  Started mike
     p_lu_vg1_lun1  (ocf::heartbeat:iSCSILogicalUnit):  Started mike
     p_ip_vg1       (ocf::heartbeat:IPaddr2):  Started mike
 Master/Slave Set: ms_drbd_vg1 [p_drbd_vg1]
     Masters: [ mike ]
     Slaves: [ nancy ]
```

Pacemaker, LVM and DRBD now jointly ensure that this specific target and LUN always are available on one of the cluster nodes. If the node hosting the target goes

Clones and Master/Slave Sets

Besides regular cluster resources (“primitives” in Pacemaker parlance), Pacemaker also supports two advanced resource types: clones and master/slave sets.

Clones are Pacemaker’s “define once, run anywhere” feature. They allow you to define a specific resource—say, a local monitoring dæmon—just once, and then merely set the number of instances of this dæmon that you want to run in the cluster. You define clones such that they “wrap” a primitive. The example below defines a cluster monitoring resource that will run in three incarnations, but only one instance is ever allowed on any node. Thus, if more than three nodes are on-line, Pacemaker can place one of each instance on any three nodes available. Should the cluster fall short of three nodes, Pacemaker would run as many instances as possible, one per node:

```
primitive p_mon ocf:pacemaker:ClusterMon \
  op monitor interval="10"
clone cl_mon p_mon \
  meta clone-max=3 clone-node-max=1
```

Master/slave sets are a special case of clones. In master/slave sets, Pacemaker “promotes” one or more of the clone instances to a master role, with the other instances serving as slaves. What exactly differentiates a master from a slave is entirely application-dependent—Pacemaker simply operates on generic “promote” and “demote” actions. The Pacemaker resource agent to manage DRBD resources is an example of a master/slave set: its Master and Slave roles map to the Primary and Secondary resource roles in DRBD proper:

```
primitive p_drbd_vg1 ocf:linbit:drbd \
  params drbd_resource="vg1" \
  op monitor interval="30" role="Slave" \
  op monitor interval="10" role="Master"
ms ms_drbd_vg1 p_drbd_vg1 \
  meta interleave="true" notify="true"
```

The interleave and notify meta attributes are specific to clones. For more information see the Resources for this article.

