Redis | Java/JSP | Drupal | Zmanda | GRUB2 | Google App Engine

LINUX JOURNAL™

Since 1994: The Original Magazine of the Linux Community

REVIEWED: Zmanda Recovery Manager for MySQL

DRUPAL FOR LARGE-SCALE WEB SITES

- Write Mobile Apps for iPhone and Android
- Google's Cloud Web Deployment Technology
- Transparent Firewalls with Commodity Hardware
- Changes in the GRUB2 Bootloader

+ HOW-TO: Java and JSP

SEPTEMBER 2010 | ISSUE 197
www.linuxjournal.com

$5.99US $5.99CAN

More TFLOPS, Fewer WATTS

Microway delivers the fastest and greenest floating point throughput in history

Enhanced GPU Computing with Tesla Fermi

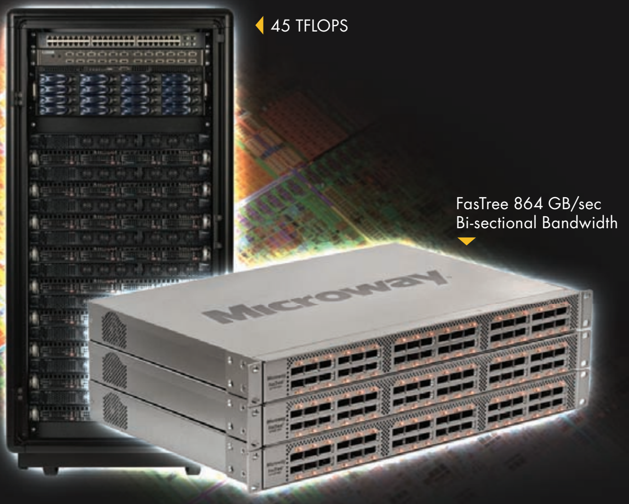
- 480 Core NVIDIA® Tesla™ Fermi GPUs deliver 1.2 TFLOP single precision & 600 GFLOP double precision performance!
- New Tesla C2050 adds 3GB ECC protected memory
- New Tesla C2070 adds 6GB ECC protected memory
- Tesla Pre-Configured Clusters with S2070 4 GPU servers
- WhisperStation - PSC with up to 4 Fermi GPUs
- OctoPuter™ with up to 8 Fermi GPUs and 144GB memory

New Processors

- 12 Core AMD Opterons with quad channel DDR3 memory
- 8 Core Intel Xeons with quad channel DDR3 memory
- Superior bandwidth with faster, wider CPU memory busses
- Increased efficiency for memory-bound floating point algorithms

Configure your next Cluster today!

www.microway.com/quickquote
508-746-7341



FasTree™ QDR InfiniBand Switches and HCAs

- 36 Port, 40 Gb/s, Low Cost Fabrics
- Compact, Scalable, Modular Architecture
- Ideal for Building Expandable Clusters and Fabrics
- MPI Link-Checker™ and InfiniScope™ Network Diagnostics

Achieve the Optimal Fabric Design for your Specific MPI Application with ProSim™ Fabric Simulator

Now you can observe the real time communication coherency of your algorithms. Use this information to evaluate whether your codes have the potential to suffer from congestion. Feeding observed data into our IB fabric queuing-theory simulator lets you examine latency and bi-sectional bandwidth tradeoffs in fabric topologies.

NVIDIA. TESLA CUDA

intel Channel Partner Premier 2010

AMD Fusion Partner Premier

GSA

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

Microway®
Technology you can count on℠

1 YEAR FOR FREE!*

You spoke, we listened. With over 1,000 in-house developers, 1&1 has been working hard to create feature-packed website plans to meet the needs of even the most demanding web professional.

1&1® FREE HOSTING PACKAGE

- 2 Domain Names Included (.com, .net, .org, .info or .biz)
- 150 GB Web Space
- UNLIMITED Traffic
- 10 FTP Accounts
- 25 MySQL Databases
- Extensive Programming Language Support: Perl®, Python®, PHP 5/6 (beta) with Zend® Framework
- 1,200 E-Mail Accounts (IMAP/ POP3)

1 YEAR FREE

After the 1st year, pay just $6.99 per month*

Hurry, Offer Ends August 31, 2010!

1&1®

MEMBER OF united internet

Get started today, call 1-877-GO-1AND1

www.1and1.com

*Offer valid through August 31, 2010 and applies to the Home Package only. 24 month minimum contract term and setup fee of $4.99 apply. Visit www.1and1.com for full promotional offer details. Program and pricing specifications and availability subject to change without notice. 1&1 and the 1&1 logo are trademarks of 1&1 Internet AG, all other trademarks are the property of their respective owners.

