Sinatra | HTML5 | Catalyst | Drupal 7 | Moodle

LINUX JOURNAL™

Since 1994: The Original Magazine of the Linux Community

INTRO TO THE SINATRA MICRO-FRAMEWORK

THE BEST WAY TO CLIENT CONCURRENCY

FEBRUARY 2012 | ISSUE 214 | www.linuxjournal.com

WEB DEVELOPMENT

CREATE Web Apps with Catalyst and Perl

AUDIO SOLUTIONS with HTML5

Use **DRUPAL 7** for Basic Web Design

INTERVIEW WITH STEPHEN WOLFRAM of Mathematica and Wolfram|Alpha

+ NEW COLUMN: THE OPEN-SOURCE CLASSROOM

SILICON MECHANICS

visit us at www.siliconmechanics.com
or call us toll free at 866-352-1173

Silicon Mechanics Announces Research Cluster Grant

We are pleased to announce our sponsorship of a unique grant opportunity: a complete high-performance compute cluster using the latest AMD Opteron™ processors and NVIDIA® GPUs.

This grant program is open to all US and Canadian qualified post-secondary institutions, university-affiliated research institutions, non-profit research institutions, and researchers at federal labs with university affiliations.

To download the complete rules, application, and hardware specification, visit
www.siliconmechanics.com/research_cluster_grant
or email
research-grant@siliconmechanics.com

Silicon Mechanics would also like to thank the many hardware partners that have made this grant possible.

Kingston TECHNOLOGY · OPTERON PROCESSOR AMD · NVIDIA TESLA · QLOGIC · SUPERMICRO · Seagate

When you partner with Silicon Mechanics, you get more than affordable, high-quality HPC — you get a team of Experts dedicated to the advancement of scientific research.

Ocean Row Solo Expedition Update

Wave Vidmar has adjusted his schedule. He will be shipping “Liberty” to Portugal for a February launch to row the North Atlantic East-to-West route, heading for an area north of the Caribbean Islands. He is currently planning to undertake the West-to-East Row in May. We will continue to follow his expedition at siliconmechanics.com/ors.

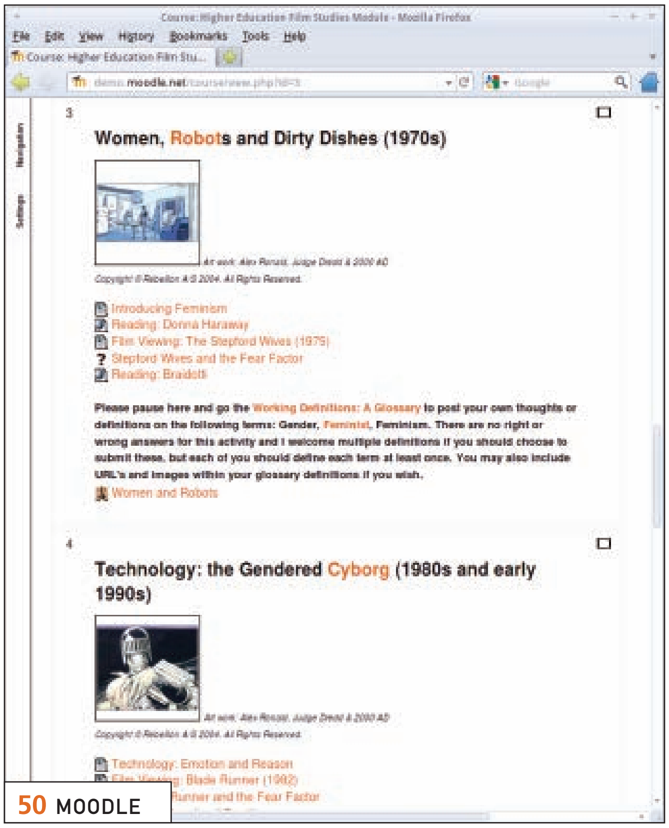
Expert included.

CONTENTS

FEBRUARY 2012
ISSUE 214

WEB DEVELOPMENT

FEATURES


Course: Higher Education Film Studies Module - Mozilla Firefox

3

Women, Robots and Dirty Dishes (1970s)

Art work: Alex Ronald, Judge Dredd & 2000 AD
Copyright © Rebellion A/S 2004. All Rights Reserved.

Introducing Feminism
Reading: Donna Haraway
Film Viewing: The Stepford Wives (1975)
Stepford Wives and the Fear Factor
Reading: Braidotti

Please pause here and go the Working Definitions: A Glossary to post your own thoughts or definitions on the following terms: Gender, Feminist, Feminism. There are no right or wrong answers for this activity and I welcome multiple definitions if you should choose to submit these, but each of you should define each term at least once. You may also include URL's and images within your glossary definitions if you wish.
Women and Robots

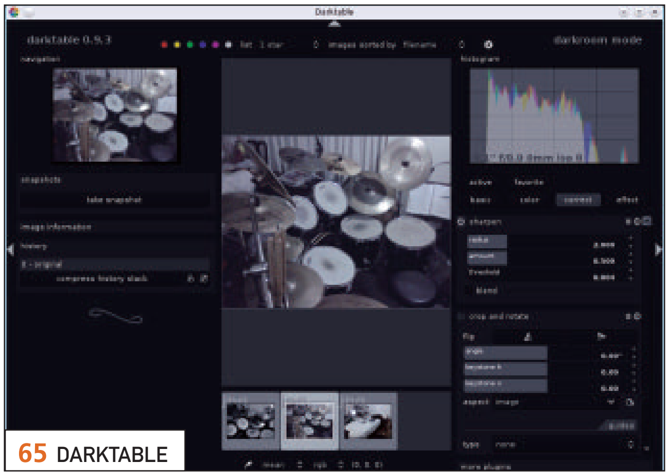
4

Technology: the Gendered Cyborg (1980s and early 1990s)

Art work: Alex Ronald, Judge Dredd & 2000 AD
Copyright © Rebellion A/S 2004. All Rights Reserved.

Technology: Emotion and Reason
Film Viewing: Blade Runner (1982)
Runner and the Fear Factor

50 MOODLE



65 DARKTABLE

ON THE COVER
- Intro to the Sinatra Micro-Framework, p. 30
- The Best Way to Client Concurrency, p. 82
- Create Web Apps with Catalyst and Perl, p. 68
- Audio Solutions with HTML5, p. 92
- Use Drupal 7 for Basic Web Design, p. 100
- Interview with Stephen Wolfram of Mathematica and WolframlAlpha, p. 108
- New Column: The Open-Source Classroom, p. 50

COVER IMAGE: © Can Stock Photo Inc. / thesupe87

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

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.

TrueNAS™ 2U Appliance: You Are the Cloud

Storage. Speed. Stability.

With a rock-solid FreeBSD® base, Zettabyte File System (ZFS) support, and a powerful Web GUI, TrueNAS™ pairs easy-to-manage FreeNAS™ software with world-class hardware and support for an unbeatable storage solution. In order to achieve maximum performance, the TrueNAS™ 2U System, equipped with the Intel® Xeon® Processor 5600 Series, supports Fusion-io's Flash Memory Cards and 10 GbE Network Cards. Titan TrueNAS™ 2U Appliances are an excellent storage solution for video streaming, file hosting, virtualization, and more. Paired with optional JBOD expansion units, the TrueNAS™ System offers excellent capacity at an affordable price.

For more information on the **TrueNAS™ 2U System**, or to request a quote, visit: **http://www.iXsystems.com/TrueNAS**.

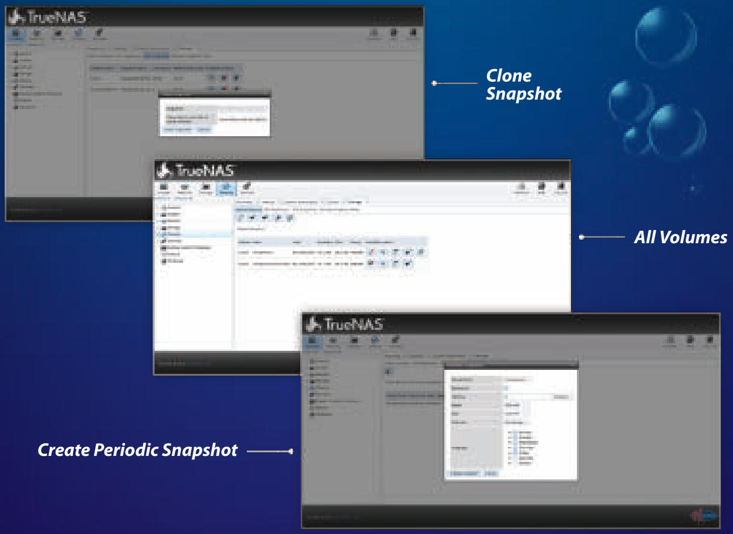
FreeNAS 8 POWERED

KEY FEATURES:

- Supports One or Two Quad-Core or Six-Core, Intel® Xeon® Processor 5600 Series
- 12 Hot-Swap Drive Bays - Up to 36TB of Data Storage Capacity*
- Periodic Snapshots Feature Allows You to Restore Data from a Previously Generated Snapshot
- Remote Replication Allows You to Copy a Snapshot to an Offsite Server, for Maximum Data Security
- Software RAID-Z with up to Triple Parity
- 2 x 1GbE Network interface (Onboard) + Up to 4 Additional 1GbE Ports or Single/Dual Port 10 GbE Network Cards

JBOD expansion is available on the 2U System

* 2.5" drive options available; please consult with your Account Manager



iXsystems®

Intel Xeon inside™ Powerful. Intelligent.

Call iXsystems toll free or visit our website today!

1-855-GREP-4-IX | www.iXsystems.com

SHAWN POWERS

More Than Dark Rooms and Red Lights

Developing photographs used to be a mysterious process that took place in rooms lit like horror films, using pans of dangerous chemicals. From those creepy rooms eventually emerged beautiful photographs of literally everything under the sun. Web development is surprisingly similar. If you replace the dangerous chemicals with a clacking keyboard and the red lights with the blue glow of an LCD screen, they're just about the same. This month, we delve into the mystical world of Web development, which turns ordinary tags and scripts into the beautiful Web sites we visit every day.

Reuven M. Lerner is right at home with our focus this month and shows us Sinatra. No, he doesn't teach us to be crooners; instead, he demonstrates the micro-framework that makes small Web apps a breeze to create. If you need to do something small, and Ruby seems like overkill, give Sinatra a try.

Dave Taylor explains how to make a small program this month as well, but his script is a little more nefarious. If you want to be a lying, cheating dog when you play *Scrabble* against your friends, Dave can help you out. He shows how to make a shell script that finds words based on the letters you have. As a purely educational endeavor, it's a great article, but as of now, I will no longer play *Words With Friends* with any *Linux Journal* readers!

Kyle Rankin continues his series on GPU-powered password cracking. Even if you're not interested in learning to brute-force attack passwords, I recommend reading his article. It will convince you to use a strong password quicker than any warning I might give you. Don't tell Kyle I told you, but I watched him enter his password once at a conference. It's just a bunch of asterisks!

I got into the Web-themed spirit this month too. This month we're launching a brand-new column, written by yours truly. The Open-Source Classroom is an education-focused column, and this month, I break down the venerable Moodle software package. Moodle has matured so much since we last covered it, I felt it deserved some attention.

Ruby on Rails is a very popular Web framework, but Henry Van Styn knows that it's not the only show in town. He

As a purely educational endeavor, it's a great article, but as of now, I will no longer play *Words With Friends* with any *Linux Journal* readers!

describes how to develop Web applications with Catalyst and Perl. Regardless of the framework you use to develop, an application is only as good as the server hosting it. Martin Kalin shows us the nitty-gritty of Web servers and explains multiprocessing, multithreading and evented I/O. The Internet demands a lot from our Web servers, and Martin helps us understand exactly how those demands are met.

Possibly bigger than even the Web 2.0 buzz of a few years ago is the transition to HTML5. Although to most folks it's just a bunch of geek jargon, for Web programmers it means user interaction like never before. Paul Freitas shows how to take advantage of HTML5 to serve up audio. No, we don't mean those annoying MIDI files playing when the page loads, but rather using plain-old HTML tags to play audio on your Web site. If it sounds too simple to be true, you'll want to read Paul's article.

What if you want to develop a Web site, but you're not a programmer? That's where content management systems like Drupal come into play. Kenneth P. J. Dyer walks through designing a Web site with Drupal 7. The learning curve for using Drupal traditionally has been pretty steep, but with version 7, that's starting to change. Kenneth teaches how things work in Drupal and gets us started on our way to developing a powerful and flexible Web site.

We never alienate our non-developer readers in issues like this, and this month is no different. We have tons of UpFront articles, product announcements and even some handy tips to make your Linux life a little easier. In fact, this month James Gray had an opportunity to interview Stephen Wolfram. Yes, *that* Stephen Wolfram. Whether you're a math nut who loves Mathematica or are fascinated by the Wolfram|Alpha computational knowledge engine, this is an interview you won't want to miss.

So until next month, keep clacking those keyboards. If turning on a red light will help you develop Web sites, by all means, feel free to do so. The only dangerous chemical we recommend, however, is caffeine. If you want to drink your coffee out of developing trays, well, I guess that's up to you. Either way, we hope you enjoy this issue! ■

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

Digital Edition for the Kindle

Congratulations on going all digital, and thank you for making it available for the Kindle. I live in Poland, and the digital edition made it possible for me to subscribe and read your magazine for a very fair price. I've one suggestion for the Kindle edition: it would be great if it had a table of contents available under directional buttons (like it has for other magazines in Kindle format—see the ones generated by Calibre, for example). As it is now, switching between articles is painful, because you have to go back to the first page. Apart from this, well done. I hope this step will be a great success for *Linux Journal*!
—Pawel Stolowski

Pawel, thanks for the suggestion! The epub/mobi edition is still new for us, and we're constantly trying to improve it. We appreciate the recommendations, and we'll do our best.—Ed.

Will Not Resubscribe

I'm not going to renew my subscription once it expires due to the fact that *LJ* went all digital. I absolutely abhor it. Even though I've downloaded the latest issues since the conversion, I haven't read any of them. I just lost complete interest in what was a magazine I anticipated and loved to read every month. I will not resubscribe again until you do on-demand printing.

I haven't used MagCloud's services, but at some point, I'm going to try it when I decide to go ahead with my own magazine (**http://www.magcloud.com**). Perhaps *LJ* should consider a print-on-demand option.

So, yeah, sorry, but I seriously cannot convey in words how much I hate this all-digital conversion. I abhor it with every fiber of my existence. It's a shame such a legendary magazine has decided to screw things up. E-publishing sucks.
—Patrick Baker

Patrick, the response to our switch has met both accolades and disdain. I do honestly appreciate the desire for a physical magazine, and your MagCloud recommendation is actually pretty cool. Although we don't have any current plans to offer a print-on-demand subscription, it certainly seems individuals could print their own. This is especially true since your example at MagCloud has no minimum *print run. Heck, I might try to print one just to see how it looks!—Ed.*

Digital Format Forces Some Personal Changes

Linux Journal's abandoning paper is taking some getting used to and will require some personal changes if I am to continue to enjoy my favorite magazine. I used to keep may latest copy next to my favorite chair and would read it in bits and pieces and during commercials on TV. Now the issues go mostly unread because I don't like reading magazines or books on my desktop or laptop. Today, I took my last paper copy to the gym to read while using the elliptical exercise machine (there is a book holder on the machine). This was very workable, but then it dawned on me that I cannot do this with the digital version unless I purchase a Kindle or other E Ink equivalent. Your conversion to digital media is going to cost me at least $79 and force some behavior changes about which I am not too happy. That said, *Linux Journal* is my favorite magazine, and I am willing to invest in making it more readable for me. Maybe you should do a deal with Amazon and sell Linux-branded Kindles!

—Edward Comer

Edward, although Linux Journal *certainly can be read on a desktop or laptop, I must admit an e-reader is my favorite way to read it. There are advantages and disadvantages to both E Ink devices and color ones, but for a traditional paper "feel", you are correct, E Ink is hard to beat. And, as far as a Linux-branded Kindle? I'm not sure Amazon would go for it, but because the Kindle runs Linux, we'll claim the victory anyway.—Ed.*

The Web Is Not Paper

HTML should be the format for Web publications. One of the first things a Web page designer learns is that Web pages are not paper. There is no need to fill a Web page like a sheet of paper. Web pages can have open space and uncluttered design with no cost. Paper is expensive; pixels are not.

One should be able to scroll through your entire magazine with the mouse wheel and an occasional click on a link. The PDF format itself is bad enough, but laying out the pages side by side makes it worse. If you stacked single pages, it would be easy to read even a PDF file by a continuous scroll down with the mouse wheel.

If you feel a need to have narrow columns like a paper magazine, don't think paper and put them side by side—have one continuous column with a larger font size. On paper, narrow columns have a purpose, but on the Web, they are just a nuisance. The computer screen is rarely both wide enough and tall enough to display a single magazine page at a decent font size. This forces the reader to scroll both up and down and side to side instead of smoothly scrolling in one direction.

It would be even more readable if you would place two or three paragraphs per Web page—at most—with links at the top and bottom to the previous and next page as well as back to an index. Then, readers could click from page to page and hop around like they can with a paper magazine. To avoid constant trips back to the server, each article could be a single download with links and whitespace, making it appear to be separate pages.

Your content is interesting, but please try delivering it in several styles and let your readers evaluate the readability of each choice.
—Jim Fuqua

Jim, I assume you're referring to the online interactive version of Linux Journal*? It (along with the PDF) is attempting to maintain a traditional magazine feel. For some folks, that's very important. In fact, some folks print the PDF, so in that case, the traditional layout really is a benefit.*

We do have epub and mobi versions available, which are not HTML, but they do offer flowing text and a much less traditional layout. One thing the digital transition has allowed us to do is offer the same information in a multitude of formats. Hopefully, one of them will fit your needs.—Ed.

LJ Shines in Digital Format

Switching to digital format was a great idea not only for you but for us, the readers. First, I did only very rarely purchase magazines, because they tend to pile up, and most issues either get lost under the sofa or shredded by my resident pets. Second, I now can carry *LJ* in my pocket, so it is always available for all those short breaks either in the bathroom (we all do that) or waiting for my significant other to try on each color and size of the same T-shirt at the mall. Having said that, it still needs improvements. The low resolution of the iPad version is a real problem. Most screenshots are almost unreadable. Also, the content could be improved with media, such as videos and downloadable code snippets. Interactivity is the key for the future. Being a paper magazine on a digital device will not be enough in a very near future. When an article mentions an event, I expect an "add to calendar" link, for example. Still, my annual subscription was the best bucks spent ever. Keep up the great work!
—M. Belgin

Those are some great ideas, thank you. We currently are working on the *screenshot and code issues. With tablets and phones being so different, coming up with a layout that looks best on all devices is a real learning experience.*

The interactive stuff is partially in place with the on-line version of the magazine. We will keep trying every month to improve the digital offerings. It's actually pretty exciting. Because the e-publishing world is so young, I think we'll see some amazing things in the coming years.—Ed.

Digital Publishing

In the December 2011 issue's Letters, Valentin asked that *LJ* write about digital publishing, because that is the only format available now. The editor's response was that he would like to get some experts to write some articles, but I think he missed the point. You (*LJ* staff) are the experts. You should write about the process you use to create an issue, why different decisions are made and the issue with the various formats. It will be informative for us subscribers, and perhaps by writing things down, you'll gain insight into the process.
—bil jeschke

Perhaps I wasn't clear—my apologies. I meant we'd like to get articles from experts using specific software for manipulating and creating digital material. Although publishing is certainly in our wheelhouse, we're the first to admit we don't know everything about everything. Contributions from the community are what make the Open Source world, and our publication itself, so great. Yes, our staff is part of that community, but we don't want to limit information to what we know.—Ed.

Digital Edition

I have been a newsstand subscriber for years, enjoying thumbing through the magazine at my local Safeway while my wife does her shopping and taking the magazine home with me for sharing with friends as a coffee-table "ice-breaker". Sadly, this past year I decided to pay for a real subscription and have it delivered to my door, only to have the print edition done away with halfway through my subscription. To this day, I have not read a single digital edition. My subscription ended in November, and I didn't even notice it. How sad, considering the years I have spent reading the magazine. I understand dead-tree editions may cost money, but I always assumed that was covered by the advertisers in the magazine, of which yours is plenty filled with!

I have an Android phone, with a good-size screen, but I do not have a Kindle or a NOOK, nor am I really interested in spending the money on one. I enjoy holding physical magazines or books, and reading them or looking through them while relaxing at my desk (and, of course, throne-room time).

I have long loved your magazine, and

although I am not and never have been an Ubuntu guy, I still look forward to the magazine. However, I feel a great disturbance in the force with the loss of a good magazine. Oh, how many conversations have been struck up due to holding the magazine while standing in a checkout line! I feel as though I wasted my year's subscription, since I never cared to read a digital edition. My money would have been better spent on *Wired* or some other magazine that still prints.

Maybe I will have to buy a NOOK, but I cannot justify spending $200 to read a $3 magazine. I wish you the best, however much longer it lasts.
—Nathan England

Nathan, I could give you the ad-to-content ratio of our magazine compared to others, showing our passion for filling pages with content and only limited, relevant ads. But really, what a boring discussion that would be. We had to switch for reasons we've highlighted during the past few months, so for better or for worse, we're 100% digital.

I am curious, however, if you've tried the several ways to consume Linux Journal *on your phone. The Linux Journal app for the Android will keep your library automatically up to date. As an e-reader, your phone also can display the epub version using a variety of programs. I'm the first to admit reading the PDF on a smartphone is a recipe for frustration, but perhaps one of the other formats will be more comfortable on your phone? And, of course, if you prefer the traditional paper look, an E Ink-based reader like the $79 Kindle is always a possibility.—Ed.*

At the Forge—December 2011 Issue

I was quite dismayed by Reuven M. Lerner's comment "I hope you're not really using a queue class but rather just using arrays". Not only does this contradict the seminal research and arguments for using abstractions in the first place (see Dijkstra, Hoare, Wirth and many others), but it has the immediate practical problem of polluting the code with functions that will not make sense should the implementation of queues have to be changed, even slightly.

It is almost axiomatic that someone will get bitten when code is written that depends on implementation details.
—David Jameson

Reuven M. Lerner replies: *I wasn't trying to say that people should avoid writing their own abstractions that wrap native data types. Rather, I was saying that if all you need is the simple implementation that I showed, you should just use a Ruby array, rather than what amounts to a wrapper around the array, which offers no additional functionality.*

Wrappers are great, and they do provide a large degree of safety and abstraction that can be useful if the implementation changes. But if you just need an array, there's no need to wrap it in a class just in case you might eventually need additional functionality.

_LJ_ Electronic Magazine

I wanted to let you know that I just subscribed to *LJ* for two years and love the material. In Western Australia, your magazine was very hard to find, and by chance, I picked up the August 2011 issue and loved it.

I looked around after that and could not find the September 2011 issue. Finally, I picked up my old copy, checked your Web site and realized why. I actually had read the Slashdot article about the cancellation, but many months before I bought your print version.

I really just wanted to make this point: without the newsstand copy, I never would have seen the magazine and never would have become a subscriber. I am a Linux lover, and I've been reading *Linux Format/Magazine/User and Developer*, and they don't cut it compared to your

YMI RackMountPro.com

POWERPRO STORAGE SERVER

Move to The next era of Cloud Storage computing with PowerPRO

Our PowerPRO storage server transforms x86 IT infrastructure into the most efficient, shared, on-demand utility, with built-in availability, scalability, and security services for all applications and simple, proactive automated management.

intel Xeon inside

Powerful. Intelligent.

Optimized with:

- **Intel® Xeon® Processor X5690**
- **LSI 6Gb/s MegaRAID® SAS 9280 Controller**

- With VMware VMFS5, you can create a 64TB datastore on a single extent. RDMs in physical compatibility mode with the size larger than 2TB can now be presented to a virtual machine.
- Storage DRS delivers resource aggregation, automated initial placement, and bottleneck avoidance to storage.
- Profile-driven storage allows you to have greater control and insight into characteristics of your storage resources.
- vStorage APIs for Storage Awareness; Storage APIs for Array Integration: Thin Provisioning
- iSCSI UI, Storage I/O Control NFS, 2TB+ LUN, Storage vMotion snapshot support.
- Swap to Host Cache, Software FCoE, Snapshot commitments and much more

LSI

Intel, the Intel logo, Xeon, and Xeon Inside are trademarks or registered trademarks of Intel Corporation in the U.S. and other countries.
LSI, the LSI logo, MegaRAID are trademarks or registered trademarks of LSI Corporation in the U.S. and other countries.

Yang Ming International Corp. (RackMountPro.com)

The Leading Server Builder in America. Enhancing Cloud Computing, The Optimized Technologies for Cloud

595 Yorbita Road, La Puente, CA 91744 ■ Tel: (800) 526-8650 ■ Fax: (626) 956-0098 ■ sales@rackmountpro.com

mag. Going electronic only, how will people ever find you?

I am an avid reader of Slashdot and LWN. I imagine that the latter especially is your core market, and I know it struggles to find advertisers (or used to, it runs AdSense now). You probably should run a PPC campaign every so often to gain new subscribers. Just make sure you have a decent landing page, as your home page makes subscribing a mission and a half.

I really hope the switch to electronic works for you, as Linux needs all the good quality journalism and documentation it can get.

PS. I bought the back-issue DVD as well, so total lifetime value of subscribers can be pretty high.
—David

David, thanks! Yes, losing a newsstand presence is something we're bummed about. You highlight one of the changes we need to make, and that is how we promote the magazine. Thank you for the suggestion, as it's an issue we need to address.—Ed.

Digital = Awesome

I love that you went all digital. Normally, I run to the local Fred Meyers each month to pick up *LJ*. I use an iPhone and have a Kindle. I hate carrying books or magazines due to the equipment I have to carry for work. I subscribed as soon as I found out you guys went all digital. I know some are annoyed by the conversion, but I'm really liking the new format. I have 200+ manuals for work, and it makes a great addition, and when I'm on the go, I can keep reading on my iTouch. Thanks for the great work. You guys rock. I have converted my wife to Linux and even she enjoys your mag.
—Robert

That's great to hear, Robert! If you've been reading the Letters for the past few months, it's clear that reactions to the switch have been mixed. We've purposefully posted letters from both sides, because we don't want to minimize anyone's frustration. It is great to hear that a large number of folks not only tolerate, but also prefer the digital format. Thanks for the feedback!—Ed.

Electronic Version

I cannot save trees, since I have found that I need to print an article in order to be able to read it easily.

The problem is that the height of a laptop screen is not sufficient to display fonts that an old geezer like me can read without overflowing the page. Page overflows would require me to scroll up and down (and maybe left and right) in order to see everything.

I understood that you used flowing text,

Mobile changes *everywhere.*

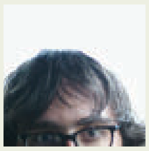
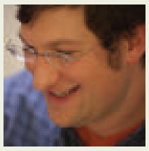
The explosion of mobile technologies has put location-aware devices everywhere. The mobile landscape is constantly evolving and making smart decisions about buying or building the right tools isn't easy.

Get a handle on what's what with location at the O'Reilly Where Conference. Learn from leading-edge developers, business leaders, marketers, and entrepreneurs what's happening how and what's lurking just below the radar—and how you can leverage location for your business.

Program Tracks

Mobile Development, Location Development, Business & Strategy, Marketing—and the return of the Location Marketing Boot Camp.

Featured Speakers

					
Thomas Goetz *Wired*	Charlene Li Altimeter Group	Jer Thorp *New York Times*	Brian McClendon Google	Noah Iliinsky Complex Diagrams	Leah Busque TaskRabbit

Save 15% on registration with code WHR12LJOU

O'REILLY®

which meant to me that page boundaries would not be hard-coded, but that zooming in could simply result in the reformatting of the pages to make the articles easier to see.
—Jim

Jim, we do use flowing text, but only in the formats that support such a thing. It sounds like you're reading the PDF version, which unfortunately is the least flexible when it comes to reformatting.

Although e-reader devices are the best for reading the "flowing" versions (epub and mobi), it is possible to read those versions on your computer as well. Calibre, for example, includes an e-book-reading application, and there are ways to read epubs from inside your Web browser. One disadvantage to having so many versions available is that it's easy to get confused regarding which version is best for different situations.—Ed.

More on E-Formatting

I'm slowly coming around to life without paper. I now have two e-zines. I have two suggestions on improving yours. 1) Add links within your Kindle version to skip the current article and jump to the next. There are just some parts that don't interest me. Right now, I have two choices: hit the Next button on my Kindle numerous times or jump to the table of contents and then jump again. 2) Give us a link to download all the formats with one link. Right now, I am downloading each of them. I use Calibre to store all my electronic reading content, and this great tool allows me to store different formats of the same document under one title.

On my Christmas list is a Kindle Fire, so I hope I am reading your magazine on this new baby come January 1.
—Scott

Scott, I'm jealous you're getting a Kindle Fire, but I hope you enjoy it nonetheless. Thank you for the formatting suggestions; we are trying every month to make the magazine more and more user-friendly. Recommendations from readers are vital, so thank you very much.—Ed.

Electronic vs. Paper

Like some of the readers who have been unpleasantly surprised by the sudden switch to pixels from paper, I was initially outraged. As one reader said, I paid for print and have issues remaining in my subscription. Your explanation of the reasons for the switch made sense, and you were obviously doing your best to give value for the money under adverse conditions, so I decided to wait and see how life in the 21st century would work. After downloading the PDF versions, and then the iPad app, I can say that I'm *glad* you switched! I much prefer the digital version. I can carry multiple issues with me, read in the dark, follow links with a finger touch rather than having to write down

URLs for later entry into a browser, and I can put the old issues up on my LAN server for future reference (the HTML versions are even searchable!). It's even easy to send thank-you notes to the editors! Thanks!
—Mike Bartman

Mike, thanks for sticking with us! We tried (and continue to try) to be as transparent as possible regarding the switch. Although there are obvious frustrations when transitioning to digital, we honestly are excited about the advantages it can offer. Of course, those advantages don't erase the frustration about the end of paper, but looking forward is exciting. We're glad you'll be here with us.—Ed.

ezPDF Reader

I read in the December 2011 issue's Letters that Dave was having issues with the PDF version on his NOOK Color. I too have a NOOK Color and would like you to please pass on some advice to Dave. Tell him to buy the app ezPDF Reader. It is far superior to the NOOK's built-in PDF reader.