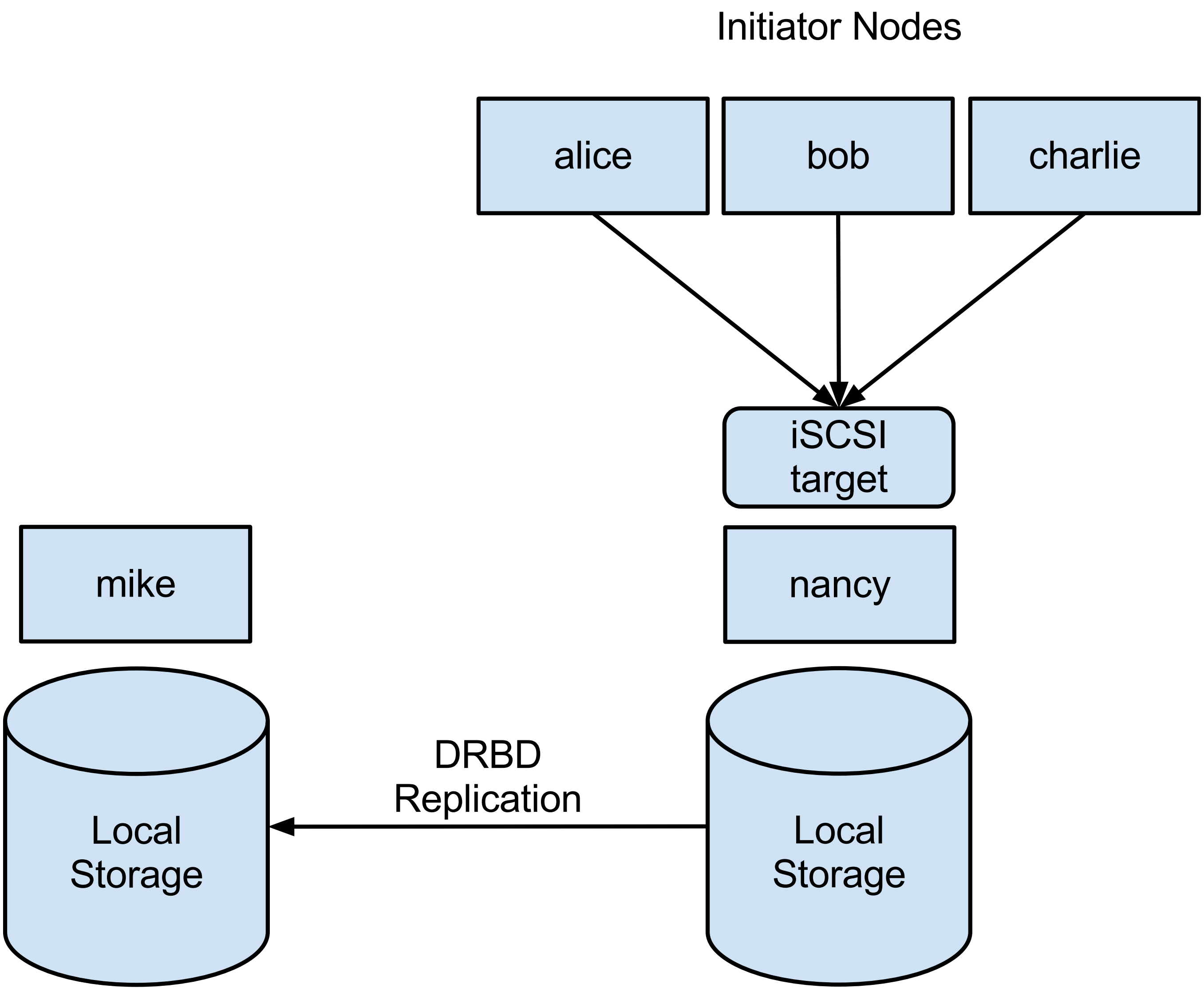


Figure 3. mike has recovered; nancy continues to host iSCSI target. DRBD resynchronizes and continues to replicate in the reverse of its original direction.

down—for reasons of node failure or scheduled maintenance—the entire stack simply shifts over to the other node:

```
============
Last updated: Sun Mar  4 22:01:15 2012
Last change: Sun Mar  4 22:01:12 2012 via crm_attribute on mike
Stack: openais
Current DC: nancy - partition with quorum
Version: 1.1.6-4.el6-89678d4947c5bd466e2f31acd58ea4e1edb854d5
2 Nodes configured, 2 expected votes
6 Resources configured.
============

Online: [ nancy ]
OFFLINE: [ mike ]

 Resource Group: g_vg1
     p_lvm_vg1      (ocf::heartbeat:LVM):  Started nancy
     p_target_vg1  (ocf::heartbeat:iSCSITarget):  Started nancy
```

```
    p_lu_vg1_lun1  (ocf::heartbeat:iSCSILogicalUnit):  Started nancy
    p_ip_vg1  (ocf::heartbeat:IPaddr2):  Started nancy
 Master/Slave Set: ms_drbd_vg1 [p_drbd_vg1]
    Masters: [ nancy ]
    Stopped: [ p_drbd_vg1:0 ]
```

Adding more LUNs to the stack is a simple matter of creating more Logical Volumes in the DRBD-backed VG, then adding additional ocf:heartbeat:iSCSILogicalUnit resources to the g_vg1 group.

Ensuring Seamless Failover Transparency

The iSCSI standard prescribes a per-connection iSCSI parameter, Time2Retain, that governs the length of a connection interruption that the initiator must tolerate. If the target goes away for a limited time—as in the case of a short network hiccup—the initiator simply blocks I/O on the affected iSCSI links. I/O resumes when the target returns. Only if the Time2Retain expires does it drop the connection and flag an I/O error. You can use this feature in building highly available iSCSI targets: as long as the cluster manages to complete failover within the time frame allotted by Time2Retain, the initiators simply resume I/O when the target comes alive on the second node. DRBD, for its part, guarantees through its synchronous replication that the new node has exactly the same data as the old one, so this resumption of I/O is perfectly safe as long as no volatile caches are involved.

The iSCSI initiator and target negotiate the Time2Retain parameter when connecting initially. Both present their preferred value, and the *minimum* presented value wins. Sadly, many iSCSI initiator implementations (the ubiquitous open-iscsi included) set this value to zero by default, thus forgoing a useful high-availability feature built right in to the iSCSI standard. However, they usually present a simple means

Active/Active Clustering

The example configuration in this article is Active/Passive: the iSCSI target portal group is only ever active on one node; the other serves as a standby and DRBD replication target.

Expanding this to an Active/Active configuration is straightforward: dedicate one more SCSI LUN on each host for DRBD replication, and duplicate the example setup with a new DRBD resource, an additional LVM Volume Group, another iSCSI target-portal group, and as many LVM LVs/iSCSI Logical Units as you wish. Pacemaker then has the capability to balance the resource groups between the nodes, improving hardware utilization.

DRBD 8.4 Syntax Changes

DRBD 8.4 users will notice that syntax changes apply to certain commands mentioned in this article. Specifically, in DRBD 8.4 `drbdadm -- --force primary <resource>` turns into `drbdadm primary --force <resource>`. At the time of this writing, DRBD 8.4 users reported stability issues and performance regressions on the relevant project mailing lists; hence, DRBD 8.3 remains the preferred DRBD release for most users.

of re-enabling it—in open-iscsi it's the DefaultTime2Retain parameter in the iscsid configuration file:

```
node.session.iscsi.DefaultTime2Wait = 60
node.session.iscsi.DefaultTime2Retain = 5
```

Setting these configuration parameters enables target failover that is smooth, transparent and almost interruption-free (except for a short, nonfatal I/O hang) to initiators.

The highly available, replicated storage stack I explored in this article is enormously solid, runs in production probably thousands of times over and has wide support from a variety of distributions. However, it does have its shortcomings when there are requirements for multinode replication, wide-area replication and scalability. In the next and final installment of this high-availability series, I will explore alternative approaches to replicated storage that promise to fix these issues.■

Florian Haas is a Principal Consultant and co-founder at hastexo, an independent professional services organization specializing in Linux high availability, storage replication and the open-source cloud. Prior to launching hastexo, he spent four years at Linbit, the company behind the DRBD Project. He is the original author of the DRBD User's Guide and continues to contribute to the project; however, neither he nor his company are affiliated or associated with Linbit.

Resources

IET Project Web Site: **http://iscsitarget.sourceforge.net**

STGT Project Web Site: **http://stgt.sourceforge.net**

SCST Project Web Site: **http://scst.sourceforge.net**

LIO Project Web Site: **http://www.linux-iscsi.org**

DRBD Community Web Site: **http://www.drbd.org**

DRBD User's Guide (DRBD 8.3 version): **http://www.drbd.org/users-guide-8.3**

"Clones" (Pacemaker Configuration Explained): **http://www.clusterlabs.org/doc/en-US/Pacemaker/1.1/html/Pacemaker_Explained/s-resource-clone.html**

"Options" (Pacemaker Configuration Explained): **http://www.clusterlabs.org/doc/en-US/Pacemaker/1.1/html/Pacemaker_Explained/ch10s02s02.html**

"Notifications" (Pacemaker Configuration Explained): **http://www.clusterlabs.org/doc/en-US/Pacemaker/1.1/html/Pacemaker_Explained/ch10s02s08.html**

Rocking Charlotte:

- Build An Open Source Cloud Day
- LPI Exam Cram
- MySQL Training Day
- Java Hack-A-Thon
- Open Database Camp
- Drupal Camp/ Drupal in a Day
- BSDA Certification
- LPI Certification
- PuppetLabs Training
- SaltStack Training

Come Play with Rock Stars

DOC SEARLS

Freedom and Identity

It's time for the former to liberate the latter.