By the way, I've been a subscriber for 14 years, and I love the digital transition. You've all done a fantastic job of it.
—Doug McComber

Doug, we decided this was the best way to pass the information on to Dave. We figured he'd read it here, and then anyone else reading would benefit as well! For the record, our Webmistress Katherine Druckman agrees with you 100% and seconds your recommendation of ezPDF.—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.

diff -u
WHAT'S NEW IN KERNEL DEVELOPMENT

After the security breach on **kernel.org**, the administrators of that site had to re-implement the site from scratch, and they had to prioritize the features that would be coming back soonest. They also had the opportunity to exclude features that seemed to be more trouble than they were worth.

One of the latter seems to be the **Mercurial**-based kernel repository mirror that **Matt Mackall** had maintained on kernel.org. Matt's repository offered an alternative for developers who didn't care for Git's approach to source control—or at least who just liked Mercurial better.

Matt intends to bring the repository back to life somewhere else, but he's not sure when that might be, and he's also affirmed that the new repository won't be fully compatible with the old. So, people wanting to update their local repositories will need to get a fresh one instead.

Davidlohr Bueso recently inaugurated an argument over **user quotas**, when he posted a patch to limit regular users' ability to fill up the **/tmp** directory on Linux systems. His justification was that filling up that directory represents a denial-of-service attack. His patch actually created a quota system for all **tmpfs** filesystems.

Everyone (except the attackers) agrees that DoS attacks are bad, but this may be an area where it's not so easy to agree on the best approach. **Christoph Hellwig**, for example, felt that Davidlohr should just use the existing user quota system and apply it to /tmp. **Lennart Poettering** argued that this would require a lot of user-space configuration that wouldn't be necessary with Davidlohr's approach. And, **Alan Cox** felt that other security issues were more pressing than this one. No one's argument won the day, and it's not clear that such a long-standing issue will catch **Linus Torvalds**' attention.

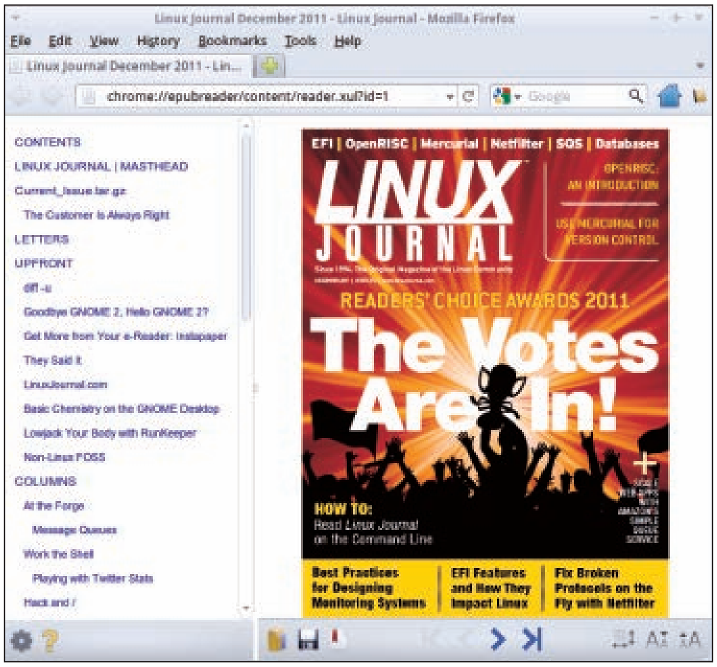
Likewise, long-standing security problems in **/proc** may not be on the radar either. **Vasiliy Kulikov** posted a patch to give tighter permissions to **/proc/interrupts**, but this got a cool reception. Apparently the longtime weaknesses of things like /proc and /tmp are dangerous only if someone already has a login account on a given machine, in which case a variety of other exploits are available as well.

Marek Szyprowski has been working on a **contiguous memory allocator** for **Samsung**. The idea is that a lot of smaller embedded systems don't support software jumping around in RAM, and

video-editing software can sometimes require a contiguous chunk of memory for its image data. But even on regular systems, contiguous RAM can provide a speedup over fragmented RAM. However, the problem with trying to make everything use contiguous memory is that eventually you get stuck with chunks of RAM that aren't big enough for anyone, and that just hang out, unused, even when the system really needs more RAM. Marek said his code wasn't quite ready for inclusion, but it was coming along.
—**ZACK BROWN**

EPUBReader

With our recent transition to a digital-only format, it's now possible to consume *Linux Journal* in a number of ways. For those so inclined, it's even possible to print each issue and bind it into a paper magazine. (The PDF lends itself quite nicely to that in fact.) Electronically speaking, however, it's hard to beat the .epub/.mobi editions.


Linux Journal December 2011 - Linux Journal - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Linux Journal December 2011 - Lin...
chrome://epubreader/content/reader.xul?id=1
Google
CONTENTS
LINUX JOURNAL | MASTHEAD
Current_Issue.tar.gz
The Customer Is Always Right
LETTERS
UPFRONT
diff -u
Goodbye GNOME 2, Hello GNOME 2?
Get More from Your e-Reader: Instapaper
They Said It
LinuxJournal.com
Basic Chemistry on the GNOME Desktop
Lowjack Your Body with RunKeeper
Non-Linux FOSS
COLUMNS
At the Forge
Message Queues
Work the Shell
Playing with Twitter Stats
Hack and /
EFI | OpenRISC | Mercurial | Netfilter | SQS | Databases
LINUX JOURNAL
OPENRISC: AN INTRODUCTION
USE MERCURIAL FOR VERSION CONTROL
READERS' CHOICE AWARDS 2011
The Votes Are In!
HOW TO: Read Linux Journal on the Command Line
Best Practices for Designing Monitoring Systems
EFI Features and How They Impact Linux
Fix Broken Protocols on the Fly with Netfilter

If you'd like to browse this month's issue without ever leaving your computer, you might want to check out EPUBReader, a Firefox add-on that turns your browser in to a mini-library. After installing the add-on, epub files are opened automatically and added to the local epub library. To access your library, simply go to Tools, and select ePub-Catalog. You'll be presented with a list of all your locally stored epub books, and with a simple click, you can be browsing.

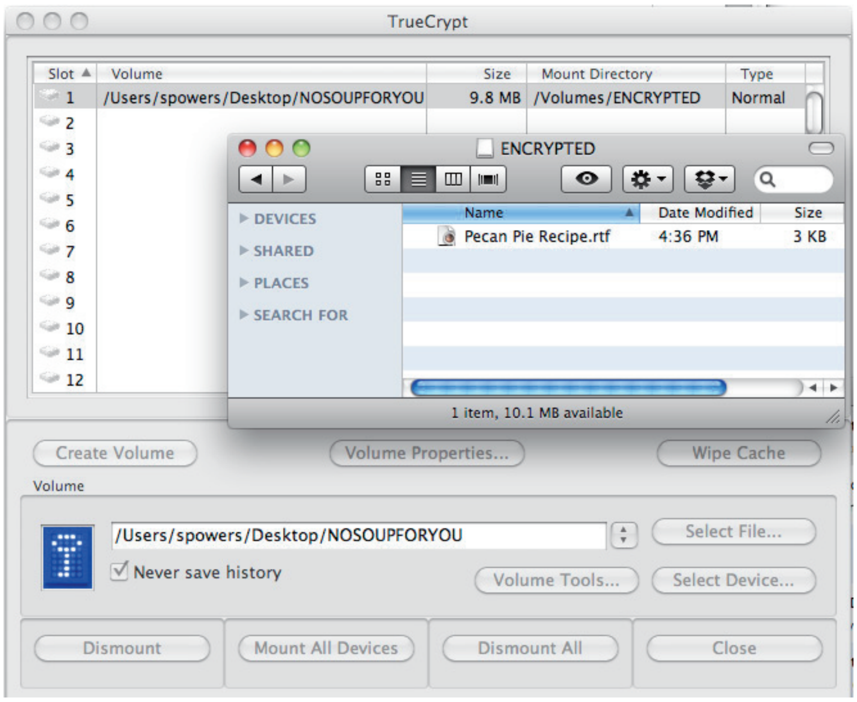
EPUBReader includes support for an interactive table of contents, which makes wise use of a computer's widescreen display. It even supports keyboard shortcuts! EPUBReader is free, cross-platform and quite honestly, a really nice e-reader app. Check it out at **http://www.epubread.com**.—**SHAWN POWERS**

Non-Linux FOSS

TrueCrypt is a fully open-source tool for encrypting data. That data can be on a completely encrypted hard drive, or just an encrypted image file. Thankfully, the encryption works the same regardless of your platform, so Windows and OS X users can share encrypted files between computers.

TRUECRYPT

We really like to use TrueCrypt in combination with Dropbox, another cross-platform tool, to protect our data in the cloud. Pictured here is the OS X version of TrueCrypt, mounting an encrypted image as a local hard drive. Whether you are storing sensitive data or Grandma's secret recipe, TrueCrypt can keep your data private, even if it's stored on someone else's server.



Slot	Volume	Size	Mount Directory	Type
1	/Users/spowers/Desktop/NOSOUPFORYOU	9.8 MB	/Volumes/ENCRYPTED	Normal
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				

Name	Date Modified	Size
Pecan Pie Recipe.rtf	4:36 PM	3 KB

1 item, 10.1 MB available

Volume
/Users/spowers/Desktop/NOSOUPFORYOU
☑ Never save history

For more information and downloadable binaries for Windows and OS X, visit **http://www.truecrypt.org**.

—SHAWN POWERS

They Said It

The glass is neither half-full nor half-empty: it's twice as big as it needs to be.

—Unknown

I would love to change the world, but they won't give me the source code.

—Unknown

I changed all my passwords to "incorrect". So my computer just tells me when I forget.

—Unknown

A stitch in time would have confused Einstein.

—Unknown

An effective way to deal with predators is to taste terrible.

—Unknown

LinuxJournal.com

This month’s issue is about my favorite Linux-related topic, Web development. The Linux-powered Web is a subject I believe we all particularly enjoy, as it is one of those fantastic instances where Linux enters many people’s lives without their awareness. It’s always fun for geeks like us to know that our work behind the scenes can have such impact on the unsuspecting public. The technologies highlighted in this issue are shaping the current and future Web, so we hope you’ll take away some inspiration to go out and make something awesome on-line.

I am more than a little biased, as Drupal powers LinuxJournal.com, but I hope you’ll be inspired to try Drupal 7 after this month’s introduction to content types and views, and if you’d like to learn even more about Drupal 7, see the following posts on LinuxJournal.com:

- “Angela Byron on Drupal 7” by Katherine Druckman: **http://www.linuxjournal.com/content/interview-angie-byron-drupal**
- “Drupal 7: the Webchick behind the Wheel” by Katherine Druckman: **http://www.linuxjournal.com/article/10999**
- “Drush: Drupal for People Who Hate Mice” by James Walker: **http://www.linuxjournal.com/article/11000**

This month, we also feature an article about HTML5’s audio capabilities, which illustrates one of the many exciting new options HTML5 brings to the table. To read more about other HTML5 possibilities, including for mobile applications, see these articles on LinuxJournal.com:

- “Augmented Reality with HTML5” by Rick Rogers: **http://www.linuxjournal.com/article/10920**
- “At the Forge—HTML5”: **http://www.linuxjournal.com/article/10926**

You’ll find more about these topics and many others in our Web Development section at **http:/www.linuxjournal.com/tag/web-development**.

—KATHERINE DRUCKMAN

Science the GNU Way, Part I

In my past several articles, I've looked at various packages to do all kinds of science. Sometimes, however, there just isn't a tool to solve a particular problem. That's the great thing about science. There is always something new to discover and study. But, this means it's up to you to develop the software tools you need to do your analysis. This article takes a look at the GNU Scientific Library, or GSL. This library is the Swiss Army library of routines that you will find useful in your work.

First, you need to to get a copy of GSL and install it on your system. Because it is part of the GNU Project, it is hosted at **http://www.gnu.org/s/gsl**. You always can download and build from the source code, but all major distributions should have packages available. For example, on Debian-based systems, you need to install the package libgsl0-dev to develop your code and gsl-bin to run that code. GSL is meant for C and C++, so you also need a compiler. Most of you probably already are familiar with GCC, so I stick with that here.

The next step is actually the last step. I'm looking at compiling now so that I can focus on all the tools available in GSL. All the header files for GSL are stored in a subdirectory named gsl. So, for example, if you wanted to include the math header file, you would use:

```
#include <gsl/gsl_math.h>
```

All the functions are stored in a single library file called libgsl.a or libgsl.so. You also need a library to handle basic linear algebra. GSL provides one (in the file libgslcblas.so), but you can use your own. Additionally, you need to link in the math library. So, the final compile and link command should look like this:

```
gcc -o hello_world hello_world.c -lgsl -lgslcblas -lm
```

There are optional inline versions for some of the performance-critical functions in GSL. To use these, you need to include `-DHAVE_INLINE` with your compile command. To try to help with portability issues, GSL offers some functions that exist only on certain platforms (and not others). As an example, the BSD math library has a function called `hypot`. GSL offers its own version, called `gsl_hypot`, that you can use on non-BSD platforms. Some functions have both a general algorithm as well as optimized versions for specific platforms. This way, if you are running on a SPARC, for example, you can select a version optimized for SPARC if it exists.

One of the first things you likely will want to do is check whether you are getting correct results from your code or if there were errors. GSL has a number of functions and data structures available in the header file named gsl_errno.h. Functions return a value of zero if everything is fine. If there were any problems in trying to complete the requested action, a nonzero value is returned. This could be an actual error condition, like a wrong data type or memory error, or it could be a condition like not being able to converge to within the requested accuracy in the function call. This is why you always need to check the return value for all GSL function calls. The actual values returned in an error condition are error codes, defined in the file gsl_errno.h. They are defined as macros that start with `GSL_`. Examples include the following:

- `GSL_EDOM` — domain error, used by functions when an argument doesn't fall into the domain over which the function is defined.
- `GSL_ERANGE` — range error, either an overflow or underflow.
- `GSL_ENOMEM` — no memory available.

The library will use values only up to 1024. Values above this are available for use in your own code. There also are string versions of these error codes available. You can translate the error code to its text value with the function `gsl_errno()`.

Now that you know how to compile your program and what to do with errors, let's start looking at what kind of work you can do with GSL. Basic mathematical functions are defined in the file gsl_math.h. The set of mathematical constants from the BSD math library are provided by this part of GSL. All of the constants start with `M_`. Here are a few of them:

- `M_PI` — pi.
- `M_SQRT2` — the square root of 2.
- `M_EULER` — Euler's constant.

There also are capabilities for dealing with infinities and non-numbers. Three macros define the values themselves:

- `GSL_POSINF` — positive infinity.
- `GSL_NEGINF` — negative infinity.
- `GSL_NAN` — not a number.

There also are functions to test variables:

- `gsl_isnan` — is it not a number?
- `gsl_isinf` — is it infinite?
- `gsl_finite` — is it finite?

There is a macro to find the sign of a number. `GSL_SIGN(x)` returns the sign of x: 1 if it is positive and –1 if it is negative. If you are interested in seeing whether a number is even or odd, two macros are defined: `GSL_IS_ODD(x)` and `GSL_IS_EVEN(x)`. These return 1 if the condition is true and 0 if it is not.

A series of elementary functions are part of the BSD math library. GSL provides versions of these for platforms that don't have native versions, including items like:

- `gsl_hypot` — calculate hypotenuse.
- `gsl_asinh`, `gsl_acosh`, `gsl_atanh` — the arc hyperbolic trig functions.

If you are calculating the power of a number, you would use `gsl_pow_int(x,n)`, which gives you x to the power of n. There are specific versions for powers less than 10. So if you wanted to find the cube of a number, you would use `gsl_pow_3`. These are very efficient and highly optimized. You even can inline these specialized functions when `HAVE_INLINE` is defined.

Several macros are defined to help you find the maximum or minimum of numbers, based on data type. The basic `GSL_MAX(a,b)` and `GSL_MIN(a,b)` simply return either the maximum or minimum of the two numbers a and b. `GSL_MAX_DBL` and `GSL_MIN_DBL` find the maximum and minimum of two doubles using an inline function. `GSL_MAX_INT` and `GSL_MIN_INT` do the same for integer arguments.

When you do any kind of numerical calculation on a computer, errors always are introduced by round-off and truncation. This is because you can't exactly reproduce numbers on a finite binary system. But, what if you want to compare two numbers and see whether they are approximately the same? GSL provides the function `gsl_fcmp(x,y,epsilon)`. This function compares the two doubles x and y, and checks to see if they are within epsilon of each other. If they are within this range, the function returns 0. If x < y, it returns –1, and it returns 1 if x > y.

Complex numbers are used in many scientific fields. Within GSL, complex data types are defined in the header file gsl_complex.h, and relevant functions are defined in gsl_complex_math.h. To store complex numbers, the data type `gsl_complex` is defined. This is a struct that stores the two portions. You can set the values with the functions `gsl_complex_rect(x,y)` or `gsl_complex_polar(x,y)`. In the first, this represents "x+iy"; whereas in the second, x is the radius, and y is the angle in a polar representation. You can pull out the real and imaginary parts of a complex number with the macros `GSL_REAL` and `GSL_IMAG`. There is a function available to find the absolute value of a complex number, `gsl_complex_abs(x)`, where x is of type gsl_complex. Because complex numbers actually are built up of two parts, even basic arithmetic is not simple. To do basic math, you can use the following:

- `gsl_complex_add(a,b)`
- `gsl_complex_sub(a,b)`
- `gsl_complex_mul(a,b)`
- `gsl_complex_div(a,b)`

You can calculate the conjugate with `gsl_complex_conjugate(a)` and the inverse with `gsl_complex_inverse(a)`.

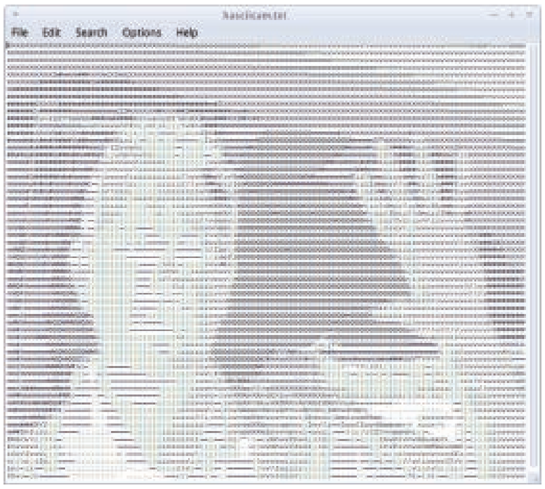
There are functions for basic mathematical functions. To calculate the square root, you would use `gsl_complex_sqrt(x)`. To calculate the logarithm, you would use `gsl_complex_log(x)`. Several others are available too.

Trigonometric functions are provided, like `gsl_complex_sin(x)`. There also are functions for hyperbolic trigonometric functions, along with the relevant inverse functions.

Now that you have the basics down, my next article will explore all the actual scientific calculations you can do. I'll look at statistics, linear algebra, random numbers and many other topics.

—JOEY BERNARD

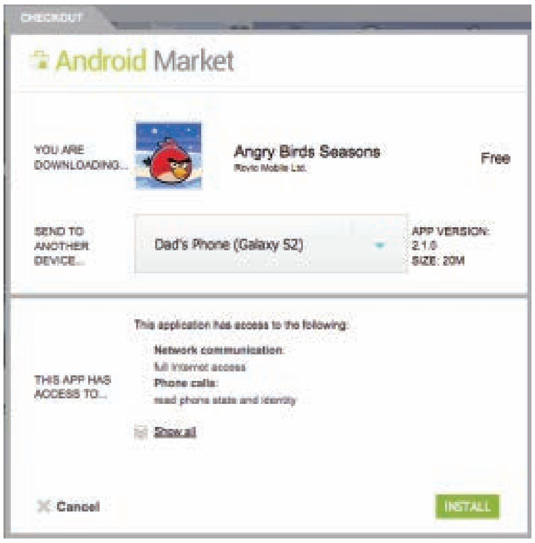
Rock Your Webcam Like It's 1995



Many Webcam applications exist for Linux. If you want to play with self-portraits, there's Cheese. If you want to set up a security system in your office, there's Motion. But, if you just want to have some fun, give HasciiCam a try.

HasciiCam does for Webcams what aalib+mplayer does for video files. It renders your live video directly into ASCII. It also has a feature to create a live Web feed of your ASCII video for the whole world to see! HasciiCam might not be the most practical application to use with your Webcam, but if you ever have been fascinated with ASCII art, it might be the most fun. It's most likely packaged with your favorite distro, but if not, the source is available from its Web site: **http://dyne.org/software/hasciicam**.—**SHAWN POWERS**

Marketplace Magic



YOU ARE DOWNLOADING... Angry Birds Seasons
Rovio Mobile Ltd.
Free

SEND TO ANOTHER DEVICE... Dad's Phone (Galaxy S2)
APP VERSION: 2.1.0
SIZE: 20M

THIS APP HAS ACCESS TO...
This application has access to the following:
Network communication:
full Internet access
Phone calls:
read phone state and identity
Show all

If you love shopping in the Android Marketplace but hate typing on that tiny keyboard, you might like the Web interface to the Marketplace. This isn't a new feature, but it's one I didn't know about, so it seemed apropos to share with the class.

On a desktop computer, simply browse over to **http://market.android.com**, and log in with your Google account. (The same account you use on your Android phone.) You can select an app and have it automatically installed on whichever Android device you like! There's no need to touch your phone; it automatically will install the app you chose from the Web site.

As you can tell from the screenshot, I found an excellent productivity app that will be on my new Galaxy S2 in a few moments. Just one more reason to love Android!—**SHAWN POWERS**

THE OPEN SOURCE WORLD COMES TO COLUMBIA, SC MARCH 28 AND 29

Keynote:
SCOTT MCNEALY
Co-founder, Sun Microsystems

Other Confirmed Speakers

CHRIS ANISZCYK
Open Source Manager, Twitter

JIM JAGIELSKI
President, Apache Software Foundation

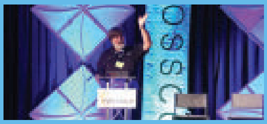
BDALE GARBEE
Chief Technologist for OS and Linux, HP

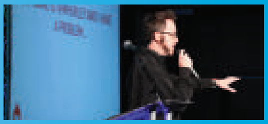
DR. DOUGLAS MAUGHAN
Director, DHS Cyber Security Division

JONATHAN LEBLANC
Technology Evangelist, PayPal

CAROL SMITH
OS Programs Manager, Google

DAVE ABRAHAMS
Founding Member, Boost.org





2012
POSSCON™
PALMETTO OPEN SOURCE SOFTWARE CONFERENCE

JOIN US FOR THE BEST SPEAKERS, TOPICS AND SOCIAL EVENTS IN THE COUNTRY

POSSCON.ORG
MARCH 28 & 29, 2012 | COLUMBIA, SC

presented by IT-oLogy™

Bringing Open Source to the Southeast
DEVELOPERS – EDUCATORS – IT LEADERS IN BUSINESS, GOVERNMENT, HEALTHCARE AND SECURITY

REUVEN M. LERNER

Simple Apps with Sinatra

Writing a small Web app? Consider Sinatra, a micro-framework that makes Web development quick and easy.

For the past month or two, I have been teaching my nine-year-old daughter to program in Ruby. She has enjoyed it a great deal, as have I. To be honest, I've been a bit surprised by how much she has gotten out of the experience and how excited she is to be discovering the joy of programming.

Not surprisingly, I'm planning to introduce her to the wonderful world of Web development. But given that we're working with Ruby, I was a bit concerned. After all, the de facto standard for Web development in Ruby is Rails, and although I think Rails is a wonderful framework, it always has assumed that newcomers were at least familiar with the basics of Web development. Modern versions of Rails, with CoffeeScript, Sass stylesheets, an asset pipeline and many other features (which are wonderful in their own right) would pose a pedagogical challenge for both my daughter and me.

Just as I was trying to figure out how to introduce Rails or whether I should give up on the idea of teaching her Web development at all, I realized I had been overlooking a gem (no pun intended) of software that is perfect for these purposes—Sinatra. Sinatra has been around for a few years already, but I never really gave it much thought. It turns out, however, that Sinatra is as easy (or easier!) than the CGI programs I wrote back in the early days of the Web, supports a huge number of plugins and features, can be hosted easily on Heroku or Phusion Passenger and is a great deal of fun to work with.

So in this article, I'm taking a look at Sinatra. I describe how to configure and install it and explain why I'm convinced that whenever I now need to do a quick-and-dirty Web application, Sinatra (and not Rails) will be my tool of choice.

Sinatra Origins and Installation

Sinatra first was written by Blake Mizerany, a longtime Ruby programmer

Sinatra makes it ridiculously easy to put together Web applications, because that's all it is meant to do.

who currently works at GitHub. Sinatra is an example of both a micro-framework—that is, a Web development framework that handles just the bare minimum of what you might want to do—and also a domain-specific language (DSL). A DSL gives you a narrow, but deep, vocabulary for writing programs in a particular domain, allowing you to concentrate on the high-level development process, rather than the nitty-gritty details.

Sinatra makes it ridiculously easy to put together Web applications, because that's all it is meant to do. At the same time, it's still written in Ruby, meaning that you have access to the language's core functionality, along with gems, ranging from ActiveRecord (from Rails) to templates to network-communication systems.

To create a simple Sinatra application, you first have to install the Sinatra gem, either as root or (if you are using rvm, the wonderful Ruby version manager) as a nonroot user:

```
gem install sinatra
```

With that in place, you're ready to get started! You can create a file that looks like this:

```
#!/usr/bin/env ruby

require 'rubygems'
require 'sinatra'

get '/' do
  "Hello"
end
```

It doesn't get easier than this, right? The first line indicates that you want to execute Ruby. The second and third lines load the Gem libraries and also the "sinatra" gem itself. The third line defines a route, such that if a Web browser uses the GET method to request /, the method defined in the "do" block is executed. Run this program, and you see this:

```
~/Desktop $ ./atf-sinatra.rb
   == Sinatra/1.3.1 has taken the stage on 4567 for
      development with backup from Thin
   >> Thin web server (v1.3.1 codename Triple Espresso)
   >> Maximum connections set to 1024
   >> Listening on 0.0.0.0:4567, CTRL+C to stop
```

In other words, you're now running a live Web application, via an HTTP server (Thin, if you have installed it on your system—otherwise, Sinatra tries to find something else). As I mentioned previously, Sinatra makes

Listing 1. atf-sinatra.rb

```
#!/usr/bin/env ruby

require 'rubygems'
require 'sinatra'
require 'erb'

get '/' do
 @first_number = rand(5000)
 @second_number = rand(@first_number)
 @operation = %w(+ - *).sample
 erb :index
end

post '/answer' do
 @first_number = params[:first_number].to_i
 @second_number = params[:second_number].to_i
 @operation = params[:operation].to_s

 @user_answer = params[:user_answer].to_i
 @right_answer = @first_number.send(@operation, @second_number)

 erb :answer
end
```

the development of Web applications easier and faster than ever was the case, even with CGI programs, and with many more capabilities.

You can stop the application by pressing Ctrl-C. Note that unlike Rails in development mode, Sinatra doesn't (by default, at least) notice when the program has changed and update itself accordingly. If you want to have such functionality, you can install the "shotgun" gem, which restarts the application with each request.

Sinatra uses the common Rack interface for Web applications under Ruby. This means that a Sinatra application, even one as simple as what I have written here, can be deployed anywhere that supports Rack, such as the various Ruby-flavored HTTP servers (for example Phusion Passenger) or on hosting sites like Heroku.

Of course, Web applications tend to receive input from the user. Simple parameters can come as part of the URL, following a question mark (?) character. A POST request also can contain parameters, but they are sent in a separate channel. Sinatra can handle both of these:

```
get '/' do
  "Hello, #{params[:name]}"
end
```

If I invoke this as:

```
http://localhost:4567/?name=Reuven
```

my Web browser happily prints the text:

```
Hello, Reuven
```

Views

Now, it's certainly possible to produce a working Web application using just the

Fortunately, Sinatra supports not just ERb templates, but also a number of other types, including popular Haml templates.

above. But, the output in this example is fairly limited. It's safe to assume that in anything more sophisticated than a "Hello, world" program, you'll want to use a templating system to display your views. Fortunately, Sinatra supports not just ERb templates, but also a number of other types, including popular Haml templates.