Two big things happen for me this month (May 2012). One is that my first book comes out. (That is, the first one written by me alone, with no co-authors.) The title is *The Intention Economy: When Customers Take Charge*. The publisher is Harvard Business Review Press, and the publication date is May Day. The other is IIW, the Internet Identity Workshop in Mountain View, California. It's also on May Day, plus the two days following. This will be our fourteenth IIW (we have two each year). Like the others, it will be an unconference, long on conversation, collaboration and hacking, and short on BS (no speakers, no panels, no keynotes, no booths). And I'm sure it will be, as all of them are, better and more productive than the last one.

There is a thread that runs between these two things—the book and identity—going back to 1994, when Phil Hughes began vetting the idea of doing a magazine about free software. That magazine is the one you're reading now.

Since the beginning of that thread, two things have been clear. One is that freedom and identity are both personal. The other is that neither are simple. "Free will does not mean one will, but many wills conflicting in one man", Flannery O'Connor wrote (in *Wise Blood*). "Freedom cannot be conceived simply." Same for identity. Writes Michael Ventura (in *Shadow Dancing in the USA*):

> ...there may be no more important project of our time than displacing the fiction of monopersonality. This fiction is the notion that each person has a central and unified "I" which determines his or her acts. "I" have been writing this to say that I don't think people experience life that way. I do think they experience language that way, and hence are doomed to speak about life in structures contrary to their experience.

We hold these identity-defining contexts like cards in a hand, and we play them all differently.

The "fiction of monopersonality" struck home for me when I read that passage in the early 1980s. Who I was to my mother, my wife, my kids, my readers, my co-workers—and even to my self in different settings—was different. What we call sanity is unifying all our different identities behind a first-person pronoun: "I", "me", "my" or "mine". Even our names are more varied than those. In my own case I am "Doc" to some people, "David" to others, "Dave" to others, "Searls" to others, and "dsearls" or "@dsearls" on the Web. In Junior High, I was "Sleepy". (Thus, I have shared nicknames with two of the Seven Dwarfs.) Ventura's point is not just that monopersonality is fiction and that we experience life and language differently, but that who we are is context-dependent, and there are many of those, most of which take the form of relationships with others, all of which come and go—and change along the way.

I recently listened to a tape of a conversation I had with my mother many years ago. I found myself wishing that I could not only talk to her again, but also that I could again be who I was with her. But I couldn't, because that part of me—the son-self in my portfolio of polypersonal identities—died when she left the planet in 2003. I was already in my fifties by then, but I was still her son: to her, and to me. My father, with whom I also had a fun and loving relationship, died in 1979, half my life ago. So I am now nobody's son.

My wife and I have wedding rings inside each of which is inscribed "the couple decides". Who I am as a husband is a member of a relationship that has an identity of its own, but one sustained by two sovereign selves.

We hold these identity-defining contexts like cards in a hand, and we play them all differently. Where it gets complicated is with the cards—often literal ones—that are loaned to us. These are what Devon Lofretto (posting on the ProjectVRM list) calls "administrative" identities. In my wallet I have administrative identities from the State of California, United Airlines, Harvard University, the Automobile Club of Southern California, VISA, Costco/American Express, Chase Bank and Blue Cross. Not in my wallet, but in my possession, are other administrative identities from Hudson County, New Jersey (where I was born), UC Santa Barbara and dozens of retail establishments. If you count the hundreds of login/password combinations recalled by my browsers, I total several hundred administrative identities in all. None of them are who I am in

Solving the conflict between sovereign source and administrative identities has been the primary challenge for digital identity development from the start.

the deepest and most original sense: the one that was born free. That is, none of them are what Devon calls my "sovereign source" identity, because the administrator of that identity provides the whole context, starting with what they call me, which often aligns with what my parents named me, but doesn't always. In either case, the a priori responsibility is theirs, not mine. We are not a couple, and the couple can decide only what the administrative party alone allows it to decide.

Solving the conflict between sovereign source and administrative identities has been the primary challenge for digital identity development from the start. Because we are so accustomed to being a zillion namespaces in a vast administrative sphere (now as large as the Web itself), we acquiesce to the insanity of it. In fact, we actually mistake the normative nature of this insanity for its opposite. That's why we keep trying to come up with ways to manage multiple logins and passwords—or to create yet another administrative system (the largest of which today are Google, Facebook and Apple)—rather than to replace the whole crazy system with one that builds upward from our sovereign source identities.

Albert Einstein said no problem can be solved at the level of consciousness that created it. That means we can't meet the challenge of fully enabling sovereign source identity at the administrative level. It just can't be done. There is no administrative answer to the identity mess.

Yes, we do need administrative compliance to whatever answers we come up with. But our answers need to come from our free, sovereign and independent selves. Devon lays out our challenge this way: "How can you administer an identity ecosystem if you cannot properly define sovereign source authority? Understanding the structural power of John Hancock should be a job requirement in the American government."

We may not have defined "sovereign source" yet, but we have experienced it, and that helps. Wrote Walt Whitman:

You make too much of articulation.

Encompass worlds but never try to encompass me.

I crowd your noisiest talk by looking toward you.

Writing and talk do not prove me.
I carry the plenum of proof and everything else
in my face.
With the hush of my lips I confound the topmost skeptic.

Also:

I know this orbit of mine cannot be swept
by a carpenter's compass,

I know that I am august.
I do not trouble my spirit to vindicate itself
or be understood.
I see that the elementary laws never apologize.

Strong stuff. But can we make that into what the wonks call "policy"?

No, we can't, because the wonks' answer will be an administrative one. But most of us here aren't wonks. Instead, we write code. Here in Hackerland, we change practice, not law.

When John Hancock signed the *Declaration of Independence*, he was declaring the independence of a nation with a signature that also declared his sovereign self. He presented his plenum and proof in script several decades before Whitman declared his own in verse.

My goal for the last six years has been to change practice through inventions that mother necessity. The practice I've worked to change is how you and I relate to administrative entities—especially vendors. I started ProjectVRM at Harvard's Berkman Center in 2006 to encourage development of tools that made individuals independent of administrative entities—and better able to engage with them, in our own ways, and on our own terms.

For example, I see no need today for the government to create "do not track" laws, or to mandate "no track" buttons on Web sites (both of which are policy solutions), when we also can create easy ways for individuals to specify "do not track" as a binding obligation in a relationship with any entity. I can say this with confidence because there are now more than 30 development projects listed on the ProjectVRM Wiki (**http://cyber.law.harvard.edu/projectvrm**), all providing code or services that give individuals their own ways to engage with other parties.

Nearly all those projects are flying under the radar of both the mainstream media and VCs, both of which remain in thrall of vendor sports (for example, Google vs. Facebook) and continue to think that the social matters more than the personal. But that won't last long. In a post titled "The Nature of the Firm and Work Markets" (**http://www.avc.com/a_vc/2012/03/the-nature-of-the-firm-and-work-markets.html**), the VC Fred Wilson wrote:

> In *The Nature of the Firm*, Coase investigates why "individuals choose to form partnerships, companies and other business entities rather than trading bilaterally through contracts on a market."
>
> Coase argues that transaction costs that make "trading bilaterally through contracts" expensive spur the organization of firms. And if those transaction costs could be eliminated, more individuals would choose to trade with each other rather than forming partnerships, companies and other business entities.
>
> Enter the Internet and having a computer in your pocket into this model and things change. Technology has been causing these transaction costs to drop precipitously for years now and the result is we have seen the emergence of work markets in which "individuals trade bilaterally through contracts".
>
> Our firm is seeing these work markets sprout up all around us and if there is a single investment theme that is dominating our deal flow right now, this would be it.