For example, let's say you want your "GET /" handler to invoke an ERb template, and that you will call the template "index". Create a subdirectory named views in the same directory where the Sinatra application is located. Then, put a file named index.erb in the views directory, and put some text into it:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Sinatra index</title>
  </head>
  <body>
    <h1>Sinatra</h1>
    <p>Hello, <%= @name %> </p>
  </body>
</html>
```

Finally, modify the "get" handler method such that it both uses the ERb template and so that it assigns the instance variable `@name` with something:

```
get '/' do
  erb :index
  @name = params[:name]
end
```

As is the case in all ERb templates, such as in Rails, instance variables (that is, those starting with @) are passed untouched to the template. Your ERb template may contain any HTML you want, as well as Ruby code inside of the <% %> delimiters. One bug (or feature?) that kept biting me was the lack of <%= -%> delimiters, as in Rails. I quickly trained my fingers not to type it, but I admittedly was confused when Sinatra gave me a stack backtrace that was less than obvious.

A Simple Application

So, what kinds of simple applications can you create with Sinatra? You can, of course, use a database on the back end or any number of other features, such as sessions or authentication. But, there are a lot of little applications I've written during the years that didn't require much in the way of server-side programming. One example is a simple

Listing 2. index.erb

```
<!DOCTYPE html>
<html>
 <head>
   <title>Sinatra Math</title>
 </head>
 <body>
   <h1>Sinatra Math</h1>

   <pre>
       <%= sprintf("%6d", @first_number) %>
       <%= @operation %> <%= sprintf("%4d", @second_number) %>
   </pre>
   <hr align="left" width="200" />

   <form method="POST" action="/answer">
     <input type="text" name="user_answer" />
     <input type="submit" value="Submit answer" />
     <input type="hidden" name="first_number"
      value="<%= @first_number %>" />
     <input type="hidden" name="second_number"
      value="<%= @second_number %>" />
     <input type="hidden" name="operation"
      value="<%= @operation %>" />
   </form>

 </body>
</html>
```

math-practice program I wrote for my daughters several years ago using Rails. How easy would it be to produce such a program in Sinatra?

First, you need two different handlers, one for GET requests and one to POST the answers. The GET handler will choose two random numbers and a math operation to perform on them. This handler then will display the results as a math problem on a page of HTML, providing a form in which the user can submit the answer. My GET handler looked like this:

```
get '/' do
  @first_number = rand(5000)
  @second_number = rand(@first_number)
  @operation = %w(+ - *).sample
  erb :index
end
```

This code sets three instance variables, the values of which then are available for display in the ERb template, which will be index.erb in the views directory. The first number can be anything up to 5,000, and the second number must be smaller than the first. I then allowed for addition, subtraction and multiplication. Doing division with two 4-digit numbers seemed like a stretch for my children at this stage, even by my tough standards. By giving the first number as the parameter to the second call to rand(), you ensure that the second number will be no larger than the first—an important consideration when producing elementary-school problems.

index.erb (Listing 2) is, as you would expect, rather plain. The most interesting part is the HTML form, into which users will provide their answers:

```
<form method="POST" action="/answer">
  <input type="text" name="user_answer" />
  <input type="submit" value="Submit answer" />
  <input type="hidden" name="first_number"
   value="<%= @first_number %>" />
  <input type="hidden" name="second_number"
   value="<%= @second_number %>" />
  <input type="hidden" name="operation"
   value="<%= @operation %>" />
</form>
```

Notice how the original operands and operation are kept as hidden fields in the form. This allows you to avoid putting them in the session and makes debugging easier.

The answer-checking program, which expects to receive three parameters (first_number, second_number and operation), then grabs them and calculates the right answer:

```
post '/answer' do
  @first_number = params[:first_number].to_i
  @second_number = params[:second_number].to_i
  @operation = params[:operation].to_s

  @user_answer = params[:user_answer].to_i
  @right_answer = @first_number.send(@operation, @second_number)

  erb :answer
end
```

There is a bit of magic in the above code. Because operators (such as +) are methods in Ruby, you can turn the operator into a symbol and then send that symbol to the first operand. In other words, the Ruby calculation:

```
5 + 3
```

is always turned into:

```
5.send(:+, 3)
```

Listing 3. answer.erb

```
<!DOCTYPE html>
<html>
 <head>
   <title>Sinatra Math</title>
 </head>
 <body>
   <h1>Sinatra Math</h1>

   <p>Problem: <%= "#{@first_number} #{@operation}
                  #{@second_number}" %>

   <p>Your answer: <%= @user_answer %></p>
   <p>Right answer: <%= @right_answer %></p>

   <% if @user_answer == @right_answer %>
   <p>You got it right!</p>
   <% else %>
   <p>Sorry, you got it wrong.</p>
   <% end %>

   <p><a href="/">Try again?</a></p>
 </body>
</html>
```

Sinatra is packed with features. It's hard to exaggerate just what this tiny framework can do.

But in this case, you do it explicitly, turning the string into a symbol.

It took me just a few minutes to throw together this program. With a bit more time, it's hard to say just what the limit would be.

Other Features

Sinatra is packed with features. It's hard to exaggerate just what this tiny framework can do. For example, if you want to use sessions, such that you can carry user data from one HTTP request to the next, you can enable the "sessions" feature:

```
enable :sessions
```

This creates a hash-like object ("session") into which you can store or retrieve any Ruby object—although it's generally considered to be a good idea to store only simple objects, keeping more serious ones inside of a database or object store.

Sinatra also provides sophisticated routing, such that you can tell it to accept requests for a particular URL template and then put one or more elements of the template into parameters. Sinatra provides before-and-after filters, which can be activated on some or all of the handlers. Really, the number of simple applications that are appropriate for Sinatra is almost limitless.

Conclusion

If you want to throw together a simple Web application, but don't want the overhead of Rails, you seriously should try Sinatra. It has a huge number of features, great documentation, lots of community support, and it can be deployed just about everywhere Rails can be. I'm planning to use Sinatra for many small projects in the future, both with and without my kids. It's not as powerful as Rails, but that's just the point!■

Reuven M. Lerner is a longtime Web developer, architect and trainer. He is a PhD candidate in learning sciences at Northwestern University, researching the design and analysis of collaborative on-line communities. Reuven lives with his wife and three children in Modi'in, Israel.

Resources

The Sinatra home page is at **http://sinatrarb.com**. From there, you also can get to the Sinatra book (**http://sinatra-book.gittr.com**) and the community-supported Rack (**http://sinatra-book-contrib.com**).

There are similar small frameworks in Python, if you're less of a Rubyist, such as "Denied". I have played with these only briefly, but they're worth examining if you and/or your project works better with Python.

Get Under the Hood of Android & Embedded Linux

- android builders summit / embedded linux conference
- february 13 - 14, 2012 / february 15 - 17, 2012
- san francisco bay area
- keynotes: tom moss(3lm), jason krider(ti), jonathan corbet, greg kroah-hartman
- +++++ register now +++++
- top android and embedded developers
- expert training for mobile developers
- view the programs at events.linuxfoundation.org

THE LINUX FOUNDATION
ANDROID™
BUILDERS
SUMMIT

Collaborate at the systems level of Android.

Embedded Linux Conference

A forum for using Linux in embedded products.

THE LINUX FOUNDATION

The Linux Foundation

DAVE TAYLOR

Scrabble and Words With Friends

Got a burning desire to cheat on your next crossword or *Scrabble* game? If so, you'll want to explore Dave's latest shell script.

It's been a while since I've delved into the world of games and game programming in this column, so I thought it'd be interesting to look at writing a word-finding or matching script from the perspective of crossword puzzles and *Scrabble*-like games.

You know the drill. You're playing and have encountered a six-letter word with the clue "cat, family", and you know only a few letters: _ E _ I _ _. You've already figured out "feline", but can you write a script that you could give that sort of partial match and have it figure out possibilities? Let's give it a whirl.

Finding a Word List

Linux includes the slick GNU aspell program, but the dictionary is compiled, and it's also not designed for word games but is instead based on common English language usage—not optimal.

Fortunately, you can grab some free word-list files off the Internet to use as the basis for this particular script. The one I work with here is the English Open Word List, which contains almost 129K words. Grab a copy for yourself at **http://dreamsteep.com/projects/the-english-open-word-list.html**.

The download is a ZIP archive, and it's a bit of a mess, sprawling across multiple directories and ultimately creating a set of 26 files called A Words.txt, B Words.txt and so on. I dumped it all into a single file called words.txt by using the following:

```
cat *Words.txt > words.txt
```

The resultant file is not tiny:

```
$ wc ~/words.txt
128985  128985 1123743 /Users/taylor/words.txt
```

That's okay though. On a modern Linux

That’s actually the problem with crossword-puzzle-finder apps. There simply are too many words in the English language.

system, it’ll still be lightning fast, even with 128,985 lines of data.

Looking for Words

Let’s start by looking up that partial match for “feline” from earlier. Fortunately, we don’t need to look much further than grep, because we can express what we need with a regular expression:

```
$ grep -E '.e.i..' words.txt
```

The problem is, a quick glance at the output reveals that this pattern isn’t good enough:

```
abbreviate
aberdevine
abiogenist
abstemious
academical
```

To root the search so that it’s an exact match, not a portion within a larger word, we need to root the regular expression. This is done with ^ to denote the beginning of the line and $ to denote the end:

```
$ grep -E '^.e.i..$' words.txt
aecium
aedile
aerial
aerier
```

That’s better, except there are a lot of words that match this pattern. In fact, it’s surprisingly common. How common? Let’s see:

```
$ grep -E '^.e.i..$' words.txt | wc -l
     264
```

Sheesh, there are 264 possibilities. Drop in another letter though, and it’s a bit more manageable:

```
$ grep -E '^.e.in.$' words.txt | wc -l
      58
```

One more letter, and although theoretically we should be able to guess it anyway, at least the choices are reduced to a viewable list:

```
$ grep -E '^.elin.$' words.txt
feline
heling
reline
```

That’s actually the problem with crossword-puzzle-finder apps. There simply are too many words in the English language. Still, it's quite possible that for some letter combinations, even having most letters missing still produces a short list:

```
$ grep -E '^...cc.t.$' words.txt
braccate
placcate
spiccato
staccato
stoccata
```

This should be motivation to learn how to solve as much of that darn crossword puzzle as possible before turning to our little shell solution!

Words With Friends

Games like *Scrabble* and *Words With Friends* present an opposite problem. We know all the letters; we just want to figure out what words we can create with them.

On first blush, the solution could be as simple as searching the words database for all words that contain zero or one occurrence of each letter in the sequence. Not so fast though—look at this:

```
$ grep -E '^s*t*a*c*c*a*t*o*$' words.txt
aa
act
at
cat
coo
oo
sac
sat
scat
scatt
so
st
staccato
ta
taco
tact
tao
tat
tatt
tattoo
to
too
```

There are two problems here. The first is that the * actually is shorthand for zero or more matches, and the second is that the letters in the pattern are analyzed in order of occurrence. So, although "tao" is a valid word extracted from the letters "STACCATO", the word "tots" never will be found, because the letter s appears before the others in the pattern.

The first problem can be solved by making a more complex regular expression: `X{0,1}` means zero or exactly one occurrence of the letter X in the pattern. It's definitely more complex:

```
$ grep -E '^s{0,1}t{0,1}a{0,1}c{0,1}c{0,1}a{0,1}t{0,1}o{0,1}$'
 ➥words.txt
aa
act
```

However, a more interesting way to try to address this is to take a complete left turn and lose this increasingly complicated regular-expression approach by using a chained set of simple one-letter grep patterns instead.

```
at
cat
sac
sat
scat
so
st
staccato
ta
taco
tact
tao
tat
to
```

The second problem of letter order is a lot harder to solve. We could generate random sequences, but some quick math shows that if we have ten characters, that's 10*9*8*7*6*5*4*3*2 possible combinations—a lot!

However, a more interesting way to try to address this is to take a complete left turn and lose this increasingly complicated regular-expression approach by using a chained set of simple one-letter grep patterns instead.

That's not quite what we'll do though, because the first thing we need to do is screen out all words that contain letters not in our acceptable set. That's done with the set notation:

```
$ grep -vE '[^staccato]' words.txt
aa
accoast
accoasts
accost
accosts
```

That's still not quite right, because we're not taking into account the frequency of individual letters (note that "accoasts" has two occurrences of the letter s, but there's only one in our original letter set).

Let's wrap this up here for this month, however. Next month, I'll dig deeper into the problem and see if I can come up with some sort of smart solution.■

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

KYLE RANKIN

Password Cracking with GPUs, Part II: Get Cracking

Your hardware is ready. Now, let's load up some software and get cracking.

In Part I of this series, I explained how password cracking works in general terms and described my specific password-cracking hardware. In this article, I dig into the software side of things and describe how to put that hardware to use cracking passwords. I also discuss the two main types of attacks: dictionary and brute-force attacks. As I describe each attack, I also give specific examples of how I used the software to attack phpass, the hashing algorithm currently used for PHP-based software like WordPress.

For the purposes of this article, I created a sample WordPress blog on my server and created a few custom accounts—some with weak passwords and others with truly random passwords. Then, I went into the database for the site and pulled out the phpass password hashes for each account and put them into a file that looked like this:

```
$P$BpgwVqlfEwuaj.FlM7.YCZ6GQMu15D/
$P$BGMZP8qAHPjTTiTMdSxGhjfQMvkm2D1
$P$BOPzST0vwsR86QfIsQdspt4M5wUGVh.
$P$BjAZ1S3pmcGOC8Op808lOK4l25Q3Ph0
$P$BPlIiO5xdHmThnjjSyJ1jBICfPkpay1
$P$BReStde51ZwKHVtiTgTJpB2zzmGJW91
```

The above hashes are legitimate phpass hashes created from six-character passwords. I could tell you the passwords, but that would defeat the fun of cracking them yourself.

Proprietary Video Drivers

For those of you who, like me, believe in open-source software, this next section may be a bit disappointing. To

At first glance, it can be a bit confusing, as you can choose from hashcat, oclHashcat, oclHashcat-plus, oclHashcat-lite and even software called maskprocessor.

get hardware-accelerated password-cracking software working on your system, you need to install the proprietary video drivers from either AMD or NVIDIA. That said, if you already have been using your system for Bitcoin mining, you already have the drivers and libraries you need, so you can skip to the next section about Hashcat. Honestly, you also could just follow the Bitcoin mining HOWTOs for Linux, and that would describe how to get all the drivers and libraries you need.

Many modern desktops make it relatively easy to pull down and install the proprietary video drivers. For instance, an Ubuntu desktop will prompt you that restricted drivers are available to install both for AMD and NVIDIA cards. Most other popular distributions provide good documentation on how to pull down the proprietary drivers as well. In the worst case, you may have to download the software directly from AMD or NVIDIA and install it that way—they both have clear instructions and software available for Linux just like for other OSes.

Once you have the proprietary drivers installed, you also need the AMD APP SDK for its OpenCL libraries or the NVIDIA CUDA libraries, depending on who made your video card. You likely will need to get these libraries directly from the AMD or NVIDIA Web sites. The install is straightforward though. In my case, I downloaded the AMD-APP-SDK-v2.5-lnx32.tgz file from AMD, extracted it, and ran the provided Install-AMD-APP.sh shell script as root.

Hashcat

Many different password-cracking suites exist both for CPU- and GPU-based cracking. After reviewing all the options, I decided on the Hashcat family of cracking tools available at **http://hashcat.net**. On the site, you will see that a number of different tools are available. At first glance, it can be a bit confusing, as you can choose from hashcat, oclHashcat, oclHashcat-plus, oclHashcat-lite and even software called maskprocessor. Each program has its purpose though, depending on what you intend to do:

hashcat:

- CPU-based, so slower than the

GPU-based software.

- Supports the widest range of hashing algorithms.

oclHashcat:

- GPU-based password cracker.
- Supports a moderate number of hashing algorithms.
- Built-in support for dictionary, brute-force and mask attacks.

oclHashcat-plus:

- GPU-based.
- Supports the most hashing algorithms of the GPU-based hashcat crackers.
- Optimized for dictionary attacks against multiple hashes.
- Can support dictionary input from a pipe, so brute-force is possible.

oclHashcat-lite:

- GPU-based.
- Optimized for attacks against a single password hash.
- Fastest of the hashcat family, but with the most-limited password hash support.

maskprocessor:

- Generates dictionaries based on patterns you supply.
- Not a password cracker in its own right, but can pipe output to oclHashcat-plus for a brute-force attack.

Even with the above lists, it may not always be clear which software to use. Basically, it comes down to what type of password you want to crack and what kind of attack you want to use. The page on hashcat.net devoted to each piece of software provides a list of the hashing algorithms they support along with benchmark speeds of how many comparisons they can do per second on different types of hardware. For a given password hash, go through those pages and see which type of Hashcat software supports your hash and has the highest benchmarks. Beyond that, use oclHashcat for mask or brute-force attacks against multiple hashes, oclHashcat-lite for single hashes or oclHashcat-plus if, as was the case with me, it's the only GPU-accelerated version that supported your hash.

Once you decide which type of Hashcat software to use, installation is relatively simple, if old-school. Just download the .7z package that corresponds to the software, and use the 7za command-line tool (which should be packaged for your distribution) to extract it. The software will extract into its own directory that provides 32- and 64-bit versions for both Linux and Windows. If you have NVIDIA hardware, you will use the binaries that begin with cuda; otherwise, you will use the versions that start with ocl. The directory also will contain a number of example hashes and dictionaries and example shell scripts you can use to make sure your libraries and drivers are in place. For instance, here's the example provided with the oclHashcat-plus software for cracking a phpass hash on a 64-bit system:

```
cat example.dict | ./oclHashcat-plus64.bin -m 400 example400.hash
```

Here's what the command output looked like when I ran the example against the 32-bit version on my test rig:

```
cat example.dict | ./oclHashcat-plus32.bin -m 400 example400.hash
oclHashcat-plus v0.06 by atom starting...

Hashes: 1
Unique salts: 1
Unique digests: 1
Bitmaps: 8 bits, 256 entries, 0x000000ff mask, 1024 bytes
Rules: 1
GPU-Loops: 128
GPU-Accel: 16
Password lengths range: 1 - 15
Platform: AMD compatible platform found
Watchdog: Temperature limit set to 90c
Device #1: Cayman, 2048MB, 0Mhz, 22MCU
Device #1: Allocating 52MB host-memory
Device #1: Kernel ./kernels/4098/m0400.Cayman.32.kernel (274238 bytes)

Starting attack in wordlist stdin mode...
$H$9y5boZ2wsUlgl2tI6b5PrRoADzYfXD1:hash234

Status.......: Cracked
Input.Mode...: Piped
Hash.Target..: $H$9y5boZ2wsUlgl2tI6b5PrRoADzYfXD1
Hash.Type....: phpass, MD5(Wordpress), MD5(phpBB3)
Time.Running.: 1 sec
Time.Util....: 1008.2ms/0.0ms Real/CPU, 0.0% idle
Speed........:    65009 c/s Real,    619.7k c/s GPU
Recovered....: 1/1 Digests, 1/1 Salts
Progress.....: 65543
Rejected.....: 0
HW.Monitor.#1:  0% GPU, 47c Temp

Started: Mon Dec  5 21:12:03 2011
Stopped: Mon Dec  5 21:12:04 2011
```

In this case, the password was hash234. For all of the hashcat commands, it's simple enough just to open a terminal and change to the directory you extracted and run the commands locally from there. At the beginning of the command output, you will be able to see what GPUs the software can detect. If you have multiple GPUs in use (even if they aren't chained), it should find them automatically. If it can't find your GPU, you will need to revisit how you installed your proprietary drivers and extra libraries.

Dictionary Attacks

The first attack you should try is a dictionary attack. With a dictionary attack, you provide the cracking software with a dictionary full of possible passwords to try, such as all the words in the English dictionary. The cracking software then tries each dictionary word until one matches your hash. Since the number of combinations in a dictionary attack is much smaller than with a brute-force attack, dictionary attacks complete much faster. As an example, when I was first researching this article, I let a brute-force attack run for days against a sample set of hashes without cracking one of them. I was able to crack three out of the five hashes with a dictionary attack in less than a minute.

To run a dictionary attack with oclHashcat-plus, first run the command with the `--help` argument. That will provide you with the number that corresponds to the hash algorithm you want to crack. In the case of phpass, that number is 400. Then, run the command a second time and specify the password hash to use with the `-m` option, the file in which to store the recovered passwords with the `-o` option, and then list the file that contains your hashes and the file or files you want to use as a dictionary. Here's an example dictionary attack against phpass hashes:

```
/path/to/oclHashcat-plus32.bin -m 400 -o recovered_hashes
➥example400.hash example.dict
```

If I had multiple dictionaries, I could list all of them on the command line or even use a shell glob. A dictionary attack is only

Try Before You Buy!

Benchmark Your Code on Our GPU Cluster with AMBER, NAMD, or Custom CUDA Codes

NEW Microway MD SimCluster with 8 Tesla M2090 GPUs, 8 CPUs and InfiniBand 30% Improvement Over Previous Teslas

Configure your WhisperStation or Cluster today!
www.microway.com/tesla or 508-746-7341

Microway®

NVIDIA TESLA
intel Technology Provider Platinum 2011
AMD Fusion Partner Premier
GSA

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

as good as its dictionaries, but a number of good password dictionaries are available on the Web that you can find with a quick search for "password cracking wordlist".

Calculating Cracking Speed

Before I discuss brute-force attacks in detail, it's important to learn how to estimate how long a particular brute-force attack will take. With a brute attack, you aren't just going through a dictionary of words, you are actually trying all possible combinations of a set of characters. In researching the article, I wasted days attempting a brute-force attack against an eight-character password before I finally did the math and realized it was completely impractical.

The first step is to figure out how fast your hardware can crack a particular type of hash. As you will discover, the number of comparisons per second your hardware can perform will vary widely depending on the hash type, so start a sample brute-force attack just long enough to get a bit of progress output, and then press Ctrl-c to exit. In my case, because I'm using oclHashcat-plus, I needed to download and extract the maskprocessor software from hashcat.net, so it, combined with oclHashcat-plus, could perform a brute-force attack against phpass (don't worry about the command syntax for now, I

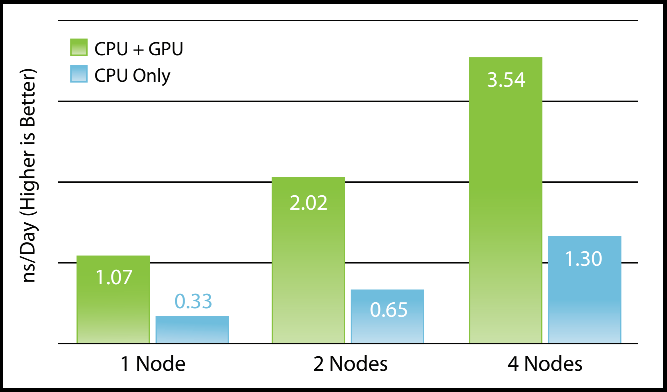
Microway's Proven GPU Expertise

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

Bio-IT World Conference & Expo '11 Best of Show Winner

Life Sciences insight '11 Award Winner for Best New Technology

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



NAMD F1-ATP Performance Gain

Visit Microway at SC11 Booth 2606

discuss the specifics later):

```
/path/to/mp32.bin -1 ?d?l?u ?1?1?1?1?1?1 | \
/path/to/oclHashcat-plus32.bin -m 400 \
-o recovered_hashes phpass-hashes

oclHashcat-plus v0.06 by atom starting...

Hashes: 6
Unique salts: 6
Unique digests: 6
Bitmaps: 8 bits, 256 entries, 0x000000ff mask, 1024 bytes
Rules: 1
GPU-Loops: 128
GPU-Accel: 16
Password lengths range: 1 - 15
Platform: AMD compatible platform found
Watchdog: Temperature limit set to 90c
Device #1: Cayman, 2048MB, 0Mhz, 22MCU
Device #1: Allocating 264MB host-memory
Device #1: Kernel ./kernels/4098/m0400.Cayman.32.kernel (274238 bytes)

Starting attack in wordlist stdin mode...

Status.......: Running
Input.Mode...: Piped
Hash.Type....: phpass, MD5(Wordpress), MD5(phpBB3)
Time.Running.: 10 secs
Time.Util....: 10001.4ms/180.8ms Real/CPU, 1.8% idle
Speed........:   315.3k c/s Real,   351.4k c/s GPU
Recovered....: 0/6 Digests, 0/6 Salts
Progress.....: 3153920
Rejected.....: 0
HW.Monitor.#1: 96% GPU, 54c Temp
```

The output line to pay attention to is the line that begins with `Speed`. As you can see from that output, my GPU can do around 350,000 comparisons per second, so I'll use that number for the rest of my calculations.

One good site I've found for doing cracking estimates is **http://www.lockdown.co.uk/?pg=combi**. This site describes all sorts of different character sets and password lengths, and it describes how long anything from a single Pentium CPU to a mythical government supercomputer might take to brute-force all combinations. Otherwise, the math is pretty straightforward. Just take the number of characters in your character set (for instance, all lowercase letters would equal 26), then figure out how long of a password you want to brute-force, then raise the first number to the power of the second.

So, for instance, all mixed-case alphanumeric characters (A–Za–z0–9) equals 62 characters. If I wanted to brute force a six-character password, that would be 62^6 = 57 billion combinations.

If you divide 57 billion combinations by a system that can do 350,000 comparisons a second, you get approximately 45 hours to complete the brute-force attack. That's not bad, but let's do the same math for eight-character passwords: 62^8 = 218 trillion combinations.

At 350,000 comparisons per second, it would take me approximately 7,200 days, or 19 years, to complete the attack. On the plus side, for another $250, I could complete the attack in less than 10 years! If you add symbols to your brute-force attack, the number jumps to 7.2 quadrillion combinations, or around 652 years.

Brute-Force Attacks

Once you've figured out whether a brute-force attack will complete in your lifetime, the next step is to run maskprocessor and tell it what kind of word list to generate. The maskprocessor command supports a number of common character sets by default with the following symbols:

- `?d` = all decimals (0–9).
- `?l` = lowercase characters (a–z).
- `?u` = uppercase characters (A–Z).
- `?s` = symbols.

You also can define a custom character set with `-1` (or `-2`, `-3`) and then use `?1` to use that custom set. For instance, if I wanted to enumerate through all three-character passwords made up of lowercase characters and numbers, I could type:

```
/path/to/mp32.bin -1 ?d?l ?1?1?1
000
001
. . .
zzy
zzz
```

In my example brute-force attack, I wanted to run through all combinations of uppercase, lowercase and numbers in a six-character password. The resulting maskprocessor command would be:

```
/path/to/mp32.bin -1 ?d?l?u ?1?1?1?1?1?1
```

Then, I would pipe the output of that command to oclHashcat-plus:

```
/path/to/mp32.bin -1 ?d?l?u ?1?1?1?1?1?1 | \
/path/to/oclHashcat-plus32.bin -m 400 \
-o recovered_hashes phpass-hashes
```

As with my dictionary attack, the `-m` option specifies I want to crack phpass hashes, the `-o` lists the file in which I want to store my recovered hashes, and finally, I specify the file that contains the phpass hashes to crack. On my hardware, it took around two days to run fully through the above brute-force attack.

Now you should be ready to get cracking, but as you'll find, the world of password cracking can get pretty dense, pretty quickly. In my next and final part of the series, I will discuss how you can tune the above attacks to get better performance, and also how to blend both dictionary and brute-force attacks to get the best of both worlds.■

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.

Resources

Hashcat: **http://hashcat.net**

Password Recovery Speeds: **http://www.lockdown.co.uk/?pg=combi**

SHAWN POWERS

Moodle. It Works.

Linux has become mainstream in server rooms, commonplace on the desktop and dominant in mobile devices. Unfortunately, it's still a rarity in the classroom. This month, I start a column dedicated to Linux in education. Grab a desk, sharpen your pencils and join me!

***LJ*'s newest column,** the one you're currently reading, is dedicated to education. Each month, I plan to equip readers with ways to improve student achievement using freely available tools. Schools are required to function on less and less money every year, yet due to the demand for technologically proficient students, they need to spend more and more every year on technology. In turn, there is no money left to invest in training the teachers. That is one reason open-source software is such a perfect fit for education. If money is spent on preparing teachers rather than buying software, we will have state-of-the-art technology and teachers fully prepared to use it. That's why this column was born. If our kids grow up with open-source tools, they'll turn into adults who understand the difference between investing in software versus investing in people. Basically, we want to take over the world, and we want to start with the kids.

Moodle is an open-source Learning Management System (LMS) designed for students of various ages. The concept of LMS isn't a new one, and it isn't limited to the FOSS world. Many colleges are using systems like Blackboard (**http://www.blackboard.com**) to facilitate a virtual classroom environment. Blackboard offers many features, including collaboration, document management, information dissemination and even live discussions. Unfortunately, those awesome features come at a substantial price. Enter: Moodle.

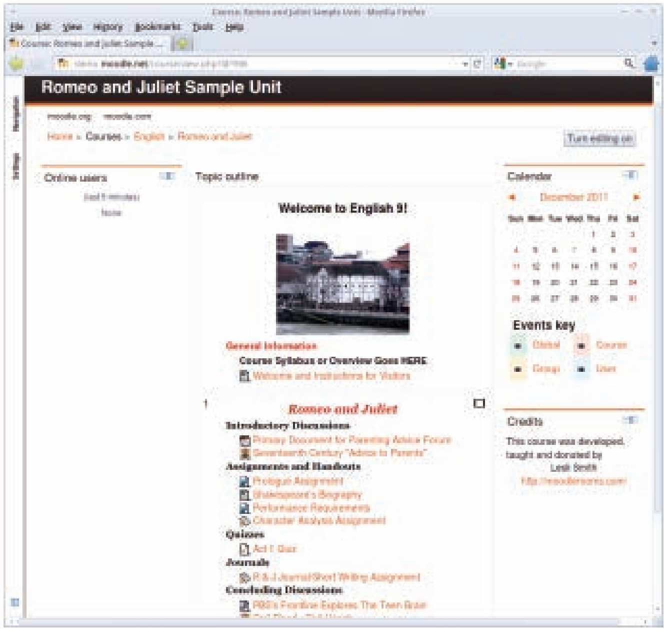
Linux Journal has covered Moodle before, but it has matured so much through the years that it deserves another look. Moodle started as a little project teachers might implement in their classrooms, but it has grown into a product used at multiple universities and schools around the world.

What Does It Do?

Figure 1 shows a demonstration course set up in Moodle. Instructors can set up their "classrooms" in a variety of ways. In the *Romeo and Juliet* example, the instructor has given a basic outline of the class with appropriate links for downloading assignments, taking on-line quizzes and on-line journaling. Figure 2 shows a demo course on Film Studies. In this example, the instructor has broken up the class into sections (apparently, each section represents a class session). Classes can be broken up in several ways—flexibility is one of Moodle's strengths. Every instructor doesn't have to use it the same way.

Why Would a School Choose Moodle?

Let's be honest. For most *Linux Journal* subscribers, this one is simple. When it comes to school boards and school administrators, the argument, "Because it's open source!", doesn't really mean much. Certainly Moodle goes along with the traditional reasons an organization might choose open source: the initial cost is lower, the maintenance cost is lower,



Romeo and Juliet Sample Unit

moodle.org moodle.com

Home » Courses » English » Romeo and Juliet

Online users

(last 5 minutes)

None

Topic outline

Welcome to English 9!

General Information

Course Syllabus or Overview Goes HERE

Welcome and Instructions for Visitors

1

Romeo and Juliet

Introductory Discussions

Primary Document for Parenting Advice Forum

Seventeenth Century "Advice to Parents"

Assignments and Handouts

Prologue Assignment

Shakespeare's Biography

Performance Requirements

Character Analysis Assignment

Quizzes

Act 1 Quiz

Journals

R & J Journal/Short Writing Assignment

Concluding Discussions

PBS's Frontline Explores The Teen Brain

Calendar

December 2011

Sun	Mon	Tue	Wed	Thu	Fri	Sat
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

Events key

Global Course

Group User

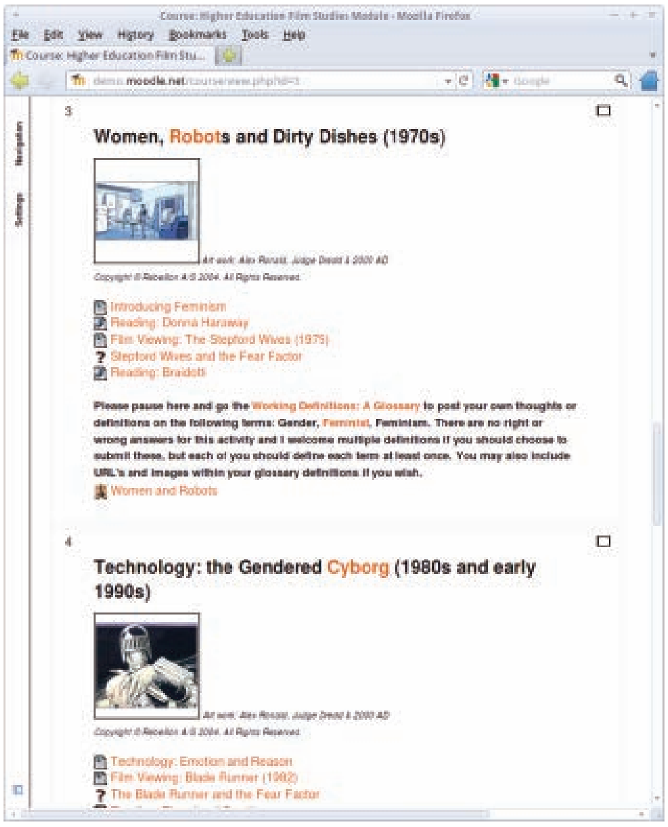
Credits

This course was developed, taught and donated by

Leslie Smith

http://moodlersons.com

Figure 1. Sample Course, Showing One Possible Layout


Course: Higher Education Film Studies Module - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Course: Higher Education Film Stu...
demo.moodle.net/course/view.php?id=3
Google
Navigation
Settings

3

Women, Robots and Dirty Dishes (1970s)

Art work: Alex Ronald, Judge Dredd & 2000 AD

Copyright © Rebellion A/S 2004. All Rights Reserved.

Introducing Feminism

Reading: Donna Haraway

Film Viewing: The Stepford Wives (1975)

Stepford Wives and the Fear Factor

Reading: Braidotti

Please pause here and go the Working Definitions: A Glossary to post your own thoughts or definitions on the following terms: Gender, Feminist, Feminism. There are no right or wrong answers for this activity and I welcome multiple definitions if you should choose to submit these, but each of you should define each term at least once. You may also include URL's and images within your glossary definitions if you wish.

Women and Robots

4

Technology: the Gendered Cyborg (1980s and early 1990s)

Art work: Alex Ronald, Judge Dredd & 2000 AD

Copyright © Rebellion A/S 2004. All Rights Reserved.

Technology: Emotion and Reason

Film Viewing: Blade Runner (1982)

The Blade Runner and the Fear Factor

Figure 2. Sample Course, Showing a Different Layout

licensing isn't an issue, it's free, free, free and so on. But with Moodle, there are even more reasons.

Moodle was designed by teachers and made for teachers. Although certainly the commercial alternatives have large ties to the educational community, ultimately their goal is to make money. There's nothing wrong with that, but it would be foolish to assume altruism when it doesn't exist. Moodle, on the other hand, was designed fully and completely for the sole purpose of improving pedagogy. Moodle exists to bolster student achievement. Period.

Features

This is where Moodle really shines. Students can interact with the course directly in their browsers. It means less paper waste and fewer lost assignments. It also means a single place for all course information. Resources like handouts, photos, URLs and study notes can be placed into the Moodle course and always be available to students. Some instructors simply use Moodle as an on-line repository for those types of documents and don't use the interactive features Moodle offers. Nothing is wrong with that approach, and Figure 1 shows that type of Moodle in action.

For teachers wanting to use Moodle as a more interactive platform, there are many built-in features:

- Assignments: students can upload finished assignments directly to the teacher, upload a file with comments or additional information, or even complete an assignment directly into the on-line text editor (Figure 3). The instructor decides how the assignment should be completed, and students complete it using the appropriate method. Moodle even allows assignments to be completely off-line, with no interaction from the student. They simply are listed so the student knows the assignment must be completed. Assignments are graded by the instructor, and students can see the results on-line.

- Gradebook: instructors can choose to keep their gradebooks directly in the Moodle database. This allows students to see their grades at any time and also interact with other modules that can assess materials and automatically enter grades.

- Chat: instructors can have "office hours" when they are available for real-time chat. Students also can use the real-time chat feature to collaborate with other students from the class. Chats can be designed to be available constantly, or they can be scheduled for specific times. If instructors aren't available, they still can go back and look at the chat log to identify areas that need more classroom discussion (Figure 4).

- Forum: each course may have a forum available for discussion. Unlike chat, the forum is not real time. Like any traditional forum, a user posts questions or discussion items and waits for responses.

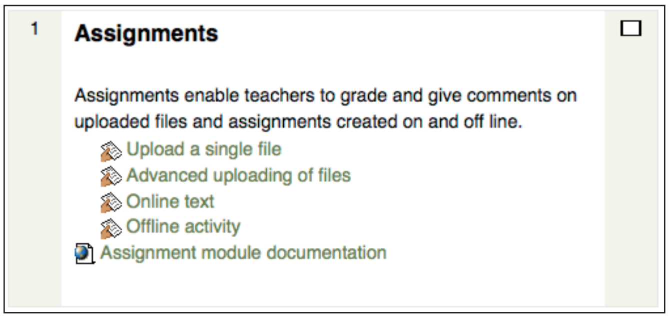
- Surveys and polls: Moodle offers instructors the opportunity to ask their students survey questions from within the course. These surveys could be used as an assessment for the teacher, as a way to gauge the level of understanding on an upcoming assignment, or to get feedback from students on any topic the teacher desires. They also can be set to anonymous so the students aren't identifiable, or they can be set to record user accounts with responses (Figure 5).

- Quizzes: Moodle quizzes are similar to surveys, but they are designed more specifically to be graded automatically. Quiz times can be specified to avoid at-home cheating, or they can be left open for students to take at their leisure. The multiple-choice nature of Moodle quizzes makes grading automatic, so students see their results immediately.

- Wikis: each course can have a wiki that students are able to edit collaboratively. This wiki might contain information gathered from multiple groups that needs to be shared with the class. Every enrolled user can edit the wiki, so it's truly collaborative.

- Workshops: assignments are set up as direct communication between students and instructors. With the workshop module, students can participate in peer reviews and assessments. Instructors can design specifically how students can respond to assignments in the workshop module, so although it is a peer assessment, the process still is guided by instructors. Figure 6 shows an example of a possible workshop, with some criterion for the users to assess.

- Site design: the classroom features mentioned above are the meat and potatoes of using Moodle. The Moodle course itself can be tweaked and modified as well, however, to make the site as useful as possible. The layout for the course page is designed using blocks. Anyone familiar with content management systems like Drupal or WordPress will understand the concept of blocks. Figure 7 shows the default layout of blocks in a newly created course. Blocks can be added, deleted and moved around. The function of available blocks varies from calendars to quiz results, and it includes the ability to embed Flickr photos or YouTube videos right into the Moodle page.



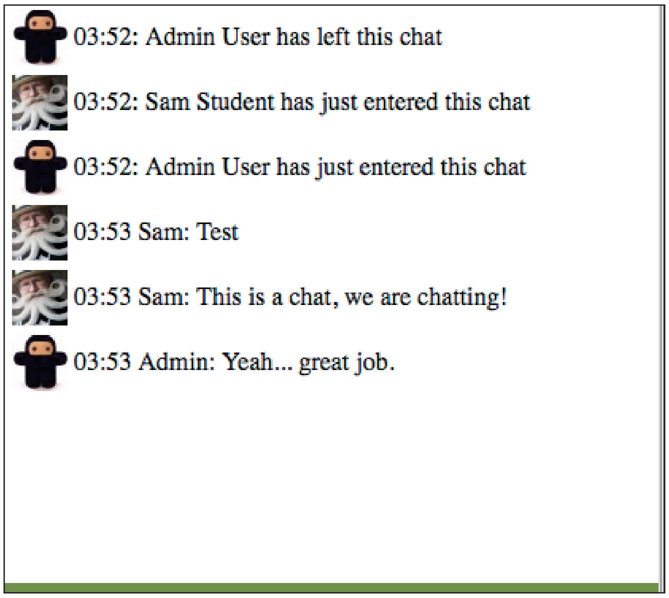
1 **Assignments**

Assignments enable teachers to grade and give comments on uploaded files and assignments created on and off line.

- Upload a single file
- Advanced uploading of files
- Online text
- Offline activity

Assignment module documentation

Figure 3. Assignments can be turned in using various methods (or none at all).



03:52: Admin User has left this chat

03:52: Sam Student has just entered this chat

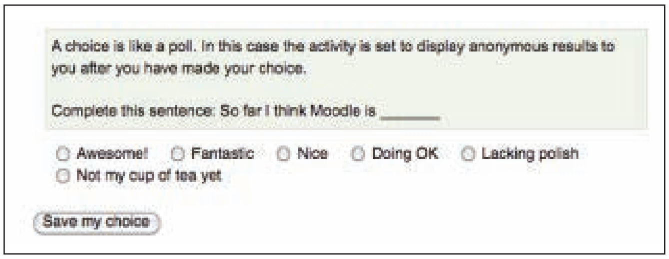
03:52: Admin User has just entered this chat

03:53 Sam: Test

03:53 Sam: This is a chat, we are chatting!

03:53 Admin: Yeah... great job.

Figure 4. Real-time chat enables students and instructors to discuss material, even if in-person classroom time isn't possible.

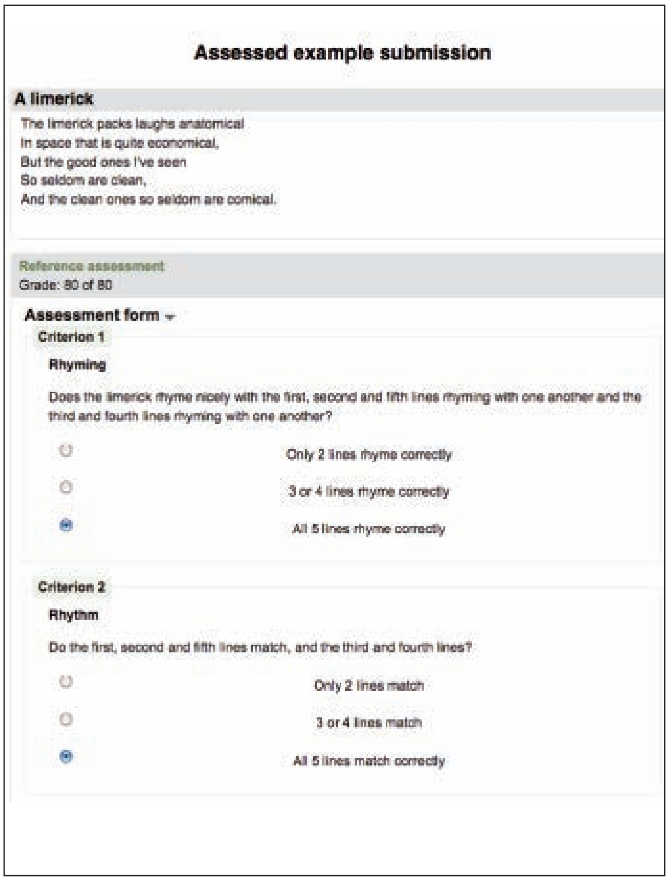


A choice is like a poll. In this case the activity is set to display anonymous results to you after you have made your choice.

Complete this sentence: So far I think Moodle is ______

○ Awesome! ○ Fantastic ○ Nice ○ Doing OK ○ Lacking polish
○ Not my cup of tea yet

Figure 5. Surveys either can be anonymous or tracked by user.



Assessed example submission

A limerick

The limerick packs laughs anatomical
In space that is quite economical,
But the good ones I've seen
So seldom are clean,
And the clean ones so seldom are comical.

Reference assessment
Grade: 80 of 80

Assessment form

Criterion 1

Rhyming

Does the limerick rhyme nicely with the first, second and fifth lines rhyming with one another and the third and fourth lines rhyming with one another?

○ Only 2 lines rhyme correctly
○ 3 or 4 lines rhyme correctly
● All 5 lines rhyme correctly

Criterion 2

Rhythm

Do the first, second and fifth lines match, and the third and fourth lines?

○ Only 2 lines match
○ 3 or 4 lines match
● All 5 lines match correctly

Figure 6. Peer assessments are guided by the instructor, but performed by other students in the course.

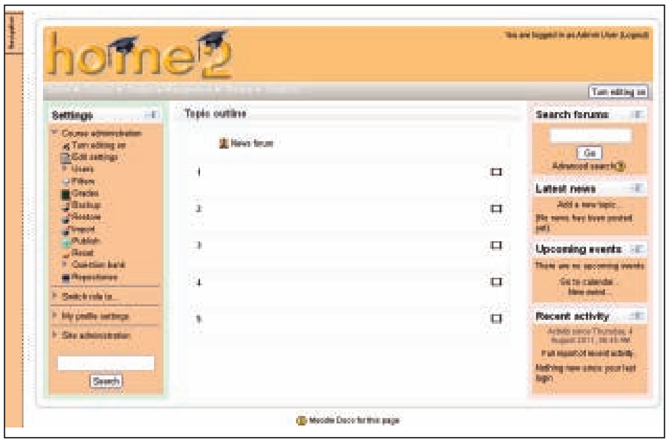


Figure 7. New courses start with this block layout, which can be modified by the instructor.

Installation

Installing Moodle is a breeze. The current stable version, at the time of this writing, is 2.2. In order to install it, your server needs to have PHP 5.3.2 or higher, MySQL 5.0.25 (or Postgres 8.3) and a Web server. That's really about it. Any recent server distro with a LAMP stack will be able to run Moodle pretty easily out of the box.

If you're not interested in installing Moodle onto a server, a few VM appliances also are available to download, although you might not find one with the latest version of Moodle installed. My recommendation for those desiring a VM of Moodle is to install a Linux VM, then install the most recent version of Moodle on top of it. It will take a little more time, but you'll have more control over the install, and you'll be assured of the most recent version.

The actual installation is pretty simple. You are required to create an empty database for Moodle to use and a data directory with write access. Once that is in place, simply extract the archive to your Web root, and point a browser to http://your.moodle.site/install.php. The installer will populate your database and walk you through initial configuration.

Managing Moodle

Moodle has tiers of management, so all the responsibility doesn't fall on a single administrator. Anyone logging in as administrator has full access to every aspect of the system, while individual users in the Manager role can be limited in what permissions they have. A few sysadmin items must be configured by whoever is in charge of those sorts of things—for example:

- Authentication: Moodle can be completely independent and store user login information in its own database. This has the advantage of working without interfacing with an established infrastructure, but it has the same as a disadvantage: it doesn't interface with the existing infrastructure. Thankfully, Moodle has extensive support for external authentication, and some of those things can be configured with minimal knowledge of the network. For example, Moodle supports authentication from IMAP or POP3 servers. That means any user with a valid e-mail account can log in to your Moodle server. Moodle also supports more complex authentication schemes like LDAP, and in Linux, it even supports PAM authentication from the local server.

- Course creation: administrators (or managers with the proper permissions) create a course and then assign an instructor to be in charge of it. Once a course is created, instructors can configure it in whatever way they desire, including adding content, changing layout and so on. Instructors also can decide how students can enroll in their courses. Students can be added manually; they can be given a code that allows them to join, or they even can gain access to a course via PayPal payment.

- Server management: this is the only part that really requires a sysadmin to manage. System backups, security settings, performance tweaks and other server-related tasks are done by the administrator. Some of these roles can be assigned to manager accounts, but at some point, the server maintenance leaves the realm of Moodle administration and enters the domain of the IT department. This responsibility line obviously will be different for every case. It's possible for Moodle to be centrally administered and accessed over the Internet for multiple locations. It's just as possible for teachers to run Moodle on their classroom workstations and administer it for their single class. The important thing is to determine in advance who will be in charge of what aspects of administration.

Let's Collaborate!

You've seen Moodle and what it can do, but the power of open source isn't just in free code. A huge community of Moodle users is providing plugins, pre-packaged assignment modules, themes and countless other enhancements for Moodle. Here are some great on-line resources for teachers interested in implementing and enhancing Moodle:

- Moodle Docs (**http://docs.moodle.org**): this Moodle-hosted wiki is a collaborative effort to document every aspect of using Moodle. Whether you're an administrator trying to configure a secure install or a teacher trying to add a quiz to your course, the Moodle Docs site is invaluable.

- Moodle Plugins (**http://moodle.org/plugins**): Moodle offers an extensive number of features in the standard install, but it also allows for third-party plugins to enhance courses. The official repository for plugins is hosted on the Moodle.org Web site. You can find pre-packaged assignments, alternate course layouts, visual themes and additional blocks to customize your Moodle site or individual course. Once you become a Moodle user, you may want to contribute back to the community as well.

- Course Exchange (**http://moodle.org/course/view.php?id=15**): many Web sites share complete Moodle courses, but this particular exchange is hosted on the Moodle.org servers. There are direct links to downloadable courses and lists of Web sites offering courses to download. In true "eat your own dog food" style, the Course Exchange is actually a Moodle course open for enrollment. You can enroll and participate in the Exchange.

- Moodle Share (**http://www.moodleshare.org**): this third-party Web site offers an extensive selection of courses available for download. Many of the courses were designed by Minnesota District 287, using funding from ARRA Grant Funds. Hats off to District 287 for sharing its resources! (Figure 8 shows a freely available lesson from Moodle Share.)

- Commercial Support (**http://www.moodle.com**): although Moodle itself is free in every sense of the word, there is a commercial side as well. That doesn't mean Moodle comes crippled or feature-stripped, it just means that commercial support is available if required. The moodle.com Web site offers links to vendors that can host a Moodle install, offer support for existing installs, train on the use of Moodle or even become certified in Moodle itself. Again, these commercial options don't offer any "Pro" features to Moodle; they're just available if customers need them.

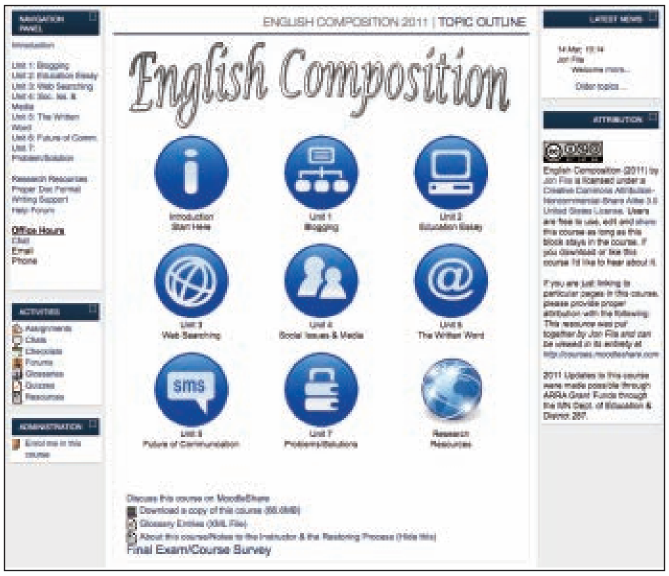


Figure 8. Moodle Share offers complete courses, available to download and use at no cost.

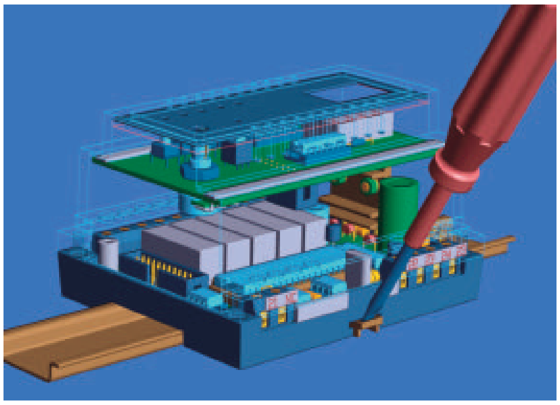
Finally, Let's Take Over the World!

The world we live in is one of on-line collaboration, 24-hour interaction and digital communication. Heck, you're reading this right now on a digital device, or you printed it from a digital source. Moodle is a perfect example of open-source ideals succeeding in regular, everyday life. The first step toward an Open Source world is to get kids excited about open-source tools. Hopefully, the next steps will take care of themselves.

Note: all images in this article are from ***http://www.moodle.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.

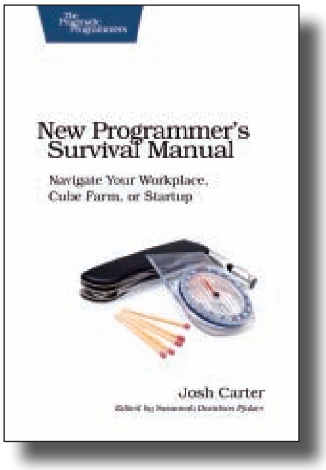
VariCAD, VariCAD Viewer



One of the most veteran CAD apps on the Linux platform, VariCAD, recently was upgraded to version 2012-1.0. The application is a 3-D/2-D CAD system for mechanical-engineering design, whose core features include tools for 3-D modeling and 2-D drafting and dimensioning; libraries of standard mechanical parts (ANSI, DIN); calculations of standard mechanical components; and tools for working with bills of materials (BOM) and blocks. Enhancements in version 2012-1.0 include significant improvements to the GUI, improved and rebuilt 2-D printing dialog, new 2-D dimensioning and editing of dimensions, improvements to DWG/DXF and STEP interfaces, a batch printing module and a better 3-D modeling kernel. Also included in the new update is VariCAD Viewer, a free combination viewer/converter/printing software for working with 2-D DWG, DXF, 3-D STEP and 2-D/3-D VariCAD file formats.
http://www.varicad.com

SecPoint Penetrator

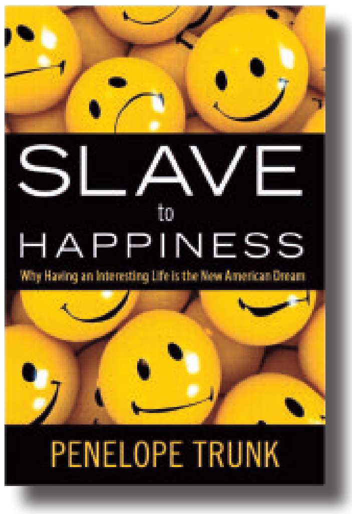
The SecPoint Penetrator, upgraded to version 9.9, is a vulnerability-scanning appliance that recently was beefed up with new applications to boost corporate security. The product helps companies prevent hackers from entering their servers and networks, as well as understand their vulnerabilities. The new version 9.9 now includes the Google Hacking Database to identify whether sensitive company information or files have been indexed by search engines, improved word lists for cracking (now more than 1.1 billion entries) and a tool to show whether a Wi-Fi network is vulnerable. SecPoint Penetrator has no limit on the amount of auditing nor a limit on the number of IP addresses.
http://www.secpoint.com



Josh Carter's *New Programmer's Survival Manual* (Pragmatic Programmers)

This one is for all of you recent comp sci grads! Think of Josh Carter's new book—*New Programmer's Survival Manual: Navigate Your Workplace, Cube Farm, or Startup*—like job insurance. The book is intended to introduce novice programmers to practices they'll find while working on large-scale, long-lived programs at a professional level of quality. They will find out how to work most efficiently with current tools and discover essential new ones. Other aspects include learning "street smarts", navigating the corporate working environment, collaborating with colleagues, fitting into the big picture and contributing to company success—thus, increasing one's job security. Finally, the title helps readers look ahead to the years to come to make plans for their long-term success as programming professionals, regardless of their area of focus.
http://www.pragprog.com

Penelope Trunk's *Slave to Happiness* (Que)



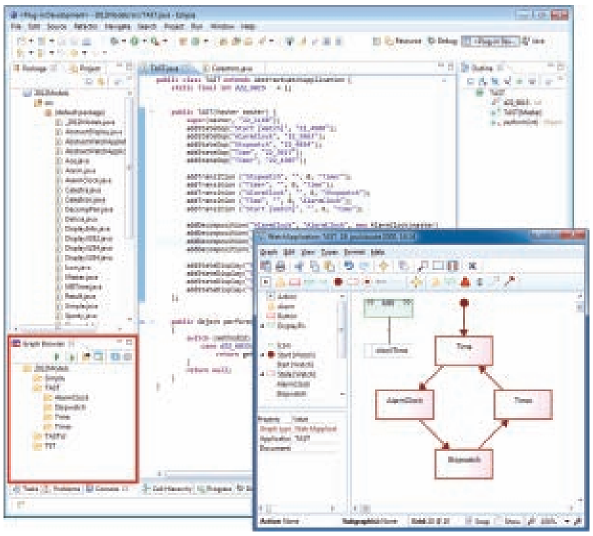
In her new book, *Slave to Happiness: Why Having an Interesting Life is the New American Dream*, author Penelope Trunk posits a societal trend that Linux folk can relate to. The new American dream, she says, is not based on the acquisition of more stuff, more money, more happiness or the like. Instead, a powerful new motivator replacing "more" for a new generation of Americans is the goal of an interesting life. Publisher Que calls Trunk's book a "powerful, revelatory book [that] captures a radical shift that's been taking place just below the surface of American culture: one that is about to explode across the American landscape". Some of her revelations include why Americans will increasingly choose interest over contentment, how to make choices when all of the world's knowledge is at your fingertips and how collaboration and connection are gaining a central place in the new American dream. Sound familiar? It will be interesting to see greater society adopt a philosophy that Linux community has embraced for nearly a generation.
http://www.informit.com

JetBrains' PhpStorm

Beside providing a delightfully psychedelic mental picture, JetBrains' PhpStorm, which is now in version 3.0, offers software developers an intelligent, productivity-enhancing, code-centric PHP IDE. The most noteworthy features of the new version include UML for quick view of the entire code structure plus the changes in the recent VCS commits, Smart Duplicated Code Detector to remove similar blocks of code without losing intended functionality, and a new PHPUnit test runner and JSTestDriver to make code run faster than ever. Other new functionality includes support for cutting-edge technologies, such as CoffeeScript and XSLT 2.0, an integrated XSLT debugger, PHP UML roundtrip diagrams, streamlined UI across all operating systems and Mac OS X Lin support, among others.
http://www.jetbrains.com

MetaCase's MetaEdit+ Plug-in for Eclipse

The case that MetaCase makes for its new MetaEdit+ Plug-in for Eclipse is that now development teams can combine the use of models and code—that is, modeling tools and programming environments—in the Eclipse environment. Until now, teams have utilized the MetaEdit+ domain-specific modeling environment on its own. The new MetaEdit+ Plug-in allows users to work with MetaEdit+ models directly from Eclipse, and it automatically imports source code generated from the MetaEdit+ models into Eclipse and builds and runs it there. Traditionally, development teams have found it difficult to combine the use of models and code due to the modeling languages used and the programming environments. Now, with the plugin, however, the domain-specific models—expressed as diagrams, matrices or tables in MetaEdit+—can be browsed and used directly from Eclipse.
http://www.metacase.com



Opengear IM4216-34

The Opengear team capped a productive 2011 with the release of Opengear IM4216-34, a high-capacity infrastructure management gateway that proactively monitors and maintains the health of all the IT equipment in data centers and remote locations. The IM4216-34 provides one point of control for managing all the servers, networking and power equipment in a rack. Provided are 16 serial ports for advanced console management, a 32-port Ethernet gateway for SNMP device management, three USB ports and 16GB of internal Flash memory. Product highlights include myriad enterprise-grade security technologies, Automated Remote Management and Support (ARMS) and remote rack-level management capabilities. Opengear says that the IM4216-34 allows overstretched IT and security organizations to get more results with fewer resources.
http://www.opengear.com

Paragon Universal File System Driver

The target market for the new Paragon Universal File System Driver (UFSD) is OEMs of consumer multimedia devices that are seeking cross-platform filesystem support. The software's creator, Paragon Software Group, says that if its Paragon UFSD, which supports all popular filesystems, is built in to digital devices, consumers can enjoy issue-free compatibility between filesystems used on all of their devices including those found not just in media centers, digital televisions and set-top boxes, but also various storage media. Key features include full read-write access, easy mounting of any NTFS, HFS+ or exFAT partition like a native one, non-Roman character support, low system requirements and no limitation on maximum file/partition size.
http://www.paragon-software.com

Paragon SOFTWARE GROUP

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.

Fresh from the Labs

Obix—Reliable Programming, Quickly

http://www.obix.lu

Do you want to create more reliable code in less time? Obix may be just what you need. However, I'm guessing this probably looks like just another programming language, but it's actually contained within a philosophy that may be an important evolutionary step in programming methods, called "Fail fast!"

Take note of that term. At first, it may seem bizarre, but on further inspection, the amusing name is actually very logical with a profound approach to programming. I explore this philosophy and its implications following the next section.

To quote some carefully chosen parts of the documentation:

> Obix is a high-level, object-oriented, statically-typed, compiled, portable programming language.
>
> The compiler generates Java binary code (.class files). Hence, applications written in Obix can run on any system with a Java Virtual Machine (JVM) installed, such as Linux, Mac OS, Windows and so on.
>
> Obix has been designed from the outset with a very specific main goal in mind, namely to help write more reliable code. In practice, this means that: 1) once your code passes compilation it contains less bugs and 2) remaining bugs are more quickly detected at runtime.
>
> As a result, you can expect to reduce development time and costs.

random_arcs is a cool tutorial that comes with Obix.

Installation Whether it be in development mode or production mode, Obix is designed around Java, so installing Java on your PC is a prerequisite. If you haven't done so already, install the Java JDK version 6.0 or later. The documentation notes that it's not enough to install the Java Runtime Environment (JRE); you need to have the Java Development Kit (JDK) installed.

If your machine is Debian/Ubuntu-based, the package name is sun-java7-jdk. If your machine is RPM-based, look for jdk-7u1-linux-x64.rpm or jdk-7u1-linux-i586.rpm. Either version 6 or 7 will do, but version 7 of Java is preferable. Other than that, there is nothing else in the way of dependencies.

To install Obix, grab the latest tarball from the Web site, extract it, and open a terminal in the new folder. From there, you now can check the installation with the following command that displays Obix's version number in your terminal:

```
$ ./obix.sh version
```

Usage As these are early days, Obix isn't supported by a graphical environment. All tools for managing projects and compiling, running and testing code are executed from the command line, so get a console and text editor ready.

So what was all the "Fail fast!" stuff about? The idea is this: *every coding error should be detected as early as possible, preferably at compile time, or else as early as possible at runtime.*

To achieve this, Obix has error-preventing concepts, such as:

- Contract programming (design by contract).
- Feature re-definition in child types.
- Unit testing.
- Generic types without type erasure at runtime.
- Objects are immutable by default.
- Void (null) values are not allowed by default.

Although this may seem like a hassle, Obix designer Christian Neumanns found that a forgiving approach to coding in a language resulted in some horrendous errors in the long run, with some very angry customers. However, start out with a "Fail fast!" approach, and your coding naturally will be tighter and more reliable.

To illustrate one of Obix's unique features for more reliable code, look at the following example source code:

```
int server_port = 8080
int age_of_my_elephant = server_port
```

Can you see the error? Of course you can. Can your compiler see the error? No, it can't. Although `server_port` and `age_of_my_elephant` obviously are two semantically different values, the compiler can't see an incompatibility, because both values are of type integer. Moreover, no runtime error will be generated, despite the fact that elephants can't be older than about 100 years.

You might say, "No programmer would ever write silly code like this!" This is true (for most programmers). However, look at this code:

```
int age_of_elephant = utilities.get_initial_value
```

Now, you can't tell if the assignment is okay just by looking at the source code. You would have to verify manually that `utilities.get_initial_value` actually returns an "age of elephant" value that can't be negative and can't exceed 100. It would be helpful if the compiler could check all this for you automatically, wouldn't it?

Without digging too deeply into how it works, here is the solution in Obix. You would define a specific type `age_of_elephant` and declare a positive maximum value of 100. The source code for this type looks like this:

```
type age_of_elephant

   inherit simple_positive32
      attribute value and_check: i_value <= 100~ end
   end

end
```

Then, you would declare a variable and assign it a value as follows:

```
var age_of_elephant age_of_my_elephant = ...
```

Now, the Obix compiler ensures that the right side of the assignment actually returns an `age_of_elephant` value (and not a "server port" value). And, it is impossible to assign an invalid value, such as -5 or 8080.

Many applications contain hundreds or thousands of integer variables with semantically different (and, therefore, incompatible) values. Thus, compile-time and runtime checks like the above are very important and effective, because any incompatibilities and illegal values are immediately and automatically reported. The same observation can be made for other data types, such as strings. For example, in Obix, you would define different types with specific validation rules for `customer_name`, `email_address` and `ISBN_number`.

Unfortunately, that's about as much as I can cover here. One area I would like to have highlighted is being able to insert Java script into Obix code (see Chapter 3 of the tutorial), but my favorite part is Chapter 4: "How to develop a standard PC application". Here you will see how Obix already has tools to develop projects, with their own compilation, tools, directory structures and so on—very cool.

In the end, Obix is built around some fantastic concepts, and I hope that it soon reaches a level of maturity where it's sitting in all major distro repositories. You also can help improve Obix by getting involved in the Open Source community or just by sending feedback to the author. Even outside of Obix, I'm hoping that Christian's "Fail fast!" ethos will take root in future programming languages, creating another evolutionary step in programming methods, and ultimately, more reliable computing.

darktable—Virtual Photo Darkroom

http://www.darktable.org

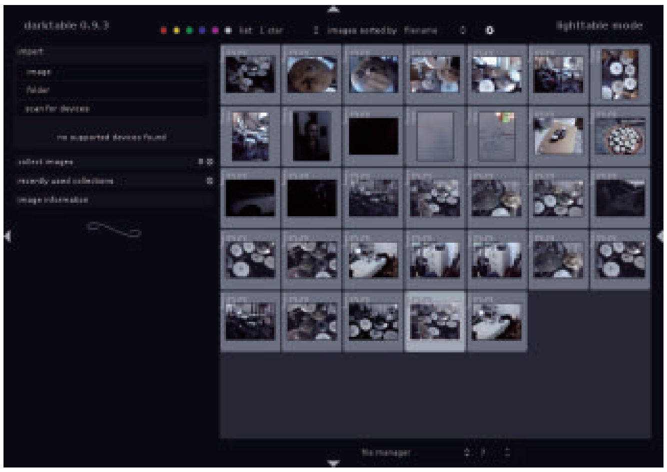
If you're a photographer dealing with many images at a time, darktable may well strike the perfect compromise between image browsing and picture enhancement.

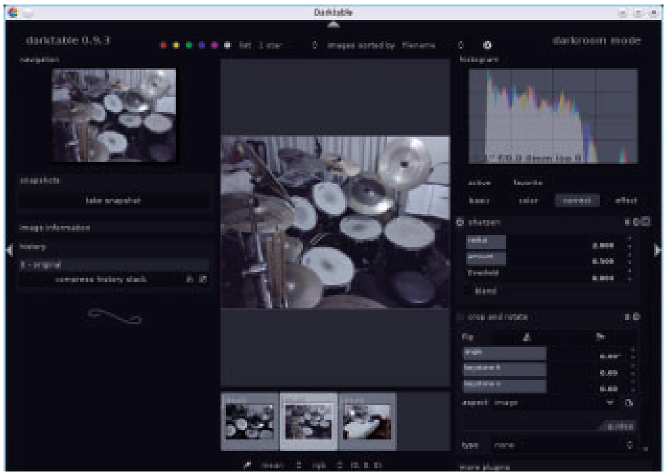
According to the Web site:

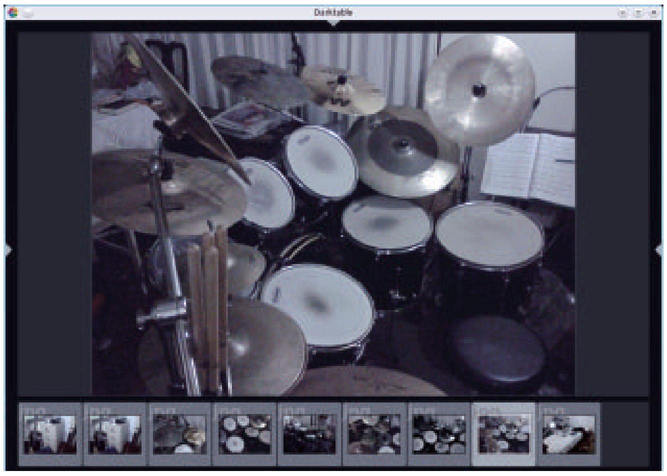
> darktable is an open-source photography work-flow application and RAW developer—a virtual lighttable and darkroom for photographers. It manages your digital negatives in a database, lets you view them through a zoomable lighttable and enables you to develop raw images and enhance them.
>
> ...darktable tries to fill the gap between the excellent existing free raw converters and image management tools (such as UFRaw, Rawstudio, F-Spot, digiKam and Shotwell). It focuses on the work flow to make it easier for the photographer to handle quickly the thousands of images a day of shooting can produce. It's also one of the very few FOSS projects able to do tethered shooting.

Installation Binary packages are provided for Ubuntu, Fedora, SUSE and Arch Linux, as well as a Portage package for Gentoo/Funtoo. There also is an experimental version for Mac OS X, and Windows also is a possibility, but amusingly, the Web site notes that the community for this commercial distribution hasn't made a native version yet, so it recommends installing Ubuntu instead.

In terms of library requirements, the Web site gives the following list of


darktable: a mass-photo browser with a strikingly good interface.


darktable with all the tools visible and ready for editing.



darktable with the tools cleared away to enjoy the picture in peace. Yes, those are my drums.

packages: libsqlite3, libjpeg, libpng, libraw (supplied), rawspeed (supplied), gtk+-2, cairo, libglade2, lcms2, exiv2, gconf, tiff, curl, gphoto2, dbus-glib, gnome-keyring, fop and openexr.

If you want to build from source, the Web site is incredibly helpful, with all the instructions you'll need, in great detail. Nevertheless, I include a simplified version here.

Download the latest tarball, extract it, and open a terminal in the new folder. Enter the command:

```
$ ./build.sh
```

Once the building has finished, change into the build folder with:

```
$ cd build
```

Now, to install darktable, if your distro uses sudo (such as Ubuntu), enter:

```
$ sudo make install
```

If your distro uses root, enter:

```
$ su
# make install
```

Note that on my machine, it installed darktable to /opt by default, in which case, at the command line, it may be required that you enter:

```
$ /opt/darktable/bin/darktable
```

For other users (especially with binaries), you can run the program with this command:

```
$ darktable
```

Usage As soon as you're inside the main window, you'll notice an arrow at each edge: this is for expanding and collapsing panes within the window. You may wonder why I'm starting with something so mundane, but this function actually is incredibly useful (more on that later).

Working within darktable, the interface is split into three view modes: lighttable (browsing), darkroom (editing) and camera tethering (a clever means of interacting with a camera). I don't really have the word space or equipment to explore the tethering mode, so I primarily cover the lighttable view and the darkroom view here.

lighttable When the program starts, you should find yourself in lighttable view. Here, you manage your images as well as do things like add tags and color labels. Keep in mind that its shortcut key is L, as darktable's design has been based around switching between the lighttable and darkroom views, easily and rapidly.

The first order of business is to import some images with the import field on the left.

Do you see a field? If not, remember the arrows that I mentioned at the start of this section. Click on the left arrow, and the import section will expand and collapse. Collapsing one section can be very useful, as it lets the actual picture section grow much larger. With the import section expanded, import some pictures (you have the choice of a singular image, a folder or to scan for devices), and they will appear in the main section in the middle.

While you're still here, try expanding and collapsing all four arrows, and see how it affects the rest of the window. With all the fields collapsed, the image section gets center stage and takes up the whole window, and with all the fields expanded, you have a great many tools available, but your images are reduced to small, insignificant squashy things. With just a small number of clicks, this clever GUI design allows you to switch between beautiful Zen-like minimalism on one end and ugly but useful functionalism on the other.

darkroom With that covered, select some of your pictures, and let's move on to the darkroom. Remember that its shortcut key is D. In fact, try alternating between lighttable and darkroom with L and D now. Nice, huh? This darkroom is where you do the editing, and you can do all sorts of things, like toy with the color profile, sharpen the image, apply cropping, rotate and so on.

While you're in the darkroom, you still can change between images with the filmstrip, toggled with Ctrl-F. Chances are that you'll have quite a number of images along the strip, so if you want to scroll along it, hover your mouse pointer over the strip and use your scroll wheel. Double-clicking will open an image.

That's about all the space I have for darktable, but it's been an absolute pleasure to use. The GUI design is one of the best I've ever come across, and I hope to see its design copied in the future. Plus, the aesthetics are great as well, with a really cool light-black color scheme that is such a nice change from the endless beige and grays that make computing so depressing at times.

For mid-level photo management, this probably is the coolest tool I've come across, and whether you like your interface minimal or maximal, it doesn't matter. You can change it around in about two seconds.■

John Knight is a 27-year-old, drumming- and bass-obsessed maniac, studying Psychology at Edith Cowan University in Western Australia. He usually can be found playing a kick-drum far too much.

BREWING SOMETHING FRESH, INNOVATIVE OR MIND-BENDING? Send e-mail to newprojects@linuxjournal.com.

An Introduction to Application Development with Catalyst and Perl

Rails isn't the only open-source Web framework in town.

HENRY VAN STYN

Catalyst is the latest in the evolution of open-source Web development frameworks. Written in modern Perl and inspired by many of the projects that came before it, including Ruby on Rails, Catalyst is elegant, powerful and refined. It's a great choice for creating any Web-based application from the simple to the very complex.

Like many other popular Perl-based projects, Catalyst has a strong focus on flexibility and choice. Catalyst is especially powerful because it provides an abundance of features and the core environment, structure and interfaces on which virtually anything can be built without forcing you to do things in any particular way.

Writing applications in Catalyst is fast too. Just because you can tackle any aspect of application design yourself, doesn't mean you have to. Catalyst provides a wide array of refined, high-level, drop-in solutions to all kinds of problems and needs without limiting access to the nuts and bolts. Templating, ORM, authentication, automatic session management and all the other high-level features you'd want from a Web framework are available in Catalyst—and more.

Catalyst's approach is to provide these high-level features as optional plugins and modules. This is one of the greatest strengths of Perl—a tremendous number of refined modules and libraries are available. So, instead of re-inventing all this functionality, Catalyst provides a framework to bring together seamlessly what already exists.

Catalyst is bigger than itself—it is also everything that's available in CPAN. That alone makes it one of the most feature-rich frameworks there are.

In this article, I provide an introduction to Catalyst and how to use it for rapid application development. I cover the basics of how to create and lay out a new application as well as how to write the actions that will handle requests. I explain how to define flexible URL dispatch logic and some of the APIs that are available. I focus on the fundamentals, but I cover some of the popular available components as well, such as Template::Toolkit. I also talk about how you can extend Catalyst itself, and how you can deploy an application with Apache.

Catalyst is bigger than itself—it is also everything that's available in CPAN.

Background Knowledge and the MVC Architecture

Catalyst and Catalyst applications are written in Perl, so some basic

Perl knowledge is necessary to use Catalyst effectively. You also should have some experience with object-oriented programming concepts, such as classes, methods, inheritance and so on.

Like Rails, Django, CakePHP and many other Web frameworks, Catalyst follows the venerable Model-View-Controller architectural pattern. MVC is a proven approach to structuring and segmenting application code for efficiency, flexibility and maintainability.

Plenty of tutorials and resources are available for MVC, so I won't spend too much time covering it here. If you've worked with other Web frameworks, chances are you're already familiar with MVC. If not, the most important thing to understand is that it is more about best practices than anything else.

The focus of this article is to explain the core details of how Catalyst operates, but since Catalyst made most of its layout decisions according to MVC, you'll still see it along the way.

Getting Catalyst

Before you can install Catalyst on your system, you obviously need Perl. Most Linux distros already have Perl installed out of the box, but if not, install it with your package manager.

Catalyst itself is a Perl library that you can install with cpan:

```
cpan Catalyst::Devel
```

The previous command installs Catalyst with development tools along with its many dependencies. For production/hosting systems that will run only applications without the need for development tools, you can install the smaller Catalyst::Runtime bundle instead.

Because Catalyst has so many dependencies, it can take quite a while to install on a fresh system. By default, CPAN asks if it should install each dependency individually, which can become redundant really quick. You can configure CPAN not to ask, but instead, I usually just cheat by holding down Enter for a few seconds to queue up a bunch of default ("yes, install the module!") keystroke/answers.

If the install fails on the first attempt, don't fret. Whatever the problem may be, it probably will be explained in the scrollback along with what to do to solve it. Typically, this involves nothing more than installing/upgrading another module that wasn't automatically in the dependency tree for whatever reason, or just running the `cpan` command a second time.

Catalyst Application Layout

Every Catalyst application is a Perl module/library/bundle—exactly like the modules on CPAN. This consists of a package/class namespace and standard structure of files and directories. The Catalyst::Devel package comes with a helper script to create new "skeleton" applications and to initialize the files and directories for you. For example, to create a new application called KillerApp, run the following:

```
catalyst.pl KillerApp
```

This creates a new application structure at KillerApp/ with the following subdirectories:

lib/: this is the Perl include directory that stores all the Perl classes (aka packages or modules) for the application. This is added to the Perl lib path at runtime, and the directory structure corresponds to the package/class namespaces. For example, the two classes that are initially created have the following namespaces and corresponding file paths:

- KillerApp — lib/KillerApp.pm
- KillerApp::Controller::Root — lib/KillerApp/Controller/Root.pm

These directories also are created but initially are empty:

- lib/KillerApp/Model/
- lib/KillerApp/View/

root/: this is where other kinds application-specific files are stored. Static Web files, such as images, CSS and JavaScript go in the subdirectory static, which usually is exposed as the URL /static. Other kinds of files go in here too, such as templates.

script/: this contains application-specific scripts, including the development server (killerapp_server.pl) that you can use to run the application in its own standalone Web server, as well as scripts to deploy the application in a "real" Web server. The helper script killerapp_create.pl creates new model, view and controller component classes.

t/: this is where "tests" go. If you follow a test-driven development process, for every new feature you write, you also will write an automated test case. Tests let you quickly catch regressions that may be introduced in the future. Writing them is a good habit to get into, but that's beyond the scope of this article.

The created skeleton application is already fully functional, and you can run it using the built-in test server:

```
cd KillerApp/
script/killerapp_server.pl
```

This fires up the app in its own dedicated Web server on port 3000. Open http://localhost:3000/ to see the default front page, which initially displays the Catalyst welcome message.

The Request/Response Cycle

All Web applications handle requests and generate responses. The fundamental job of any Web framework/platform/

environment is to provide a useful structure to manage this process. Although there are different ways of going about this—from elegant MVC applications to ugly, monolithic CGI scripts—ultimately, they're all doing the same basic thing:

1. Decide what to call when a request comes in.

2. Supply an API for generating the response.

In Catalyst, this happens in special methods called "actions". On every request, Catalyst identifies one or more actions and calls them with special arguments, including a reference to the "context" object that provides a convenient and practical API through which everything else is accomplished.

Actions are contained within classes called "controllers", which live in a special path/namespace in the application (lib/KillerApp/Controller/). The skeleton application sets up one controller ("Root"), but you can create more with the helper script. For example, this creates a new controller class KillerApp::Controller::Something:

```
script/killerapp_create.pl controller Something
```

The only reason to have more than one controller is for organization; you can put all your actions in the Root controller with no loss of features or ability. Controllers are just the containers for actions.

In the following sections, I describe how Catalyst decides which actions to call on each request ("dispatch") and then explain how to use the supplied context object within them.

Dispatch

Catalyst provides a particularly flexible and powerful mechanism for configuring dispatch rules. Rather than having a separate configuration to assign URLs to specific actions, Catalyst uses the actions themselves to determine URL mappings dynamically.

Each action definition (which is just a Perl subroutine) represents not only a block of code, but also what URL paths apply to it. This is specified in *subroutine attributes*—a lesser-known Perl feature that provides arbitrary labels that can be used for introspection.

Catalyst supports a handful of parameterized attributes to determine the URL path to action mappings in a variety ways. For example, the following action has an absolute path set using the `:Path` attribute:

```
sub myaction :Path('/some/place') {
        my ( $self, $c, @args ) = @_;
        # do stuff...
}
```

Regardless of what controller you

put it in, the above action would map to all URLs starting with /some/place (http://localhost:3000/some/place with the development server).

If you omitted the starting slash and used `:Path('some/place')`, the action would map to a path relative to the namespace of the controller. For example, if it were in `KillerApp::Controller::Foobar`, it would be mapped to URL paths starting with /foobar/some/place.

Instead of using `:Path` to set the path explicitly, you can set `:Local` to use the name of the controller and method. For instance, the following action, if contained in the controller `KillerApp::Controller::Some`, would also map to /some/place:

```
sub place :Local {
        my ( $self, $c, @args ) = @_;
        # do stuff...
}
```

If it were contained in the controller `KillerApp::Controller::Some::Other`, it would map to /some/other/place.

Actions include subpaths by default, so the above also would match /some/other/place/blah/foo/1. When this happens, the leftover parts of the path are supplied as arguments to the action method ('blah','foo','1'). You can use the `:Args` attribute to limit how deep the action will match subpaths, if at all. With an Args value of 0, this action would match only /some/place, but nothing below it:

```
sub myaction :Path('/some/place') :Args(0) {
        my ( $self, $c ) = @_;
        # do stuff...
}
```

Other attributes are available too. `:Global` works like `:Local` but ignores the controller name, and path pattern matching can be done with `:Regex` and `:LocalRegex`.

When a URL matches more than one action, Catalyst picks the one that matches best. However, there are a few built-in actions (method names "begin", "end" and "auto") that, if defined, are called at various stages of every request in addition to the matched action. Using the advanced `:Chained` attribute type, you can configure additional/multiple actions to be called with single requests in any order you like.

You also can programmatically dispatch to other action/paths from within the action code itself:

```
sub myaction :Path('/some/place') {
        my ( $self, $c, @args ) = @_;
        $c->forward('/some/other/place');
}
```

The Context Object ($c)

Controller actions serve as entry points to application code. A special per-request object called the "context" is supplied as an argument to every action when it is called by the dispatcher. The context object typically is read into a variable named `$c`, but it could be called anything.

The context provides interfaces and information about the application and its current state. It contains the details of the request currently being processed (`$c->request`) and access to what will become the response (`$c->response`).

At the beginning of the request, before any actions are called, the response object is created with empty/default data. Each of the actions that are called then has an opportunity to manipulate the response. At the end of the request, its final state is sent back to the client. This iterative approach to generating the response lends itself to a modular and dynamic structure.

The following action illustrates a few of the simple APIs that are available, such as inspecting the User-Agent and query/post parameters in the request, and setting the body and headers of the response:

```
sub myaction :Path('/some/place')  {
    my ( $self, $c, @args ) = @_;

    my $myparam = $c->request->params->{myparam};

    if(defined $myparam) {
        $c->response->body("myparam is $myparam");
    }
    else {
        $c->response->body("myparam was not supplied!!");
    }

    $c->response->body(
        $c->response->body .
        "\n\nExtra path args: " . join('/',@args)
    ) if (@args > 0);

    $c->response->headers->header( 'Content-Type' => 'text/plain' );

    $c->response->body("Bad command or file name")
        if ($c->request->user_agent =~ /MSIE/);
}
```

Accessing the URL http://localhost:3000/some/place/boo/baz?myparam=foo would display the text that follows (except when using IE, in which case "Bad command or file name" is displayed instead):

```
myparam is foo

Extra path args: boo/baz
```

Within the action code, you can write any logic you like to build your application. Because the context object is just a variable, you can pass it as an argument into other functions. Following normal Perl programming rules, you can use other classes and libraries, instantiate objects and so on.

This is the extent of what you *have* to do—write controller actions and use the context object—but it's only the beginning of what you *can* do.

Over and above the core functionality, Catalyst provides a robust MVC structure to build your application.

Catalyst Components

Over and above the core functionality, Catalyst provides a robust MVC structure to build your application. This includes useful base classes, sensible default behaviors and helpful sugar functions. You can leverage a tremendous amount of turnkey functionality by creating your classes within the supplied framework as models, views and controllers.

All these are considered "components" within Catalyst, and there aren't actually many functional differences between them. The Model-View-Controller monikers primarily are used for the purpose of categorization. Models are meant to contain data and business logic; views are supposed to handle rendering and display; and controllers tie everything together.

Operationally, components are essentially application classes with some extra, Catalyst-specific functionally. They are loaded automatically at startup as static object instances. Any component can be accessed throughout the application via the context object:

```
sub myaction :Path('/some/place') {
        my ( $self, $c, @args ) = @_;
        $c->model('MyModel')->do_something;
        $c->forward( $c->view('MyView') );
}
```

In the above example action, you simply are calling the method `do_something` in a model named `MyModel` (KillerApp::Model::MyModel), and then `forward` to the view named `MyView` (KillerApp::View::MyView).

Earlier, I showed how to use `forward` to dispatch to another action by supplying a path. When you pass a component to `forward`, the `process` method of the supplied component is called as if it were a controller action, which is roughly equivalent to this:

```
$c->view('MyView')->process($c,@args);
```

These are just a few examples of the available conventions and shortcuts. The important thing to understand is that all these sugar functions just boil down to calling methods and normal program flow.

Template::Toolkit Views

One of the most common needs of a Web application is a templating system for rendering content. Templates probably are the best all-around approach to rendering text-based content, especially for markup like HTML.

Catalyst can use multiple Perl template systems, but the most popular is Template::Toolkit—a powerful, general-purpose template-processing system that is fast and feature-rich. It has a versatile and robust syntax that is simple and easy to use, but it also supports advanced capabilities, such as control structures, stream processing and extendability. Template::Toolkit is a whole programming language in its own right.

Catalyst provides a drop-in interface to Template::Toolkit via the view/component class Catalyst::View::TT. You can create a view within your application that extends this class using the helper script. Run this to create a new view named "HTML":

```
script/killerapp_create.pl view HTML TT
```

The new view is fully functional out of the box. As a general-purpose wrapper around Template::Toolkit, it provides a simple API to select templates and supply input data. The rest of the view-specific code goes in the templates themselves, which are stored within "root" in your application directory.

Here is an example of a simple Template::Toolkit template to render an HTML page:

```
<html><head>
<title>[% title %]</title>
</head><body>
<h1>Message: [% message %]</h1>
</body>
</html>
```

The character sequences within [% %] are "directives"—snippets of code that get replaced when the template is processed. The directives above are simple variable substitutions, which are the most basic kind. In this case, the values supplied for `title` and `message` will be inserted when the template is rendered.

If you saved the above template in a file named main.tt within root/templates, for example, you could use it with an action like this:

```
sub myaction :Path('/some/place') :Args(0) {
        my ( $self, $c ) = @_;

        $c->stash->{template} = 'templates/main.tt';
        $c->stash->{data}->{title} = 'TT rendered page';
        $c->stash->{data}->{message} = 'A cool message!';

        $c->forward( $c->view('HTML') );
}
```

The `stash` object above is another built-in feature of Catalyst that I haven't covered so far. It isn't very complicated; it's simply a hashref within the context object. It provides a standard per-request place to share data across components, similar to request and response, but for general use.

Catalyst::View::TT-based views use the content of the stash to determine operation. The value of `template` identifies the template to call, and the stash as a whole is used as the input data—each key in the stash becomes a variable in the template. The content generated from processing the template is used to set the body of the response.

The data in a real application probably will be more complex than the simple key/values in the previous example. One of Template::Toolkit's great features is its ability to handle Perl data structures directly. Consider the following action:

```
sub myaction :Path('/some/place') :Args(0) {
        my ( $self, $c ) = @_;

        $c->stash->{template} = 'templates/main.tt';

        $c->stash->{data} = {
                title  => 'TT rendered page',
                subhash => {
                        alpha => 'foo',
                        bravo => 'abdc',
                        charlie => 'xyz'
                },
                thinglist => [
                        'Thing 1',
                        'Thing 2',
                        'Big thing',
                        'Small thing'
                ]
        };

        $c->forward( $c->view('HTML') );
}
```

This would work with a template like this:

```
<html><head>
<title>[% data.title %]</title>
</head><body>
<b>Alpha:</b> [% data.subhash.alpha %]<br>
<b>Bravo:</b> [% data.subhash.bravo %]<br>
<b>Charlie:</b> [% data.subhash.charlie %]<br>
<br>
<b>List of Things:</b><br>
[% FOREACH item IN data.thinglist %]
        [% item %]<br>
[% END %]
</body>
</html>
```

Objects also can be supplied and accessed in the same manner as hashes. In fact, the context object is supplied automatically in "c". For example, if you want to display the client's IP address, instead of separately putting `$c->request->address` in the stash, you can just access it directly within the template like this:

```
[% c.request.address %]
```

Template::Toolkit has many more features and abilities, including wrappers, conditional statements, filters, function calls and so on. Catalyst::View::TT also has additional defaults and configuration options that I didn't cover here (see the documentation for more details).

It is totally up to you how to balance logic between the templates and the rest of your application. Depending on what you are trying to achieve, your application easily could be written more in Template::Toolkit than in Perl!

DBIx::Class Models

One of the other most common needs of an application is a database. DBIx::Class (often shortened to DBIC) has emerged as the most popular ORM (Object Relational Mapper) library available for Perl. It is an exceptionally powerful, robust, object-oriented interface to many relational database servers (including MySQL, PostgreSQL, Oracle, MSSQL and many others).

Like Template::Toolkit, but to an even greater degree, Catalyst provides refined, drop-in component wrappers to interface with DBIx::Class (Catalyst::Model::DBIC::Schema).

Using DBIx::Class is a whole topic in and of itself that I don't have space to cover here, but it is a must-have if you plan to integrate your application with a database. See Resources for information on where to go to start learning about this wonderful library.

Plugins and Application-Wide Settings

Besides pre-built component classes for drop-in functionality, many plugins are available to modify the behavior and extend the functionality of Catalyst itself. A few of the most common are the optional authentication, authorization and session plugins.

These plugins provide a consistent API for handling these tasks with a variety of available back ends. Like the core request/response object interfaces, they are available as application-wide features that are accessed and controlled through methods in the context object, which become available once these plugins have been loaded.

You can authenticate a user (within an action handling a login form post, for example) like this:

```
$c->authenticate({
   username => $c->request->params->{username},
   password => $c->request->params->{password}
});
```

If this succeeds, the `$c->user` object is available in subsequent requests to control and direct application flow based on the authenticated user. This is accomplished using sessions (usually cookie-based) that are handled for you automatically. You also have access to `$c->session` to persist any additional per-session data across requests.

The API of this framework is agnostic to the back end, and many are available. You can handle authentication and user storage via databases (DBIC), system accounts, PAM and LDAP, to name a few. There also are multiple ways to handle session data to support different application needs, such as distributed server deployments and so on. (See the documentation for Catalyst::Plugin::Authentication, Catalyst::Plugin::Authorization and Catalyst::Plugin::Session for more information.)

Plugins and application-wide settings are configured within the main/core class (lib/KillerApp.pm). Within this file, you can specify global configuration parameters, load Plugins and even add your own code to override and extend core functionality.

The top-level "KillerApp" class actually *is* the application—it programmatically loads and integrates the other components and classes throughout the system. Like any derived class, its behaviors can be selectively altered from that of its parent class ("Catalyst"). Since it uses the powerful "Moose" object system, in addition to adding and replacing methods, you also can take advantage of additional powerful features like method modifiers and Roles (in fact, Plugins are essentially Moose Roles applied to this class).

Catalyst was written with customization and extensibility in mind. It's structured to allow its functions and behaviors to be modified easily in a fine-grained manner.

For example, you could configure every response to be set with "no-cache" across the whole application simply by adding a method modifier like this to lib/KillerApp.pm:

```
before 'finalize_headers' => sub {
       my $c = shift;
       $c->response->headers->header( 'Cache-control' => 'no-cache' );
};
```

Catalyst calls methods with meaningful names (such as 'finalize_headers') throughout the various stages of processing that you are free to hook into or override.

Deploying Your Application

Like most things in Catalyst, many options are available when you're ready to deploy your application to a real Web server—Apache/FastCGI is one of the best choices available. I briefly cover this below.

If you put your application in /var/www, for example, you can deploy with an Apache virtual host configuration like this:

```
<VirtualHost *:80>
    ServerName www.example.com
    ServerAdmin webmaster@example.com

    Alias /static/ /var/www/KillerApp/root/static/

    FastCgiServer /var/www/KillerApp/script/killerapp_fastcgi.pl \
        -processes 5

    Alias / /var/www/KillerApp/script/killerapp_fastcgi.pl/

    ErrorLog /var/www/logs/error_log
    CustomLog /var/www/logs/access_log combined

</VirtualHost>
```

FastCGI is a proven, language-independent interface for running Web applications. It is essentially just plain CGI, but it keeps programs running in the background instead of firing them up for every request. This is the major limitation of CGI that FastCGI overcomes. FastCGI has been around for a long time. The use of this efficient protocol is another example of how Catalyst leverages existing solutions.

FastCGI allows you to specify the number of processes to run (five in the example above), making your application multithreaded. Incoming requests are distributed evenly among the processes, which are maintained in a pool.

The alias for `/static/` above tells Apache to serve the files directly in this directory (images, CSS, JavaScript files and so on). This is more efficient than serving these files through the application, which isn't necessary.

Conclusion

This article is meant to provide only a taste of Catalyst and its capabilities. As I hope you have seen, Catalyst is a viable platform for any Web development project. With a flexible design and many available mature features, you can use Catalyst to build robust applications quickly and conveniently.

Catalyst is actively developed and is getting better all the time, including its ever-increasing quality documentation. It also has a very active user community with top experts available via IRC.

When you're ready to start writing an application, you should be able to find the information and support you need to hit the ground running. See the Resources for this article for important links and where to get started.■

Henry Van Styn is the founder of IntelliTree Solutions, an IT consulting and software development firm located in Cincinnati, Ohio. Henry has been developing software and solutions for more than ten years, ranging from sophisticated Web applications to low-level network and system utilities. He is the author of Strong Branch Linux, an in-house server distribution based on Gentoo. Henry can be contacted at www.intellitree.com.

Resources

Catalyst Home Page: **http://www.catalystframework.org**

Catalyst::Manual: **http://search.cpan.org/perldoc?Catalyst::Manual**

Template Toolkit Home Page: **http://www.template-toolkit.org**

DBIx::Class::Manual: **http://search.cpan.org/perldoc?DBIx::Class::Manual**

Catalyst IRC Channel: #catalyst on **http://irc.perl.org**

"Moose" by Henry Van Styn, *LJ*, September 2011: **http://www.linuxjournal.com/content/moose**

The Definitive Guide to Catalyst (c) 2009 ISBN: 978-1-4302-2365-8 (print) and 978-1-4302-2366-5 (on-line).

FAST '12

10th USENIX Conference on File and Storage Technologies

sponsored by USENIX
in cooperation with ACM SIGOPS

FEBRUARY 14—17, 2012 | SAN JOSE, CA

FAST '12 brings together storage system researchers and practitioners to explore new directions in the design, implementation, evaluation, and deployment of storage systems. The program includes:

Tutorial Sessions

- Brent Welch on Clustered and Parallel Storage System Technologies
- Jeff Darcy on Building a Cloud Storage System
- Rich Freitas and Larry Chiu on Storage Class Memory
- Jiri Schindler on Understanding the I/O of Columnar and NoSQL Database

Technical Sessions

- Refereed papers on topics such as flash and SSDs, mobile storage, cloud, and more
- Two poster sessions
- Work-in-Progress Reports

REGISTER BY JAN. 23 AND SAVE!

ADDITIONAL DISCOUNTS AVAILABLE!

STAY CONNECTED:

FACEBOOK.COM/USENIXASSOCIATION TWITTER.COM/USENIX #FAST12

Three Ways to Web Server Concurrency

Multiprocessing, multithreading and evented I/O: the trade-offs in Web servers.

MARTIN KALIN

A Web server needs to support concurrency. The server should service clients in a timely, fair manner to ensure that no client starves because some other client causes the server to hang. Multiprocessing and multithreading, and hybrids of these, are traditional ways to achieve concurrency. Node.js represents another way, one based on system libraries for asynchronous I/O, such as epoll (Linux) and kqueue (FreeBSD). To highlight the trade-offs among the approaches, I have three echo servers written in close-to-the-metal C: a forking_server, a threading_server and a polling_server.

Shared Code

The Web servers use utils.c (Listing 1). The function `error_msg` prints messages and optionally terminates the server; `announce_client` dumps information about a connection; and `generate_echo_response` creates a syntactically correct HTTP response.

The central function is `create_server_socket`, which creates a blocking or a nonblocking listening socket. This function invokes three system functions:

- `socket` — create socket.
- `bind` — set port.
- `listen` — await connections.

The first call creates a TCP-based socket, and the bind call then specifies the port number at which the Web server awaits connections. The listen call starts the waiting for up to `BACKLOG` connections:

```
if (listen(sock, BACKLOG) < 0) /* BACKLOG == 12 */
  error_msg("Problem with listen call", true);
```

A Multiprocessing Server

The forking_server in Listing 2 supports concurrency through multiprocessing, an approach that early Web servers, such as Apache 1 used to launch Web applications written as, for example, C or Perl scripts. Separate processes

Listing 1. utils.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <fcntl.h>
#include "utils.h"

void error_msg(const char* msg, bool halt_flag) {
    perror(msg);
    if (halt_flag) exit(-1);
}

/* listening socket */
int create_server_socket(bool non_blocking) {
  /* Modify as needed. */
  const int port = 3000;

  struct sockaddr_in server_addr;

  /* create, bind, listen */
  int sock = socket(AF_INET,     /* family */
                    SOCK_STREAM, /* TCP */
                    0);
  if (socket < 0) error_msg("Problem with socket call", true);

  /* non-blocking? */
  if (non_blocking) fcntl(sock, F_SETFL, O_NONBLOCK);

  /* bind */
  bzero(&server_addr, sizeof(server_addr));
  server_addr.sin_family = AF_INET;
  server_addr.sin_addr.s_addr = INADDR_ANY;
  server_addr.sin_port = htons(port); /* host to network endian */
  if (bind(sock, (struct sockaddr*) &server_addr,
   ➥sizeof(server_addr)) < 0)
    error_msg("Problem with bind call", true);

  /* listen */
  fprintf(stderr, "Listening for requests on port %i...\n", port);
  if (listen(sock, BACKLOG) < 0)
    error_msg("Problem with listen call", true);

  return sock;
}

void announce_client(struct in_addr* addr) {
  char buffer[BUFF_SIZE + 1];

  inet_ntop(AF_INET, addr, buffer, sizeof(buffer));
  fprintf(stderr, "Client connected from %s...\n", buffer);
}

void generate_echo_response(char request[ ], char response[ ]) {
  strcpy(response, "HTTP/1.1 200 OK\n");
  strcat(response, "Content-Type: text/*\n");
  strcat(response, "Accept-Ranges: bytes\n");
  strcat(response, "Connection: close\n\n");
  strcat(response, request);
}
```

Listing 2. forking_server.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <netinet/in.h>
#include <signal.h>
#include "utils.h"

int main() {
  /* Avoid zombies. */
  signal(SIGCHLD, SIG_IGN);

  char buffer[BUFF_SIZE + 1];

  struct sockaddr_in client_addr;
  socklen_t len = sizeof(struct sockaddr_in);

  /* listening socket */
  int sock = create_server_socket(false);

  /* connections + requests */
  while (true) {
     int client = accept(sock,
                         (struct sockaddr*) &client_addr,
                         &len);
    if (client < 0) error_msg("Problem with accept call", true);

    announce_client(&client_addr.sin_addr);

    /* fork child */
    pid_t pid = fork();
    if (pid < 0) error_msg("Problem with fork call", false);

    /* 0 to child, child's PID to parent */
    if (0 == pid) {  /** child **/
      close(sock);   /* child's listening socket */

      /* request */
      bzero(buffer, sizeof(buffer));
      int bytes_read = recv(client, buffer, sizeof(buffer), 0);
      if (bytes_read < 0) error_msg("Problem with
          ➥recv call", false);

      /* response */
      char response[BUFF_SIZE * 2];
      bzero(response, sizeof(response));
      generate_echo_response(buffer, response);
      int bytes_written = send(client, response,
        ➥strlen(response), 0);
      if (bytes_written < 0) error_msg("Problem with
         ➥send call", false);

      close(client);
      exit(0);       /* terminate */
    }
    else             /** parent **/
      close(client); /* parent's read/write socket. */
  }

  return 0;
}
```

handle separate connections. This approach is hardly outdated, although modern servers, such as Apache 2, typically combine multiprocessing and multithreading.

The `forking_server` divides the labor among a parent process and as many child processes as there are connected clients. A client is active until the connection closes, which ends the session.

The parent process executes `main` from the first instruction. The parent listens for connections and per connection:

- Spawns a new process to handle the connection.
- Resumes listening for other connections.

The following is the critical code segment:

```
pid_t pid = fork(); /* spawn child */
if (0 == pid) {     /* child */
   close(sock);     /* close inherited listening socket */
   /* handle request and terminate */
   ...
}
else                /* parent */
  close(client);    /* close client, resume listening */
```

The parent executes the call to fork. If the call succeeds, fork returns a non-negative integer: 0 to the forked child process and the child's process identifier to the parent. The child inherits the parent's open

Creating and destroying processes are expensive. Modules, such as FastCGI, remedy this inefficiency through pre-forking.

socket descriptors, which explains the if-else construct:

- if clause: the child closes its copy of the listening socket because accepting clients is the parent's job. The child handles the client's request and then terminates with a call to exit.

- else clause: the parent closes the client socket because a forked child handles the client. The parent resumes listening for connections.

Creating and destroying processes are expensive. Modules, such as FastCGI, remedy this inefficiency through pre-forking. At startup, FastCGI creates a pool of reusable client-handling processes.

An inefficiency remains, however. When one process preempts another, a context switch occurs with the resultant system working to ensure that the switched-in and switched-out process behaves properly. The kernel maintains per-process context information so that a preempted process can restart. The context's three main structures are:

- The page table: maps virtual addresses to physical ones.

- The process table: stores vital information.

- The file table: tracks the process' open files.

The CPU cycles that the system spends on context switches cannot be spent on applications such as Web servers. Although measuring the latency of a context switch is nontrivial, 5ms–10ms per switch is a ballpark and even optimistic range. Pre-forking mitigates the inefficiency of process creation and destruction but does not eliminate context switching.

What good is multiprocessing? The process structure frees the programmer from synchronizing concurrent access to shared memory locations. Imagine, for example, a Web application that lets a user play a simple word game. The application displays scrambled letters, such as "kcddoe", and a player tries to unscramble the letters to form a word—in this case, "docked". This is a single-player game, and the application must track the game's state: the string to be unscrambled,

Listing 3. threading_server.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <netinet/in.h>
#include <signal.h>
#include <pthread.h>
#include "utils.h"

/* thread routine */
void* handle_client(void* client_ptr) {
  pthread_detach(pthread_self()); /* terminates on return */

  /* read/write socket */
  int client = *((int*) client_ptr);

  /* request */
  char buffer[BUFF_SIZE + 1];
  bzero(buffer, sizeof(buffer));
  int bytes_read = recv(client, buffer, sizeof(buffer), 0);
  if (bytes_read < 0) error_msg("Problem with recv call", false);

  /* response */
  char response[BUFF_SIZE * 2];
  bzero(response, sizeof(response));
  generate_echo_response(buffer, response);
  int bytes_written = send(client, response, strlen(response), 0);
  if (bytes_written < 0) error_msg("Problem with send call", false);

  close(client);

  return NULL;
} /* detached thread terminates on return */

int main() {
  char buffer[BUFF_SIZE + 1];

  struct sockaddr_in client_addr;
  socklen_t len = sizeof(struct sockaddr_in);

  /* listening socket */
  int sock = create_server_socket(false);

  /* connections */
  while (true) {
    int client = accept(sock,
                        (struct sockaddr*) &client_addr,
                        &len);
    if (client < 0) error_msg("Problem accepting a
      ➥client request", true);

    announce_client(&client_addr.sin_addr);

    /* client handler */
    pthread_t tid;
    int flag = pthread_create(&tid,          /* id */
                              NULL,          /* attributes */
                              handle_client, /* routine */
                              &client);      /* routine's arg */
    if (flag < 0) error_msg("Problem creating thread", false);
  }

  return 0;
}
```

the player's movement of letters one at a time and on so. Suppose that there is a global variable:

```
typedef struct {
  /* variables to track game's state */
} WordGame;
WordGame game; /* single, global instance */
```

so that the application has, in the source code, exactly one WordGame instance visible across the application's functions (for example, move_letter, submit_guess). Concurrent players need separate WordGames, so that one player cannot interfere with another's game.

Now, consider two child processes C1 and C2, each handling a player. Under Linux, a forked child inherits its parent's address space; hence, C1 and C2 begin with the same address space as each other and their parent. If none of the three processes thereafter executes a write operation, no harm results. The need for separate address spaces first arises with write operations. Linux thus enforces a copy-on-write (COW) policy with respect to memory pages. If C1 or C2 performs a write operation on an inherited page, this child gets its own copy of the page, and the child's page table is updated. In effect, therefore, the parent and the forked children have separate address spaces, and each client effectively has its own copy of the writable WordGame.

A Multithreading Server

The multithreading_server shown in Listing 3 avoids the context-switch downside of the forking_server but faces challenges of its own. Each process has at least one thread of execution. A single multithreaded process has multiple threads. The threading_server is multithreaded.

The threading_server mimics the division-of-labor strategy in the forking_server, but the client handlers are now threads within a single process instead of forked child processes. This difference is huge. Thanks to COW, separate processes effectively have separate address spaces, but separate threads within a process share one address space.

When a client connects, the threading_server delegates the handling to a new thread:

```
pthread_create(&tid,           /* id */
               NULL,           /* attributes */
               handle_client,  /* routine */
               &client);       /* arg to routine */
```

The thread gets a unique identifier and executes a thread routine—in this case, handle_client. The threading_server passes the client socket to the thread routine, which reads from and writes to the client.

How could the WordGame be ported to the forking_server? This server must ensure one WordGame instance per client. The single WordGame:

```
WordGame game; /* one instance */
```

could become an array of these:

```
WordGame games[BACKLOG]; /* BACKLOG == max clients */
```

When a client connects, the threading_server could search for an available game instance and pass this to the client-handling thread:

```
int game_index = get_open_game(); /* called in main so thread safe */
```

In the function main, the threading_server would invoke get_open_game, and each client-handling thread then would have access to its own WordGame instance:

```
games[game_index].socket = client;
pthread_create(&tid,                 /* id */
               NULL,                 /* attributes */
               handle_client,        /* routine */
               &games[game_index]);  /* WordGame arg */
```

A WordGame local to the thread_routine also would work:

```
void* handle_client(void* client_ptr) {
  WordGame game; /* local so thread safe */
  /* ... */
}
```

Each thread gets its own copy of locals, which are thereby threadsafe. Of importance is that the programmer rather than the system ensures one

WordGame per client.

The threading_server would be more efficient with a thread pool. The pre-forking strategy used in FastCGI for processes extends nicely to threads.

A Polling Server

Listing 4 is a polling_server, which resembles the forking_server in some respects and the threading_server in others.

The polling_server is complicated:

Listing 4. polling_server.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <netinet/in.h>
#include <signal.h>
#include <sys/epoll.h>
#include <fcntl.h>
#include <errno.h>
#include "utils.h"

#define MAX_BUFFERS (BACKLOG + 1) /* max clients + listener */

int main() {
  char buffer[BUFF_SIZE + 1];

  /* epoll structures */
  struct epoll_event event,     /* server2epoll */
    event_buffers[MAX_BUFFERS]; /* epoll2server */

  int epollfd = epoll_create(MAX_BUFFERS); /* arg just a hint */
  if (epollfd &lt; 0) error_msg("Problem with epoll_create",
      ➥true);

  struct sockaddr_in client_addr;
  socklen_t len = sizeof(struct sockaddr_in);

  int sock = create_server_socket(true); /* non-blocking */

  /* polling */
  event.events = EPOLLIN | EPOLLET; /* incoming, edge-triggered */
  event.data.fd = sock;             /* register listener */
  if (epoll_ctl(epollfd, EPOLL_CTL_ADD, sock, &event) < 0)
    error_msg("Problem with epoll_ctl call", true);

  /* connections + requests */
  while (true) {
    /* event count */
    int n = epoll_wait(epollfd, event_buffers, MAX_BUFFERS, -1);
    if (n < 0) error_msg("Problem with epoll_wait call", true);

    /*
       -- If connection, add to polling: may be none or more
       -- If request, read and echo
    */
    int i;
    for (i = 0; i < n; i++) {
      /* listener? */
      if (event_buffers[i].data.fd == sock) {
        while (true) {
         socklen_t len = sizeof(client_addr);
         int client = accept(sock,
                             (struct sockaddr *) &client_addr,
                             &len);

         /* no client? */
         if (client < 0 && (EAGAIN == errno ||
         ➥EWOULDBLOCK == errno)) break;

         /* client */
         fcntl(client, F_SETFL, O_NONBLOCK); /* non-blocking */
         event.events = EPOLLIN | EPOLLET;   /* incoming,
                                                edge-triggered */
         event.data.fd = client;
         if (epoll_ctl(epollfd, EPOLL_CTL_ADD, client, &event) < 0)
          error_msg("Problem with epoll_ctl ADD call", false);

         announce_client(&client_addr.sin_addr);
        }
      }
      /* request */
      else {
           bzero(buffer, sizeof(buffer));
           int bytes_read = recv(event_buffers[i].data.fd, buffer,
        ➥sizeof(buffer), 0);

           /* echo request */
           if (bytes_read < 0) {
             char response[BUFF_SIZE * 2];
             bzero(response, sizeof(response));
             generate_echo_response(buffer, response);
             int bytes_written =
               send(event_buffers[i].data.fd, response,
           ➥strlen(response), 0);
             if (bytes_written < 0) error_msg("Problem with
             ➥send call", false);

             close(event_buffers[i].data.fd); /* epoll stops
                                                 polling fd */
           }
      }
    }
  }

  return 0;
}
```

```
while (true)          /* listening loop */
  for (...)           /* event loop */
     if (...)         /* accepting event? */
       while (true)   /* accepting loop  */
     else             /* request event */
```

This server executes as one thread in one process and so must support concurrency by jumping quickly from one task (for example, accepting connections) to another (for example, reading requests). These nimble jumps are among nonblocking I/O operations, in particular calls to accept (connections) and recv (requests).

The polling_server's call to accept returns immediately:

- If there are no clients waiting to connect, the server moves on to check whether there are requests to read.
- If there are waiting clients, the polling_server accepts them in a loop.

The polling_server uses the epoll system library, declaring a single epoll structure and an array of these:

```
struct epoll_event
    event,                       /* from server to epoll */
    event_buffers[MAX_EVENTS]; /* from epoll to server */
```

The server uses the single structure to register interest in connections on the listening socket and in requests on the client sockets. The epoll library uses the array of epoll structures to record such events. The division of labor is:

- The polling_server registers events of interest with epoll.
- The epoll library records detected events in epoll_event structures.
- The polling_server handles epoll-detected events.

The polling_server is interested in incoming (EPOLLIN) events and in

---

Low Cost Panel PC

PDX-057T

- Vortex86DX 1 GHz Fanless CPU
- 5.7" VGA LCD with Touchscreen
- 1 RS232/422/485 serial port
- Mini-PCI Expansion slot
- 2 USB 2.0 Host Ports
- 10/100 BaseT Ethernet & Audio
- PS/2 mouse & keyboard
- CompactFlash & MicroSD card sockets
- Resolution/Colors: 640 x 480 @ 256K
- Low Power Consumption
- Linux, Embedded XP or Windows CE OS Options
- Wide Input Voltage and Wireless Options

2.6 KERNEL

Setting up a Panel PC can be a *Puzzling* experience. However, the PDX-057T comes ready to run with Linux Operating System installed on flash disk. Just apply power and watch the Linux X-Windows desktop User Interface appear on the vivid color LCD . Interact with the PDX-057T using the responsive integrated touchscreen. 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... Starting at $440 Qty 1.

For more info visit: www.emacinc.com/panel_pc/pdx089.htm

Since 1985 OVER 24 YEARS OF SINGLE BOARD SOLUTIONS

EMAC, inc.
EQUIPMENT MONITOR AND CONTROL

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

Unlike the forking_server, the polling_server does not incur the cost of context switches among forked children.

edge-triggered (EPOLLET) rather than level-triggered events. The distinction comes from digital logic design but examples abound. A red traffic light is a level-triggered event signaling that a vehicle should remain stopped, whereas the transition from green to red is an edge-triggered event signaling that a vehicle should come to a stop. The polling_server is interested in connecting and requesting events when these first occur.

A for loop iterates through detected events. Above the loop, the statement:

```
int n = epoll_wait(epollfd, event_buffers, MAX_EVENTS, -1);
```

gets an event count, where the events are either connections or requests.

My polling_server takes a shortcut. When the server reads a request, it reads only the bytes then available. Yet the server might require several reads to get the full request; hence, the server should buffer the partials until the request is complete. I leave this fix as an exercise for the reader.

How could the WordGame be ported to the polling_server? This server, like the threading_server, must ensure one WordGame instance per client and must coordinate a client's access to its WordGame. On the upside, the polling_server is single-threaded and thereby threadsafe. Unlike the forking_server, the polling_server does not incur the cost of context switches among forked children.

Conclusions

Which is the best way to client concurrency? A reasoned answer must consider traditional multiprocessing and multithreading, together with hybrids of these. The "evented I/O" way that epoll exemplifies also deserves study. In the end, the selected method must meet the challenges of supporting concurrency across real-world Web applications under real-world conditions.■

Martin Kalin is a professor at the College of Computing and Digital Media at DePaul University, Chicago, Illinois. He earned his PhD at Northwestern University. Martin has co-authored various books on C and C++, authored a book on Java and, most recently, a book on Java Web services.

Resources

The three Web servers together with an iterative_server are available at **http://condor.depaul.edu/mkalin**.

For more on Node.js, see: **http://nodejs.org**.

Please visit us at http://northeastlinuxfest.org for more information

Join us for the 2nd annual Northeast GNU-Linuxfest!

MARCH 17, 2012

WORCESTER STATE UNIVERSITY

STUDENT CENTER

486 CHANDLER STREET

WORCESTER MA

http://northeastlinuxfest.org

http://northeastlinuxfest.org

Among the highlights -

Speakers:
Dru Lavigne
John Sullivan
James Turk

Events
Hackerspace competition
Open Street Map workshop
LPI & BSD exams

Please visit us at **http://northeastlinuxfest.org** for more information

HTML5 for Audio Applications

HTML5 lets you play music through compliant browsers—no “cloud” required.

PAUL FREITAS

Recently, “cloud”-based music services, from big names like Amazon, Google and Apple, have been getting attention in the press. These services allow you to store your music on a corporate server and access it through your own Internet-connected device anytime you like. It’s easy to see the appeal of these services. This is the kind of thing the Internet is for, right?

If you're reading this article, you're probably a Linux user, and as often happens, support for Linux is being neglected by these big corporate solutions. For example, Apple's service relies on its proprietary iTunes application, which doesn't exist in Linux. Other products have a Web interface, but uploading works only through a proprietary "app" not available for Linux users. Who wants to use proprietary software anyway? File-type support is limited as well with all the corporate products I've mentioned. Most of my own music is stored in Ogg Vorbis files, and none of the big company services seem to support it, lack of patents notwithstanding. There also are financial costs associated with the corporate offerings (explicit fees comparable to Internet-hosting costs, vendor lock-in and so on) that also are unattractive.

"Cloud" music services have other downsides as well. They are all intended for personal use. What if you want to share music with other people? (I am, of course, talking about legal music sharing involving files with Creative Commons-type free licensing or recorded cover songs with appropriate royalty payments being made to songwriters through licensing agencies like the Harry Fox Agency.) Cloud services can't help you. Also, transfer to the service is one-way. Aside from listening to your music, once you transfer music to the service, you can't download it again easily if something happens to your personal storage. What if you want to use your own storage solution, like your own personal (and appropriately secured) Internet-accessible Web server? What if you live outside the United States, where some cloud services are not yet available?

All these problems make "cloud" solutions more like fog, obscuring the truth that there is another way. Modern HTML5-compliant Web browsers like Chrome, Firefox (Iceweasel to Debian users) and Apple's Safari all support the HTML5 audio element, which can make audio files as easy to handle as image files. Adobe Flash is not necessary. Any Web server can be your own personal "cloud" server that you can use for personal or business use. With some customized HTML and JavaScript, the interface can be anything you want. Let's see how.

All these problems make "cloud" solutions more like fog, obscuring the truth that there is another way.

Playing Music through Your Browser

A typical browser supports multiple audio file types, regardless of the operating system running it. They all support different file types, however, and there are some interesting surprises, the most notable one being that Firefox/Iceweasel doesn't support MP3 files because of patent and licensing issues. That's okay, because it supports the patent-unencumbered Ogg Vorbis format used by many Linux users. So does Google's Chrome, and even Apple's Safari can play Ogg Vorbis with Xiph's QuickTime Components installed (see Resources).

Let's try it. Imagine you have an Ogg Vorbis file named test.ogg in the directory /home/me/music/. You can open it in a Linux-based browser like Chrome or Firefox with the URL file:///home/me/music/test.ogg. Your browser should load the file and start to play it. You'll see a player in the browser tab/window similar to the one shown in Figure 1.

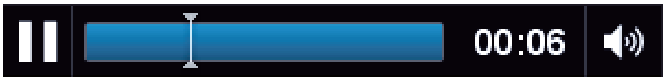


Figure 1. Google Chrome's default audio player control—other browsers have similar control sets, but different appearances.

Browsers render controls differently, but you'll see all the usual controls: play/pause button, position slider, current track position indicator and volume control. Note how much HTML you've had to write so far: none. If all you really want to do is play one file at a time, and you don't care what exactly shows up in the browser window, you can stop reading now.

Most Web users care about appearances, so the default audio controls alone probably won't cut it for most people. Fortunately, you can put an audio player explicitly in any HTML page. It's a lot like adding an image to a page. Instead of using an img element, you use an audio element, and the syntax of the two elements is similar. For example, to load and play test.ogg in an otherwise-empty HTML page using the browser's default controls, you could use the following page:

```
<html>
<body>
    <audio src="test.ogg" controls>Get an HTML5 browser!</audio>
</body>
</html>
```

Note that HTML5 simplifies the syntax of the root HTML element from what you might be used to. The message "Get an HTML5 browser!" will appear only when the page is loaded in a non-HTML5 browser. Other users will see a player element like the one shown in Figure 1, thanks to the controls attribute being set. By default, the audio element has no controls. (HTML5 allows you to set an attribute without a value. If you prefer something more XML-like, you can set an attribute to itself instead—for example, `controls="controls"`.)

It's easy to modify the audio tag for some simple use cases. If you want a sound clip to play in the background with no controls, you can omit the controls attribute:

```
<audio src="test.ogg" autoplay>Get an HTML5 browser!</audio>
```

You can add a loop attribute to make the audio file restart upon completion:

```
<audio src="test.ogg" autoplay loop>Get an HTML5 browser!</audio>
```

As I mentioned earlier, different browsers support different file formats, and they aren't all the same. Ogg Vorbis works for Chrome, Firefox and Safari (with Xiph's extension), but other browsers need something else, like MP3. At the time of this writing, it seems that all browsers support one or the other, but not necessarily both. What can you do to fix this problem?

HTML5 has another element called source that replaces the src attribute of the audio tag. It goes inside the audio element. Unlike the src attribute, you can have multiple source elements inside an audio tag. A browser will load the file referenced by the first source element it finds that it can actually work with. Let's look at the first example again, this time with source attributes:

```
<audio controls>
        <source src="test.ogg"/>
        <source src="test.mp3"/>
        Get an HTML5 browser!
</audio>
```

A browser will attempt to read and play test.ogg first. If it fails for whatever reason (it can't find the file, it can't play that file type and so on), it simply will move on to test.mp3 and use that instead. Use as many source elements as you like, although in practice, two is usually enough.

Adding a Custom User Interface

These simple examples have their uses, of course. Still, most Web developers rather would see a more-consistent user interface than these examples can provide. Having three different UIs appear on three different browsers can affect the usability of a page dramatically. For a professional site, you'll need a professional-looking page. Fortunately, the audio element has a JavaScript API you can use to control the player in real time. To get the interface you want, write it using standard HTML techniques, and set event handlers to access the audio player via JavaScript's API.

Here's a simple example. The following page displays a single play/pause button instead of the native controls:

```
<html>
<body>
    <audio src="test.ogg" id="player" loop>Get an HTML5 browser!</audio>
    <form id="interface">
        <input type="button" value="Play"
         ➥onclick="PlayPause()" id="playpause"/>
    </form>
    <script type="text/javascript">
    var audioPlayer = document.getElementById("player");

    function PlayPause()
    {
        if (audioPlayer.paused)
        {
            audioPlayer.play();
            document.getElementById("playpause").value = "Pause";
        }
        else
        {
            audioPlayer.pause();
            document.getElementById("playpause").value = "Play";
        }
    }
    </script>
</body>
</html>
```

After the page loads, press the Play button to start playing test.ogg in a loop. When a user presses the button, the browser calls the audio player object's `play()` function to start playing the track. The button label then changes to Pause. You then can press the same button to pause the audio, which calls the audio player object's `pause()` function to pause playback.

There are many attributes and methods associated with an audio player. You can change the current play time of the player by changing the `currentTime` property. Change the volume by changing the `volume` attribute. There are many others, not all of them implemented in all browsers at the time of this writing. For more information, see the W3C HTML5 specification listed in the Resources section of this article, as well as the documentation for your preferred browser.

A More-Detailed Example: Playlist HTML5 Audio Player

By now you've seen the basics of developing HTML5 for audio applications. You might want to see a more-detailed example. Unfortunately, magazine articles are limited in size, and Web development code takes up a lot of space. Instead, I'd like to refer you to an open-source project I've written called the Playlist HTML5 Audio Player (see Resources). Here is some background on the project.

The goal of Playlist is to provide a user interface for playing a collection of audio files in series, like an album of recorded music. To use the interface, all you need are a collection of audio files and a text file, called a playlist, that lists the files individually. If cover art is available, Playlist will show that as well. Playlist also can load extra information about the collection for display in its About tab. Neither of the last two items are required. Figure 2 shows Playlist playing some freely redistributable sample music.

Playlist supports controls commonly found on a car's CD player, including a shuffle button. It will loop through the

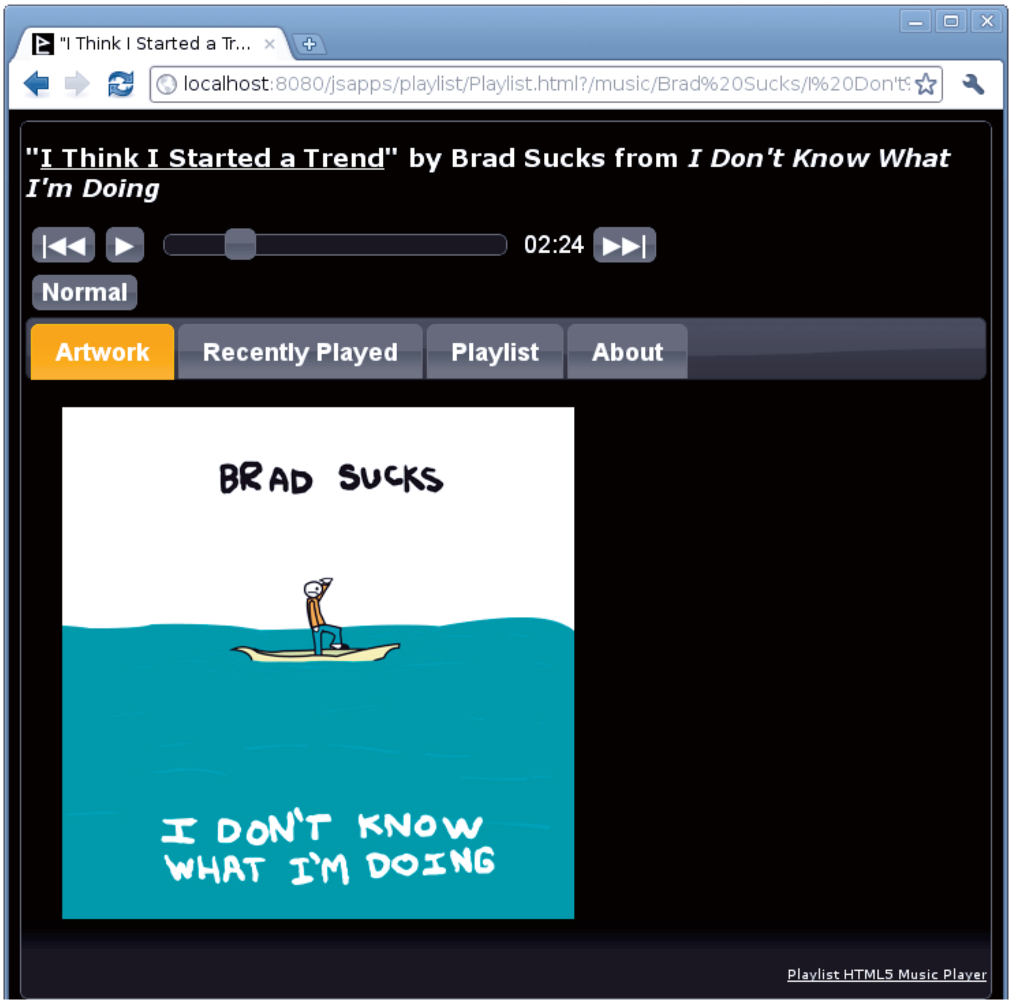

"I Think I Started a Tr...
localhost:8080/jsapps/playlist/Playlist.html?/music/Brad%20Sucks/I%20Don't
"I Think I Started a Trend" by Brad Sucks from I Don't Know What I'm Doing
02:24
Normal
Artwork
Recently Played
Playlist
About
BRAD SUCKS
I DON'T KNOW WHAT I'M DOING
Playlist HTML5 Music Player

Figure 2. Playlist HTML5 Audio Player Playing Some Freely Redistributable Sample Music

chosen playlist indefinitely until you pause it. It has jQuery-based controls and tabs, which allow you to change the appearance easily using jQuery's Themeroller tool. Thanks to jQuery's table-sorting and re-ordering abilities, you can re-organize the track order in the Playlist tab any way you like for the duration of the session. For example, Wikipedia says that the Beatles' song "Her Majesty" originally appeared between "Mean Mr. Mustard" and "Polythene Pam" on the *Abbey Road* album. With Playlist, you can drag it from the end of the album and drop it between the other two tracks to hear what that actually sounds like. Playlist is client-side only, so the next time you load your *Abbey Road* page, "Her Majesty" will be back at the end of the album again.

You can use Playlist for your own personal music collection on your own computer through its own Web server. If you want the full "cloud" experience, you also can use it on your own Web site on any of the ubiquitous Web-hosting services that are available today. You don't need any special server-side tools like PHP or MySQL, because Playlist contains no server-side code. You can access your music locally or remotely, through any HTML5-compliant browser that supports your preferred audio file type. I use Playlist for my own music collection, and for my purposes, it works as well as or better than Linux players like Rhythmbox, Exaile and Amarok.

Playlist also can be useful for more-commercial applications. Bands, musicians and songwriters often want to distribute recordings of their own work as part of their overall promotional strategy. If the recording is of original music, something like Playlist can be used directly. If any of the songs are re-recordings of other songwriters' work, appropriate licenses must be obtained, such as the mechanical licenses available for US distribution from the Harry Fox Agency. Accounting for such licenses must happen at least in part on the server, but you still can use Playlist for the UI.

Installing Playlist is reasonably straightforward. In its most simple form, Playlist HTML5 Audio Player consists of a series of ordinary Web files, including Playlist.html (an HTML5 file), Playlist.js (a JavaScript file) and a series of accessory files, including images and jQuery theming elements. All these files are contained in a directory named jsapps (think "JavaScript Applications"). Drop this folder on a Web server where it can be accessed easily. The content to be played (which is, of course, separated from the player) goes somewhere else that makes logical sense to your site, like a folder named music at the same level as jsapps.

Once your audio files are in place, you will need a text file in the same directory named Playlist.m3u that lists the files to play. For example, if you have two tracks named Track1.ogg and Track2.ogg, your file would look like this:

```
Track1.ogg
Track2.ogg
```

You can use command-line tools like `ls` or GUI tools like Audio Tag Tool to create Playlist.m3u more easily.

Finally, you need a file that redirects the browser to Playlist.html in the jsapps directory, along with query parameters that reference the Playlist file. For your convenience, there's one named redirect.html in the jsapps/playlist directory that will do the job via JavaScript. Copy or link that file into your music's directory, alongside Playlist.m3u. I often name the redirect file index.html. Add cover art (Cover.jpg) and HTML-formatted additional information (About.xml) if you like. You're done. Browse to the redirect file and see what happens.

I have Playlist set up on my own Web site so you can easily try it (see Resources). To try Playlist on your own machine, download it from the project home on SourceForge (see Resources). I've packaged it inside a version of Jetty, an open-source Java-based Web server that will run on any platform with a Java Runtime Environment. I've added Jetty as a convenience, but you don't need to use it. You either can download the source from the project page with Git or get the full download with Jetty and copy the music and jsapps directories out of Jetty's webapps directory into an appropriate location on your own Web server. If you want to use Jetty, run `start.sh` (or `start.bat` in Windows) to get the server going, then browse to it via port 8080 (for example, http://localhost:8080/music). All of these test cases will cause freely redistributable sample music to play that will give you an idea of Playlist's capabilities.

Conclusion (and Warning)

HTML5 is a powerful tool for listening to audio via the Internet, as powerful as anything the "cloud" services have to offer and much more versatile. Coding audio into HTML5 pages is fairly straightforward. If writing HTML doesn't interest you and you're looking for a packaged solution, try my open-sourced Playlist HTML5 Audio Player.

I'll end with a warning. With HTML5's great power comes great responsibility—to yourself first and foremost. We've all heard about lawsuits and arrests related to copyright violations and file sharing. You don't want to be part of that.

To protect yourself, use HTML5 audio responsibly. Don't put nonfree music in a publicly accessible or search-engine-crawlable location on any Web server. Firewalls, SSL certificates, Web server configuration files (including Apache's htaccess and norobots.txt), user authentication and other common-sense server-side strategies can help you enjoy your audio anywhere while keeping your private music collection private. If you're distributing music legally, make sure you have all appropriate rights and licenses, mechanical and otherwise. ■

Paul Freitas (paulfreitas.com) is an independent technology consultant based in Boise, Idaho.

Resources

HTML5 in General: **http://www.w3.org/TR/html5**

HTML5's Audio Element in Detail: **http://www.w3.org/TR/html5/the-iframe-element.html#the-audio-element**

Xiph.org's QuickTime Components: **http://www.xiph.org/quicktime**

Playlist HTML5 Audio Player: **https://sourceforge.net/projects/playlistplayer**

JQuery: **http://jquery.com**

An example of Playlist, served from my own Web site and playing music with a free license: **http://paulfreitas.com/jsapps/playlist/Playlist.html?/music/Brad%20Sucks/I%20Don't%20Know%20What%20I'm%20Doing/Playlist.m3u**

Harry Fox Agency's Web Site (for obtaining "cover music" mechanical licenses): **http://www.harryfox.com**

Basic Web Design with Drupal 7

Using Drupal to manage and display data.

KENNETH P. J. DYER

Drupal is one of the most popular and versatile platforms for Web design. It's free, open source and will run on Linux. Early last year, a new version was released (Drupal 7), making it even better with improvements in usability, performance and security. If you've looked at Drupal before, but didn't end up using it, you may want to take another look.

Drupal is built using a modular structure to accommodate extensions from third-party developers. These modules allow you to add features that otherwise might be unavailable. The improvements to Drupal 7 make them far more accessible and easier for new users.

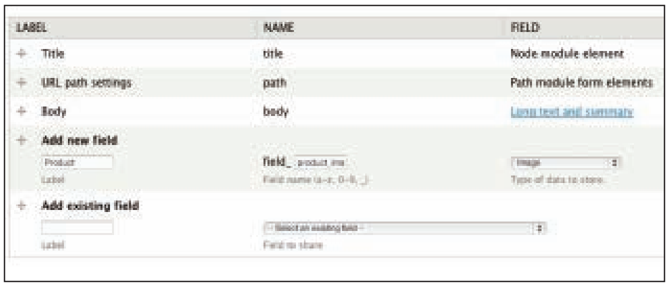
This article covers the basics of creating a data scheme (that is, content types), how to manipulate that data (that is, views) and fundamental ways to display data in Drupal 7. For the examples in this article, I use products of a business site that sells writing instruments. I assume you have already downloaded the Drupal 7 from drupal.org and have installed it successfully.

Content Types

A content type is simply a definition of a table of data, such as text, numbers and images. Behind the scenes, Drupal uses MySQL for storing data. When you create a content type in Drupal, in essence, you're creating a table in MySQL and setting the columns in MySQL. For this example writing instruments site, let's create a content type for the products. It will include the name and description, as well as a field for storing a photograph of each product.

To create a content type, you need to do a few things. After logging in to Drupal as an administrator, go to the Administration page. From there, click the link for Structure. On the Structure page, click on Content Types. On the Content Types page, click the Add content type link. Incidentally, in the Drupal community, navigation to this page would be described typically in a shorter method: Click on Administration→Structure→Content Types→Add content type. This menu-tree method generally is accompanied with an absolute path for the domain shown within parentheses (for example, /admin/structure/types/add). From this point on, I use the shortened method.

After you've opened the admin page for creating a content type (Figure 1), in the Name field, enter "Product". Leave the rest, and click the button that reads "Save and Add Fields". On the next page, you'll see a table with a few fields

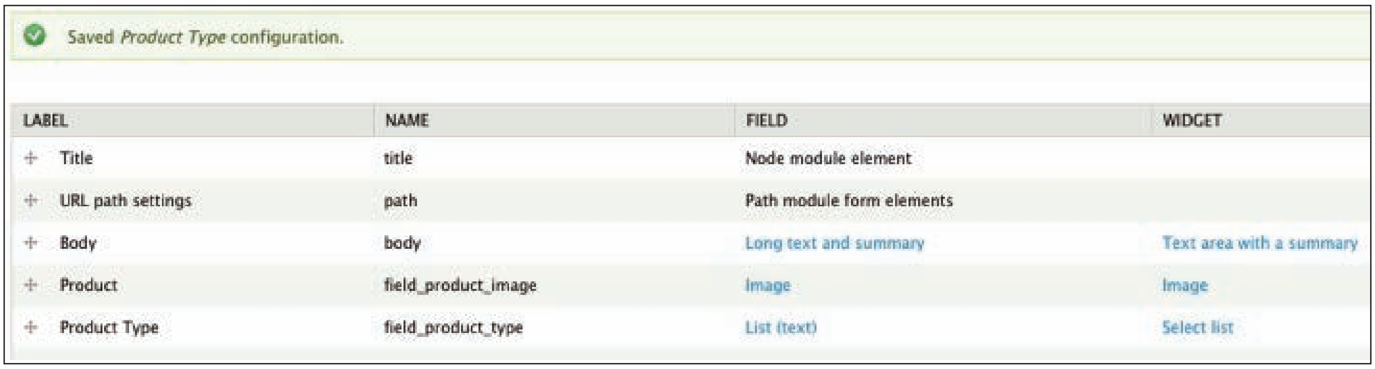


LABEL	NAME	FIELD
Title	title	Node module element
URL path settings	path	Path module form elements
Body	body	Long text and summary
Add new field		
Product Label	field_ product_ima Field name (a–z, 0–9, _)	Image Type of data to store.
Add existing field		
 Label	- Select an existing field - Field to share	

Figure 1. Content Type Add Fields

(for example, Title, Body). Let's add a couple more for the site's specific needs. In the Add new field section, type in the first box, "Product Image". For the Name column, enter "product_image" (this creates a field in MySQL, field_product_image). For the Field Type, select "Image" from the drop-down list. Now click Save. On the next two screens, accept the defaults and click the save buttons for each page.

Completion of the above will take you back to the table of fields for Product. Now, let's add another field to indicate whether an item is a pen or a pencil. Call it "Product Type", and set the Field Type to "List (text)", leaving the Widget set to Select List, so that when content is entered later, you can select the product type from a list. This time after you save, you'll see a form containing a field called, "Allowed values list". Enter in that field, on separate lines, "1|Pen" and "2|Pencil" (without the quotation marks). This is a key and value hash that will become the available choices for users. Now save your way out, accepting default settings. The field list should then look like the screenshot shown in Figure 2.

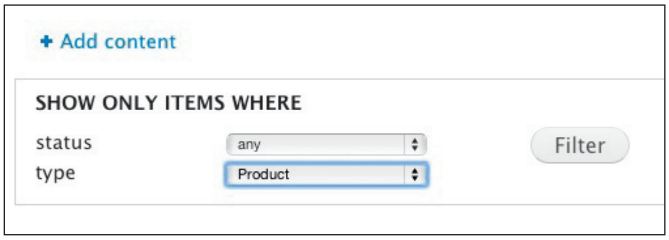


Saved *Product Type* configuration.

LABEL	NAME	FIELD	WIDGET
Title	title	Node module element	
URL path settings	path	Path module form elements	
Body	body	Long text and summary	Text area with a summary
Product	field_product_image	Image	Image
Product Type	field_product_type	List (text)	Select list

Figure 2. Product Type Manage Fields

You may want to experiment and add a few more fields (such as Price, Ink Color and so on). Otherwise, you now can create products on the site. To make things easier to understand, add a few pens and pencils. Search the Web for photos of pens and pencils, and grab them so you'll have images to go with the products you add. To enter products, go to Administration→Content→Add content and click on the content type, "Product" (/node/add/product). If you make a mistake and want to change a product after it's saved, or if you just want to see a list of products, go to Administration→Content (/admin/content). There, select the type, "Product", and click Filter to show only products (Figure 3).



SHOW ONLY ITEMS WHERE

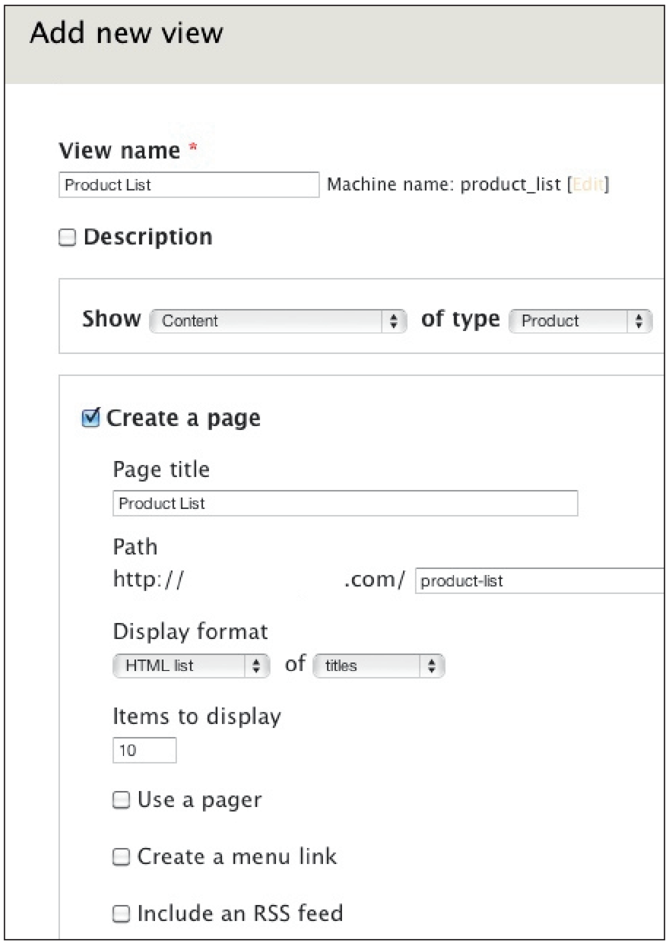
status: any

type: Product

Figure 3. Content List

Building Views

Each content entry that you create, you can access individually, as an on-the-fly page, which are referred to as nodes. However, let's also create a page that lists



Add new view

View name *: Product List Machine name: product_list [Edit]

☐ Description

Show: Content of type: Product

☑ Create a page

Page title: Product List

Path: http://   .com/ product-list

Display format: HTML list of titles

Items to display: 10

☐ Use a pager

☐ Create a menu link

☐ Include an RSS feed

Figure 4. Add New View

all of the products with links to these nodes. To do this, you need to create a view. A view allows you to aggregate and arrange content on a site, and to organize and group data based on field values and other factors.

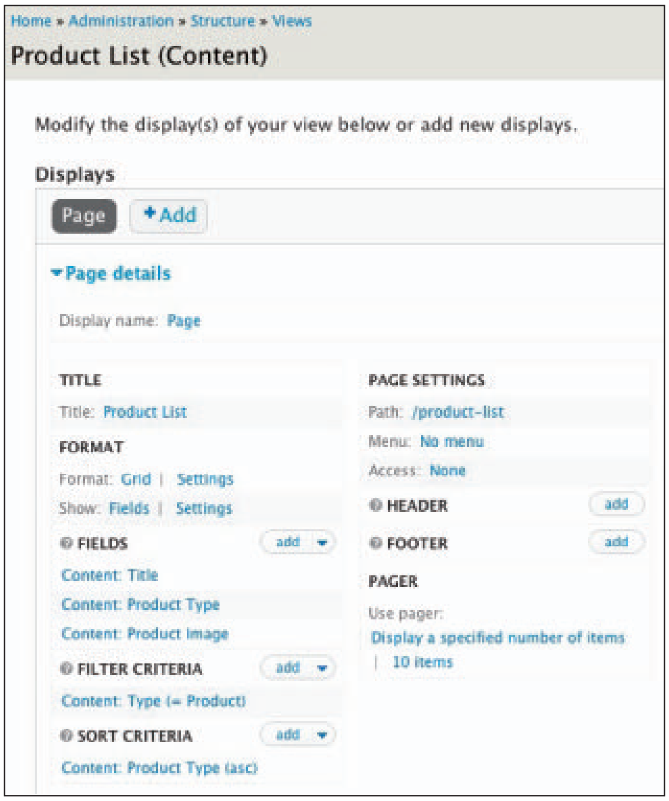
To create a view showing a list of pens and pencils, go to Administration→Structure→Views, and then click on "Add new view" (/admin/structure/views/add). This displays a form (Figure 4) for creating a page—one that draws data from MySQL and generates a Web page when called.

In the View Name field, enter "Product List". In the section that begins with Show, you can select the kind of view to create. For our purposes, select "Content". To the right of this, after the options update, you may select the content type to show. Choose "Product" from the list. If you hadn't noticed, this form is something of a wizard. Now change the sorting method to "Unsorted".

Next you have two check boxes: one for creating a page and another for creating a block. Basically, pages in this sense are views and may be used to generate standalone pages. Blocks are views that can display the same information, but that may be used as components within other pages. Pages are better at this point, so check "Create a page".

In the section that opens for creating a page, type in a title for the top of the page. Next enter the file path. The wizard provides the domain name and leading slash. Just add the page name in the box as "product-list". This means that when people using a Web browser go to http://domain_name.com/product-list, they will see the results of the display. Next, you need to set the display format. For what's to come, choose Grid of Fields. Ignore the other settings and click the button, "Continue & Edit" to tweak the view further.

In the Fields section, by default, the view creates a Title field. Let's add at least two more fields. Click on the link "add" across from the FIELDS label to add a field. This lists many fields that may be added. Scan the list for "Content: Product Type"—that's the


Home » Administration » Structure » Views

Product List (Content)

Modify the display(s) of your view below or add new displays.

Displays

Page +Add

Page details

Display name: Page

TITLE
Title: Product List
FORMAT
Format: Grid | Settings
Show: Fields | Settings
FIELDS add
Content: Title
Content: Product Type
Content: Product Image
FILTER CRITERIA add
Content: Type (= Product)
SORT CRITERIA add
Content: Product Type (asc)

PAGE SETTINGS
Path: /product-list
Menu: No menu
Access: None
HEADER add
FOOTER add
PAGER
Use pager:
Display a specified number of items | 10 items

Figure 5. Product List View

one for choosing whether a product is a pen or pencil. When you find it, click the check box next to it. Probably just above it, you will find "Content: Product Image". Check that as well. Then click the button at the bottom, labeled "Add and configure fields".

The fields configuration page opens for each of the fields you're adding, one at a time. For the Product Type field, check the box that says, "Exclude from display", since it won't be necessary to display the word Pen for each pen. Instead, let's use this field for grouping the data. Now, click the Apply button. When you see the form for Product Image, for style preferences, ensure that "Create a label" is unchecked. For the Image Style, select "thumbnail" from the list and then click Apply. Your page should look like the screenshot shown in Figure 5.

Now that you've added the Product Type field, you can use it for grouping the products. To do this, in the Format section, next to the Grid link, click on the link for its Settings. Near the top of the box that opens, where it says "Grouping field", choose "Content: Product Type" from the list. Enter 4 in the field for the number of columns and then click Apply. The name of each item and its photo now appear in the display, but each product is grouped based on whether it's a pen or a pencil. A larger, bold-faced heading appears for each group. Click on the button near the bottom

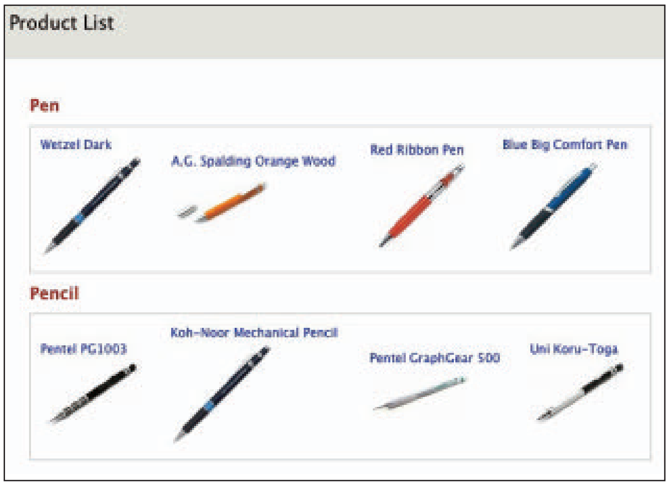


Figure 6. Product List Page

labeled "Update Preview", so that you can see the results of the query in MySQL generated by the view.

By default, the view sorts the groups of products in descending alphanumerical order, based on their Product Type headings. So, the pencils display before the pens. Because there's more money in pens than mechanical pencils, you might want to put pens first. You can reverse the order by adding a Sort Criteria for "Content: Product Type" and setting it to "Sort ascending".

There's plenty more that you can do, but this is enough to get started. Click the Save button at the top right. Then, check the results of your efforts by opening the page that users will see (/product-lists).

Notice in Figure 6 that the pens are grouped together, separated from the mechanical pencils. Because I entered content entries for four pens and four pencils for testing and set the number of columns to four in the view, the results are showing only four of each in one row. Had I entered eight pens, two rows of pens would show. The name of each pen and pencil functions as a link to the node, which presents a larger photo of the product, as well as all of the other information on the pen and any other fields I may have added. You can adjust the view to provide more information at this level. You also can have the links set to go to a different page generated by the view. And, you can change the formatting (that is, CSS using the CSS Injector module) to alter the colors and layout, as well as choose a different theme for Drupal. However, what I've done here demonstrates the basics of creating a content type and a view to filter and organize the data for a particular content, as well as how to display that data dynamically.

Other Considerations: Taxonomy

Having explained the absolute basics of using Drupal to manage and display data, I would be remiss in speaking on content types and views if I did not also mention the advantages of the Taxonomy module. Taxonomy is a property in which a vocabulary of related words are assembled. In the example site, I'm using the Product Type field to differentiate between different kinds of products available. This arrangement works fine if all you have are two kinds of products and rarely change them, but it proves cumbersome if the list of product types is long or were to change occasionally.

For instance, instead of just pen, you may want to distinguish between ball-point pens, felt-tip pens and fountain pens. You might want to have wooden pencils and mechanical pencils instead of just a product type of pencil. You may want to rearrange the ordering of the products so that red felt-tip pens appear above blue ball-points, but do not overshadow orange fountain pens. The more complex the list of product types, the less feasible it becomes to handle them within a field in the content type, and the more reasonable it becomes to adopt taxonomy instead.

The Taxonomy module allows you to create lists of custom values for use in organizing your products. Unlike, the present Product Type field, these values, called terms, provide more configuration options, allowing you greater precision in adding and arranging new product types.

Terms do not exist on their own in Drupal. Instead, you create them in lists called vocabularies. A vocabulary allows you to keep all related terms together so that you can add them as possible values for content types. Compared to the Product Type field in the example site, "Pen" and "Pencil" in the "Allowed values list" are terms, and collectively these terms are a vocabulary.

To create a vocabulary, go to the Administration→Structure→Taxonomy page, and click the "Add vocabulary" link (/admin/structure/taxonomy/add). In the Name field, enter "Product Types", then save. This returns you to the Taxonomy page, which has added an entry for Product Types. To add terms to your vocabulary, click the add terms link for Product Types. In the Name field, enter "Fountain Pen". Ignore the rest of the settings for now and click Save. The add term form re-opens. Continue to add terms for different types of writing instruments.

In order to use these terms with the products, you must replace the existing Product Type field with a new one that is linked to the new vocabulary. Go back to the Content types page at Administration→Structure→Content Types. On the listing for Products, click its "Manage Fields" link (/admin/structure/types/manage/product/fields). Click the Delete link for Product Type field. Then re-create the Product Type field, this time selecting "Term reference" for the Field column. On the field configuration page, at the bottom, from the drop-down list for Vocabulary select "Product Types", and then save.

The Products content types will update with the new field. Although it appears the same as the old field, the options available now are drawn from the Taxonomy, providing easier control over their arrangement and number.

Now that you have deleted the old Product Type field, the products list view no longer groups results correctly by types of products. To restore this feature, edit the Product List view to include the new field. Go to Administration→Structure→Views (/admin/structure/views), and click the Edit link for Product List. Then add the "Content: Product Type" field using the previous configurations. Repeat the steps given above to format your view, grouping products around the Product Type field. Save when finished.

Conclusion

You now should have a good sense of how to create content types and how to manipulate them with a view to display data in simple ways with Drupal 7. Now you can start creating your own Web sites and learn more as you go. Compared to other methods of building Web sites, Drupal may seem restrictive and awkward at first. However, in time, you will come to see that it's powerful and works well in an organization with many people of various technical backgrounds. Once you've developed a site, nontechnical users can add and maintain content easily. Plus, as a developer, with modules, you can add functionality easily with additional modules. If you would like to learn more about Drupal 7, see **http://drupal.org**. ■

Kenneth P. J. Dyer is a writer living in the Boston area, who has worked on Acquia Drupal's documentation. He's from New Orleans, where he studied English and Classical Studies at Loyola University.

ConFoo.CA
WEB TECHNO CONFERENCE

PRESENTED BY: MANWIN CANADA

FEB 29 - MAR 2 IN MONTREAL

Master the art of Web

Conference for developers
160 presentations
Experts from around the world

PHP
Python
Java
Ruby
HTML5
Javascript
Mobile

Visit confoo.ca for more information about the conference and the training.

Follow us on Twitter: @confooca

SPONSORED BY:

easyDNS www.easyDNS.com

thePHP.cc

MAKE WEB NOT WAR

nurun

Engine Yard

MEDIA SPONSOR:

LINUX JOURNAL

Interview with Dr Stephen Wolfram

***LJ* chats with the developer of Mathematica and the Wolfram|Alpha computational knowledge engine.**

JAMES GRAY

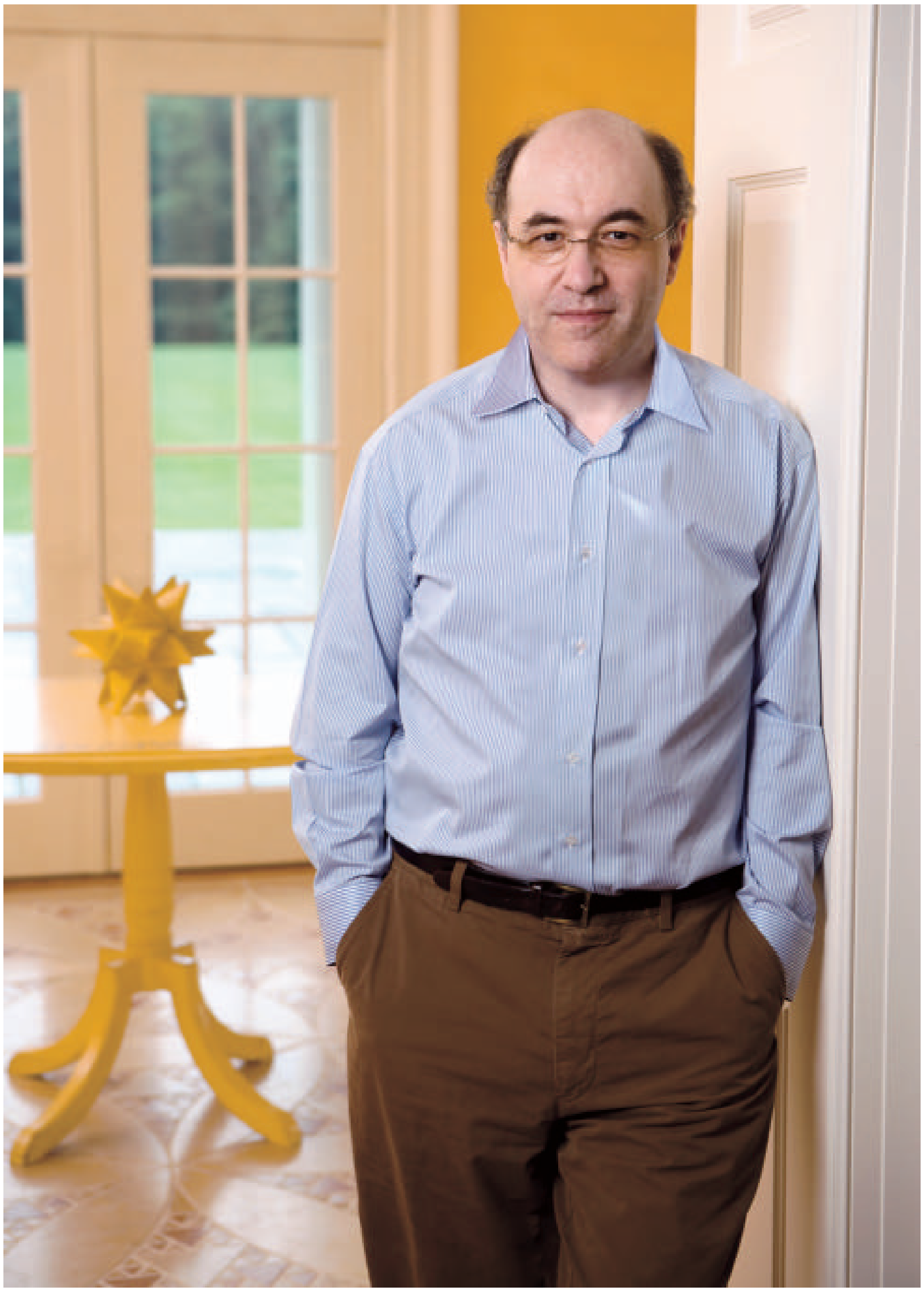
There are myriad ways to describe the prolific Dr Stephen Wolfram—inventor, author, groundbreaking scientist, IT CEO and entrepreneur and MacArthur Genius Grant recipient (also toss in Renaissance man to emphasize the diversity of his achievements). Despite Wolfram's many laurels, there's a fair chance you never have heard of him. This accomplished Briton, who has lived most of his adult life here on American soil, developed Mathematica, the world-standard technical computing package, and is refining the ambitious Wolfram|Alpha computational knowledge engine, a long-term project to make the world's knowledge computable and accessible to everyone.

Before Mathematica and Wolfram|Alpha, Wolfram published his first scientific paper at age 15 and knocked out his PhD in theoretical physics from Caltech by age 20. He's a man who seems to be living well ahead of his age as well as his time.

Recently, *Linux Journal* caught up with Dr Stephen Wolfram to learn more about this fascinating man and the important, cutting-edge work he continues to do.

JG: Your "computational knowledge engine" Wolfram|Alpha has been running for more than two years now. How has it been doing out in "the real world"? And, are there any surprises regarding who is using it and how?

SW: It was a difficult decision when to first release Wolfram|Alpha. Eventually, we just couldn't learn much more without seeing actual users using it. Watching that has been fascinating, and it has let us steadily make Wolfram|Alpha better. Our goal is to understand queries in whatever form a user chooses to give them—with no "manual". We have a whole calculus of linguistic variation, but it's amazing all the strange forms people actually use. These days though, we can understand more than 95% of queries the first time, which I think is pretty good.



Dr Stephen Wolfram

As far as users and uses are concerned, I'm continually amazed at the diversity of people I run into who turn out to be avid Wolfram|Alpha users. We intended Wolfram|Alpha to become a very widespread tool, and that's steadily happening.

JG: A fellow thinker, Rudy Rucker, called Wolfram|Alpha a "platonic search engine"—one that unearths eternal truths that may never have been written down before. Is that an accurate observation?

SW: It's a nice description. At a practical level, I can say that these days the majority of queries people enter in Wolfram|Alpha get zero hits in a search engine. But we can handle them because we're computing answers afresh, not trying to find something someone happened to write down before.

Sometimes the truths we compute are "eternal", like the solution of a differential equation or the property of a polyhedron. But a lot of the

I’ve done some pretty complex projects in my life, but I have to say that Wolfram|Alpha is outrageously more complex than any of the others.

time, they’re dynamic. They are based on data that comes from the outside world, like the weather somewhere, or the price of a stock or the position of a spacecraft. And at some level, what starts as a philosophical issue about eternal truths ends up as a very practical issue for the Wolfram|Alpha caching system.

JG: What is your long-term vision for Wolfram|Alpha?

SW: I’d like to see as much of the world’s knowledge as possible be computable, so that it can be used to answer whatever specific query one might have. With the progress of technology, we’ve been able to automate so many things. But a lot of the achievements of our civilization are still encapsulated only in the knowledge of human experts. I’d like to automate using that knowledge and get to the point that any question that could be answered by a human expert based on accumulated knowledge and data can be answered automatically by Wolfram|Alpha. I think that kind of automation is going to let us humans get a lot further on many fronts.

At an engineering level, Wolfram|Alpha gives us a whole new approach that I call “knowledge-based computing”. We’re used to the idea that software has to be built from raw computational primitives. But Wolfram|Alpha gives us a platform where we can in a sense start from the knowledge of the world, and then build from that. The result is a major change in the economics of all sorts of software development, as well as making lots of new and sometimes unexpected things possible. We’re seeing some of these things as Wolfram|Alpha extends from just being a Web site to supporting a whole ecosystem.

As of right now, Wolfram|Alpha takes short textual queries and computes results from them. We’re about to release a version that can take all sorts of other input—whether it’s images or data files or programs. And we’re going to see it not just compute results, but also cause things to happen. For example, last year we released a new version of Mathematica that uses Wolfram|Alpha to synthesize runnable programs from free-form linguistic input.

JG: What have been your technical challenges regarding Wolfram|Alpha?

SW: I’ve done some pretty complex projects in my life, but I have to say that Wolfram|Alpha is outrageously more complex than any of the others. It’s got an incredible number of different kinds of moving parts.

We’ve got to get computable data

about every kind of thing. We've built a pipeline for curating data that's a mixture of automation and expert human input. And even though we've handled thousands of domains now, it's amazing that almost every new one seems to have some new twist. And, to get it all right often requires talking to the world expert in that domain.

After the data, we've got to actually implement all the methods and models and algorithms that are needed to compute useful things from that data. It's all done in Mathematica (which is what makes it possible), but by now Wolfram|Alpha is about 15 million lines of Mathematica code.

Then there's the linguistic understanding, which at first I thought—based on the history of natural language processing (NLP)—might just be plain impossible. But the problem we're solving is sort of the inverse of the usual NLP problem, and particularly by using insights from an area of basic science that I've developed, we've been able to make incredible progress.

There are issues with generating output too—automating aesthetic judgments and the architecture of information presentation. And then, of course, there's lots of supercomputer-style infrastructure that's needed to actually run the computations for Wolfram|Alpha.

From a software-engineering point of view, we've made use of all the tools that we've developed during the past 25 years for Mathematica. These days, the tools are almost all written in the Mathematica language, and they do a really good job of automating building, testing and deployment. Wolfram|Alpha is an incredibly nontrivial system to monitor and test, but so far we've been pushing out new versions successfully every week ever since it launched.

JG: I imagine your development team for Wolfram|Alpha must be quite eclectic. Can you tell us a little bit about that team?

SW: Yes, it's a very eclectic team. On the data and algorithms side, we have experts in lots and lots of different fields. For the overall frameworks, I think the highest concentrations of backgrounds are physics and computer science. There's also a huge concentration—particularly in the linguistic algorithms area—of people who've worked on science based on my book *A New Kind of Science*.

Wolfram|Alpha almost by definition requires incredible diversity of skills and knowledge. We've ended up having to keep a WhoKnowsWhat database at our company so we can quickly find who might happen to know about forestry, or Vietnamese character sets, or machine tools or whatever.

It's also interesting that Wolfram|Alpha needs people with skills that I doubt have ever been defined before—for example, linguistic curation.

JG: As you look back on 25 years and eight versions of your original and flagship application, Mathematica, how would you assess its impact and importance in the world of technical computing?

SW: Before Mathematica, it was really rare for technical people to "do their own computing". They usually had to have special "programmers" who did it. Mathematica has made that unnecessary, and the result is that an awful lot of advances have been made. It's pretty satisfying to see how much has been discovered and invented through the years with Mathematica. Fields like experimental mathematics have pretty much been made possible by it.

I'm not quite sure, but I think I probably coined the term "technical computing"—not that I like it very much. I have to say that I think we pretty much defined what "technical computing" should be. Through the years, we've redefined it a few times, and I think that particularly as we bring Wolfram|Alpha and Mathematica together, we're going to see the most dramatic changes that have happened since we first released Mathematica.

Even after 25 years though, the story of Mathematica is only just beginning. Through the years, I've slowly understood more and more that's possible on the basis of the fundamental ideas in Mathematica, like symbolic programming. We're now pretty much finished with the to-do lists that I had for Mathematica in the late 1980s, but now I realize there's so much more that's possible.

Through the years, Mathematica has emerged as an incredibly strong platform for algorithm-rich software development. We certainly use it very successfully for pretty much everything at our company, and of course, it's the basis for Wolfram|Alpha. It's used in plenty of other organizations too, but we're building some things now that I suspect will make it a lot more widespread in the future.

JG: Can you share some interesting insights about Mathematica that would interest our technical readers?

SW: At a technical level, the core of Mathematica is its symbolic programming language. And, that language is based on one big idea: that everything—whether it's data, documents, programs, interfaces, whatever—can be represented in a very uniform way, as a symbolic expression. And because of that unification, it's possible to have a fairly small number of very powerful primitives that give a vast amount of functionality.

In terms of scope, we've had a pretty simple principle: just implement every definite algorithm we can. At the beginning, we emphasized things that would often be classified as "math", but we long ago expanded far, far, beyond that. Whether it's image processing, or control theory, or network analysis or text processing, if there are useful algorithms, they're probably in Mathematica.

A strong principle in developing Mathematica has been to automate as much as possible. The humans say what they want to achieve; it's up to Mathematica to figure out which algorithm to use, or whatever, to achieve that. We've also worked very hard to keep the design of Mathematica as coherent and consistent as possible. And, the result of all this is that in a sense we get to build Mathematica with bigger and bigger bricks. Each new piece of functionality finds it easy to use larger and larger chunks of existing functionality.

We've done a lot of work on both system design and software engineering during the years, and it's increasingly been paying off. So if you look, for example, at a plot of the number of functions supported by Mathematica versus time, it has a pretty impressive, perhaps exponentially looking form in recent years.

JG: To what extent did Mathematica inspire and inform Wolfram|Alpha?

SW: Well, of course, at a practical level, Wolfram|Alpha is implemented in the Mathematica language. It's all Mathematica. And if we hadn't had Mathematica, Wolfram|Alpha never would have been possible.

We use Mathematica for all its built-in algorithms, but more important, we use it as an incredibly powerful language for representing knowledge and defining new algorithms. We also use it as a large-scale deployment environment—actually delivering the computations and results from Wolfram|Alpha on the Web and elsewhere. And, needless to say, all our software engineering and testing and monitoring systems are written in Mathematica too.

In a sense, Mathematica and Wolfram|Alpha are conceptual opposites. Mathematica is based on a very clean and precise language that can be used to build very large structures. Wolfram|Alpha has to handle all the messiness of the world, and of human language, and initially is used mainly for one-shot queries.

For me, it was very interesting to think about Wolfram|Alpha after so many years of working on Mathematica. I could, in a sense, do everything exactly the opposite way. But, now it's really exciting to see how these different approaches can be brought together, so in the latest version of Mathematica, we use Wolfram|Alpha technology to let people enter free-form natural language inputs and get precise Mathematica code out.

It's pretty interesting. It's a new way to do programming—just by using natural language. And, I am frankly quite surprised at how well it already works, and how useful even I, as an expert Mathematica programmer, find it.

JG: Unlike many software developers, you've had your application Mathematica available on the Linux platform for years. When did the first Linux version arrive, and what motivated Wolfram Research to be such an early Linux supporter?

SW: I just looked at my e-mail archive, and I see that the first internal discussion about Linux at our company happened in 1993, and we started the actual port of Mathematica to Linux in late 1994. I'd have to read through lots of messages in my e-mail archive to remember exactly how the discussion about Linux went. But in general, it's been our principle ever since the beginning that we want Mathematica to be available on as many platforms as possible.

It's good we have a well-developed Linux version of Mathematica, because nowadays all our Wolfram|Alpha servers are running Linux!

JG: Could Wolfram|Alpha or Mathematica benefit from a model similar to open-source software—that is, with a distributed group of contributors to complement your core development teams?

SW: In Mathematica, we use lots and lots of open-source software these days. In fact, all the acknowledgements for it are getting to be the length of a small book. It's wonderful when we can use a strong piece of open-source software for an algorithm or a component. It lets us put more effort into the unique parts of the Mathematica system, like algorithm automation and its overall design.

In Wolfram|Alpha, we have an increasingly large volunteer program, with people around the world helping us with particular data curation tasks that they're interested in. We're also thinking about how to develop open projects and collaborations to work on very large-scale data projects that we think would be good for the world, but which we can't afford to do ourselves.

JG: You and [the late] Steve Jobs went way back. How did that relationship come about?

SW: Steve Jobs was working in stealth mode on the NeXT computer. It must have been 1987. We met through a chain of acquaintances and ended up making a deal to have Mathematica bundled on every NeXT machine sold. It was actually the first major deal we made for Mathematica. And Steve Jobs was also the person who pushed me to use the name "Mathematica" instead of various alternatives.

JG: You were the youngest person ever to win the MacArthur "Genius" Grant/Award. What did you win it for?

SW: I got that award in 1981. When Rod MacArthur (son of the late John MacArthur) called me to tell me about it, I actually asked what it was for, and he said they didn't ever specifically say that. So I guess I don't specifically know.

I can say that in 1981, I was 21 years old, and I had mainly worked on two things. First, starting when I was 14, I had published all sorts of papers in theoretical physics—mainly about particle physics and about cosmology. I'd also become very involved in using computers to automate the calculations I needed to do. And starting in 1979, I'd been developing a symbolic manipulation software system that was in some ways a forerunner of Mathematica.

JG: From the outside, it appears that you are a person who is living the way he wishes without much compromise. Is that observation accurate?

SW: It's certainly what I aspire to, and I've spent a lot of effort doing my best to get myself into that position.

Of course, I've made plenty of choices. For example, I could work on things that are somehow maximally intellectual, but don't have practical significance and don't make money. Or I could try to make the maximum amount of money, but work on things that are intellectually rather dull. I've chosen a middle path, which I'm pretty happy with. I get to do lots of very fascinating things, but they're useful enough that they've let me make a very decent amount of money, which gives me the freedom to go on doing interesting things.

I've spent years building up Wolfram Research and collecting lots of very brilliant people there who I enjoy working with. I've never taken the company public or had investors. So I've been able to concentrate on long-term things that I think are really interesting (like Mathematica and Wolfram|Alpha). And I also don't get fired as CEO when I spend years working on basic science. (Although actually that basic science has now turned out to be crucial to our products too, but that's a different story.) My theory is that having Wolfram Research is the best way I can turn ideas that I care about into real things in the world.

I've been a remote CEO for more than 20 years now, working at home most of the time. I've ended up using technology to make my work as efficient as possible. I figure that whatever I can automate and make routine, I don't have to think about. So I can spend my thinking effort on the stuff that's really worth thinking about. And that I really enjoy.

JG: From a young age, you've been ahead of your time, regularly producing innovations in mathematics, science and computing that typically come from someone ten or 20 years older, if ever. What can you tell us about what is going on internally—your thoughts, impulses, philosophy and so on—that would help us understand how you've managed to be so ahead of your time all of your life?

SW: Thanks for the kind words! I guess I like to understand things at the most fundamental level possible, and when I succeed in doing that, it often seems fairly obvious to me what direction the things should go in. It's been very common in my life that I think about doing something, and then a decade or two later it becomes very popular.

I think I achieve the most by doing large long-term projects (although I find it a lot of fun to do the occasional short project), but one doesn't get to do that many large projects in a lifetime. One has to pick and choose. I like to pick ones that I think won't get done as part of the general development of things—ones where I can feel I'm making a unique contribution by doing them.

JG: Readers of *Linux Journal* tend to be "outside the box" kind of people. (We think you'd fit right in!) Do you have any advice for any budding Stephen Wolframs out there—people who have great ideas and want to make important contributions to their field, but aren't sure how to proceed?

SW: I think the first thing is to have confidence. Figure things out as best you can for yourself—perhaps asking lots of people for advice too—then stick to what you've concluded. At any given time, there's always a conventional wisdom about how things should be done or what's possible. If you believe something different, go with what you believe. You always can define success to be whatever it is that you achieve.

I happen to find people really interesting (a great asset if you're trying to build a company like ours!), and I've worked with lots and lots of people who've done great things. My main conclusion is that there are an awful lot of different ways to achieve things. For any given person, it's important to understand yourself as well as you can. Even after all these years, I gradually understand myself better and better, and every increment in understanding seems to help me a lot in managing to do the right things.

It's a fascinating problem matching people (including oneself) with the best possible directions and niches in the world. It's amazing how often people get stuck in the wrong niches—sometimes because they just didn't know or understand other niches that would be much better for them, and sometimes because they think they have too much investment in one niche and can't see how to bridge to another. My observation (for myself and others) is that with sufficient creativity, one always can come up with a strategy to get from here to there, and the result is much more satisfaction, fun and productivity.

JG: Do you have more big initiatives up your sleeve, or are you content right now to focus on Mathematica and Wolfram|Alpha?

SW: I've done three large projects so far in my life: Mathematica, *A New Kind of Science* (*NKS*) and Wolfram|Alpha. They're all never-ending projects, and I hope to go on pursuing all of them forever, but I also hope to do some more big projects.

I have quite a long list of projects I'd like to do, mostly very large and ambitious. The big challenge is to pick the right time to do them. I'd been thinking about the general idea of Wolfram|Alpha for several decades, but always had decided it was too early to try actually doing it. One has to wait until the ambient technology is to the right point. Some of the projects I'm considering may be close; some may be 50 years away.

One project I'm committed to doing is to use the work I've done in *A New Kind of Science* to try to find a truly fundamental theory of physics. I'm ready to dive into that project, but right now, the amazing collection of opportunities around Wolfram|Alpha are keeping me far too busy.

There are also some very, very interesting technology projects that are made possible by *NKS*; I just have to figure out when is the right decade to start them. I often think about projects for decades before I actually dive into doing them. But it's always exciting when a new project gets going, and one starts seeing how something can be created from nothing. It's always a great experience, both for me and for the people I'm working with.

JG: Thank you for your insights, and good luck in your fascinating and important work!

SW: Thanks for some very interesting questions! ■

When not working as *Linux Journal*'s Products Editor, James Gray is a conservation professional working on climate action projects around the world. He is based in Kenosha, Wisconsin.

Resources

Wolfram|Alpha: **http://www.wolframalpha.com**

Wolfram Research: **http://www.wolfram.com**

DOC SEARLS

The Near-Death of Blog Search

A once-hot category is now down to just Google. DOC SEARLS

Blogging was born in the late 1990s, and it got hot after the turn of the millennium. Several characteristics made blogs distinctive. First, they were personal. Second, they consisted of posts organized in reverse chronological order: the latest stuff on top, the older stuff scrolling toward the bottom. Third, each post had its own URL, called a permalink, which survived with its own archival path after it scrolled off the current page. Fourth, with the advent of RSS (Really Simple Syndication), blogs had a journalistic advantage: notifying the world that a post had just gone up. Thus, blogs were more current and live than ordinary Web sites, and they were designed to accumulate a corpus of work that respected the future by organizing and preserving the past.

By contrast, ordinary Web sites tended to be static and have no archival function. This was especially true of commercial sites. Last year's shoes were as gone as last year's snow.

Google and other search engines of the time paid little attention to blogs. They assumed that the Web was more like real estate than like anything that was alive and constantly changing. "Sites" that were "designed", "constructed" or "built" at "locations" you could "visit" or "browse" mattered more than "journals" that were "posted", "updated" and "syndicated".

That left open a hole in the marketplace for search engines that indexed the live Web, leaving the static one to Google and its direct competitors.

The first blog search engine was PubSub, which showed up sometime in 2002. It was inventive and took some getting used to, but it was fast and did a good job of finding current postings in the blogosphere.

Second was Technorati, which came along in late 2002, out of research

That left open a hole in the marketplace for search engines that indexed the live Web, leaving the static one to Google and its direct competitors.

David Sifry and I were doing for the article "Building with Blogs", which ran in March 2003 issue of *Linux Journal* (**http://www.linuxjournal.com/article/6497**). The first incarnation of Technorati was a MySQL hack that ran on a Debian box in the basement of David's San Francisco apartment. When he exposed it to the public on the Web, it was an instant success. Not long after that, David quit his day job and worked on building Technorati full-time.

Third was Bloglines, which came along in early 2003. Bloglines was a Web-based RSS news aggregator, rather than a search engine in the usual sense. Still, it served the need for readers to know what was being published right now on the Web.

During the next several years, other fish in the blog-search pond came to include Google Blog Search, BlogPulse, Yahoo, Feedster, IceRocket and Blogdigger.

Advertiser Index

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

ADVERTISER	URL	PAGE #
ConFoo	http://confoo.ca	107
Content & Apps for Automotive Europe 2012	http://www.telematicsupdate.com/contenteu	122
Emac, Inc.	http://www.emacinc.com	89
iXsystems	http://www.ixsystems.com	7
LinuxFest Northwest	http://linuxfestnorthwest.org	121
Linux Foundation	http://events.linuxfoundation.org	37
Lullabot	http://store.lullabot.com	2
Microway	http://www.microway.com	46, 47
Northeast LinuxFest	http://northeastlinuxfest.org	91
O'Reilly Where	http://whereconf.com/where2012	17
Posscon	http://posscon.org	29
RackMountPro	http://www.rackmountpro.com	15
Silicon Mechanics	http://www.siliconmechanics.com	3
USENIX FAST	http://www.usenix.org/events/fast12/	81

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.

This is what Twitter hath wrought, and Facebook as well. They've buried real news....

After a while fake blogs—a form of spam Mark Cuban called "splogs"—came to comprise about 99% of the blogs in the world. This was a huge problem that even Google had trouble keeping up with. Thanks to the splog issue and other problems, Feedster and Blogdigger died off. Yahoo never was serious about blog search and quit the game. IceRocket and BlogPulse both were sold and now are mostly buzz search engines that don't remember anything more than a few months old. Blogdigger's page still is up but doesn't do anything. And Technorati, which once maintained a complete index of all syndicated sources, including all blogs from the beginning of the company's existence, turned into one of those "content" mills several years back.

The only true blog search engine still operating is Google Blog Search, which basically is a specialized search in Google's main engine, which has been modified in recent years to reduce time-to-index to minutes or even seconds. I hope it keeps going, because it's an essential resource for finding the kind of news that's syndicated live, still curates itself, and isn't just about pushing or riding whatever happens to be buzzing at the moment.

For a while after Technorati gave up, my favorite blog search engine was IceRocket. Now owned by Meltwater Buzz, it's about "social media monitoring" that "helps you mine conversations across social channels for nuggets of insight". Note that the second-person "you" is not you and me, the users. We're the producers of ore from which insight nuggets are mined. The "you" they're talking to is advertisers.

This is what Twitter hath wrought, and Facebook as well. They've buried real news—stuff worth keeping around—under a mountain of buzz, all of which melts away after minutes, weeks or months.

But the durable stuff still matters. Journalism, compromised and corrupted as it has become, still matters—perhaps more than ever. That means blogs, and journals like this one, still matter too. But only to the degree that the work still can be found.■

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.

LinuxFest Northwest

Bellingham, WA
April 28th & 29th

Grassroots Linux gathering
Exhibits of all flavors
Presentations of all levels
Prizes and after party
FREE admission & parking
FREE open source software
Bring the whole family!

Hosted By
Bellingham Technical College

linuxfestnorthwest.org

Organised by:

Telematics UPDATE

The only conference focussed on content & apps for automotive

SAVE €100
Off current ticket price.
Quote discount code
1937LJ
www.telematicsupdate.com/contenteu

CONTENT & APPS FOR AUTOMOTIVE EUROPE 2012

18th-19th April 2012, Kempinski Hotel, Munich, Germany

Cloud Hosted Services and In-Car Apps Revolutionise the Driving Experience to Bridge the Ultimate Connected Lifestyle Gap

- **HYBRIDISED PLATFORMS AND OPERATING SYSTEMS ACHIEVE INTEROPERABILITY:** Utilise embedded, smartphone and cloud hosted services to blend native and cached cloud data and analyse standardised OSs, required SDKs and APIs to attract 3rd party developers to the auto space
- **OEMS TO CAPITALISE ON INCREASED CONNECTIVITY OPPORTUNITIES:** Analyse pan-European consumer requirements to integrate relevant content such as HD Radio, parking services and traffic updates whilst benefitting from increased connectivity for enhanced customer relationships
- **MOBILE NETWORK OPERATORS (MNOS) STEP-UP & REPOSITION IN-LINE WITH OEM DEMANDS:** Outline business models such as data bundling through smartphone connectivity, or, in-vehicle SIM and MNOs' plans to leverage consumer relationships through a fully managed app store complete with billing options for tried and tested content download
- **END-TO-END SERVICE OFFERINGS REVOLUTIONISED:** Discuss where traditional players can update services to retain their position in the telematics value-chain by identifying suitable content and leading the development of an ecosystem for tailored applications to fit different UIs (user interface)
- **REALISTIC IN-CAR CONTENT:** Analyse the role of social networking to develop communities to offer driver centric services and driver-to-driver interactions and gaming content, especially for rear-seat downloads to maximise your content portfolio
- **BUILD AUTO APP DEVELOPER COMMUNITIES:** Understand the app developer mind set for the auto environment and methods for increasing volume demand such as globalised apps, in-house OEM app development frameworks and requirements to qualify for OEM R&D subsidies

✓ 250+ Executive Delegates
✓ 30+ Expert Speakers
✓ 25+ Business-Focussed Sessions
✓ 20+ Hours of Supreme Networking
✓ New Speakers, New Intelligence, New Networking... NEW SHOW!

EXPERT SPEAKERS INCLUDE:

Ford, FIAT, JAGUAR, LAND ROVER, VOLVO, BMW, NOKIA, audible.com, QNX, HARMAN, MAGNETI MARELLI, telenor, DENSO, STRATEGY ANALYTICS, CHLEON, Continental, Sony Ericsson, THE LINUX FOUNDATION, waze

Badge Sponsor:

Airbiquity

“Excellent networking opportunities with well balanced debates and discussions on key industry issues.” RIM

Scan me for latest speaker additions and new sessions

Visit the website today for the latest updates on top speakers, sessions and the full conference program!

www.telematicsupdate.com/contenteu