In fact, Fred's firm, Union Square Ventures, is already invested in at least one VRM company (**http://Getabl.com**). In that same piece, Fred sources a post by his colleague Christina Cacioppo (**http://www.usv.com/2011/11/what-comes-next.php**). She writes:

> One reason to create firms is the coordination and signaling problems of situations with imperfect information and transaction costs. As technology increases information flows and decreases transaction costs, individuals can leave their old employers and strike out on their own. Their livelihoods will still depend on providing valuable services in exchange for fees, but they'll do so as freelancers—and on their own, they'll capture more of the value generated by their work....
>
> These free agents, disaggregated and newly empowered, can promote and sustain themselves with new tools....

Sound familiar? How many *Linux Journal* readers (especially ones who write code) began doing this long ago? Christina also visits the subject of identity:

> Between identified, liberated individuals and the nameless, faceless drones of Mechanical Turk lies identity: does it matter who performs the task at hand? If the worker's background, skills, or experience matter, there's likely to

Advertiser Index

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

ADVERTISER	URL	PAGE #
ACENET	http://www.ace-host.net	15
DRUPAL BUSINESS SUMMIT VANCOUVER	http://www.drupalsummit.com/city/vancouver	33
EMAC, INC.	http://www.emacinc.com	45
EMPERORLINUX	http://www.emperorlinux.com	37
IXSYSTEMS, INC.	http://www.ixsystems.com	7
LINARO CONNECT Q2.12	http://connect.linaro.org/register	17
LOGIC SUPPLY, INC.	http://www.logicsupply.com	48, 59
MICROWAY, INC.	http://www.microway.com	94, 95
O'REILLY FLUENT CONFERENCE	http://fluentconf.com/fluent2012	105
SILICON MECHANICS	http://www.siliconmechanics.com	3
SOUTHEAST LINUXFEST	http://www.southeastlinuxfest.org	119
USENIX FEDERATION CONFERENCE WEEK	https://www.usenix.org/conference/fcw12	53

ATTENTION ADVERTISERS

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

> be higher variance in demand for a particular person's services, and free agents will be sought after and chosen by reputation on services built for those purposes. Less-skilled people are likely better suited for tasks for which identity doesn't matter, and other marketplaces that don't include a concept of reputation will provide access to a global pool of workers.

In other words, there is a growing distinction between dronework by faceless sources of simple labor and distinctive constructive work by truly free (and faceful) agents. This post is for the latter.

To get full respect for the free agents that all of us were born as, the freedom-loving hackers among us need to help build the tools that give our sovereign sources full authority over the administrative namespaces currently managed by second and third parties. (We're the first parties here.)

A number of these tools are already in the works, and many of them are open source. Singly's Locker Project (**http://lockerproject.org**), for example, is an open-source personal data locker on which apps are already being built. Individuals (or their friendly programmers) can write rules for managing relationships through countless potentially interrelated events on the Net, using KRL (kinetic rules language), written by Phil Windley (**http://windley.com**). KRL and its rules engine are both open source (at GitHub). On the legal front, Joe Andrieu, Judi Clark and Iain Henderson (along with more than 40 others) have been working on an Information Sharing Agreement (**http://kantarainitiative.org/confluence/display/infosharing**) that treats sovereign and administrative identities as equals from the start. The group and its work are both open. Drummond Reed and colleagues at Connect.me have been working on the Respect Trust Framework (**http://connect.me/c/trust**)—built around freely made assertions by sovereign individuals regarding the trustworthiness of others—which is listed with the Open Identity Exchange (OIX) and hosted by the Personal Data Ecosystem Consortium. That's open too. (That is, if others want to use the same framework, they can. Individuals' data is also their own.)

These developers will be at IIW. Same will go, I hope, for the VCs I just mentioned as well. (Christina and her colleague Brad Burnham were both at the last IIW.) So, if you're up for raising the Barn of the Sovereign Self with the rest of us, come give us even more to celebrate on May Day.■

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.