CONTENTS

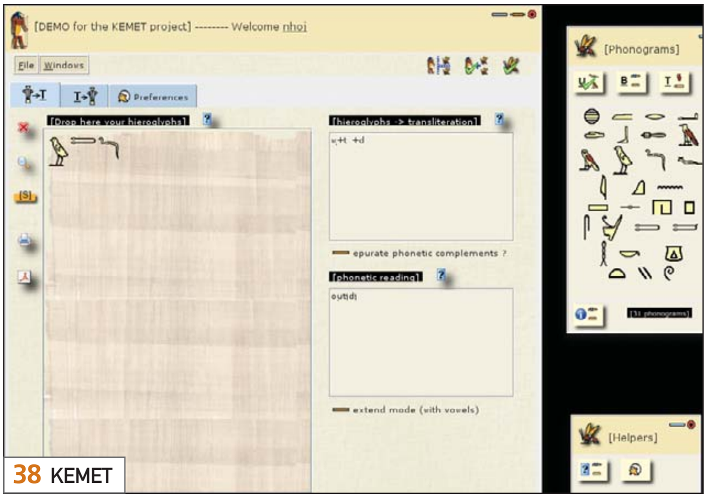
SEPTEMBER 2010
Issue 197

FEATURES

WEB DEVELOPMENT

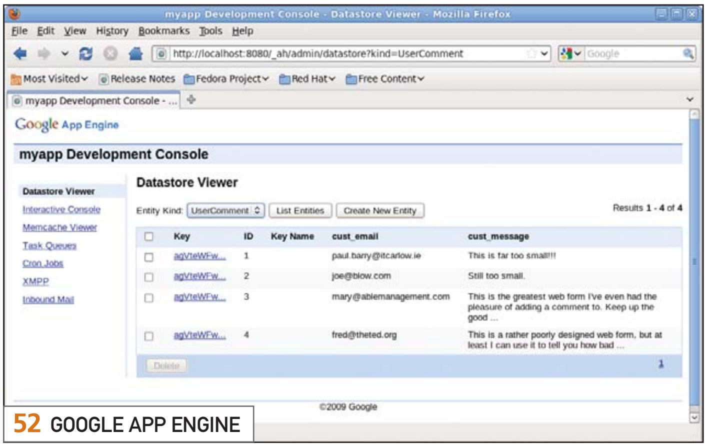
ON THE COVER

CONTENTS SEPTEMBER 2010 Issue 197


[DEMO for the KEMET project] -------- Welcome nhoi
File
Windows
Preferences
[Drop here your hieroglyphs]
[hieroglyphs -> transliteration]
epurate phonetic complements ?
[phonetic reading]
extend mode (with vowels)
[Phonograms]
[31 phonograms]
[Helpers]

38 KEMET

39 VIRTMUS



http://localhost:8080/_ah/admin/datastore?kind=UserComment

myapp Development Console

Datastore Viewer

Entity Kind: UserComment

Results **1 - 4** of **4**

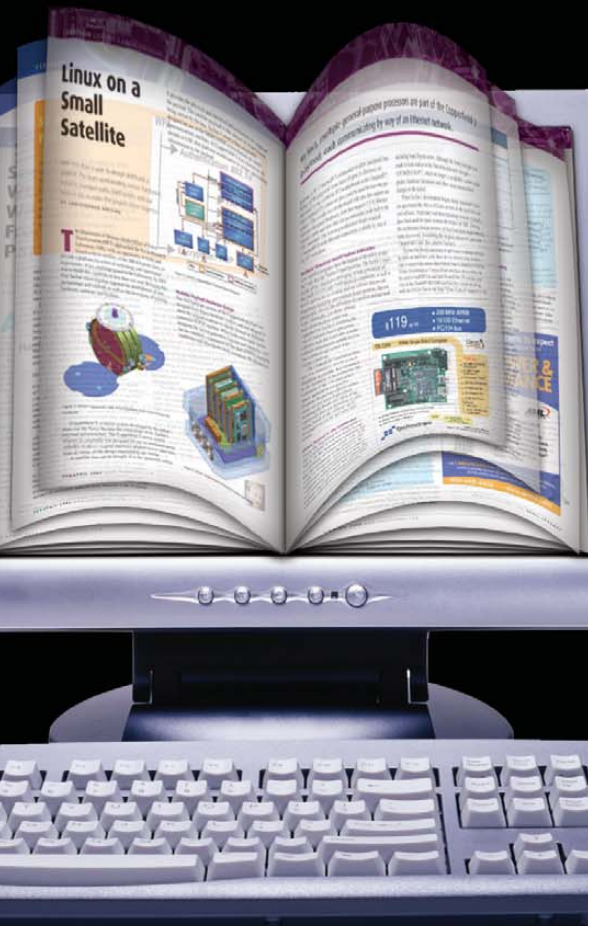
□	Key	ID	Key Name	cust_email	cust_message
□	agVteWFw...	1		paul.barry@itcarlow.ie	This is far too small!!!
□	agVteWFw...	2		joe@blow.com	Still too small.
□	agVteWFw...	3		mary@ablemanagement.com	This is the greatest web form I've even had the pleasure of adding a comment to. Keep up the good ...
□	agVteWFw...	4		fred@theted.org	This is a rather poorly designed web form, but at least I can use it to tell you how bad ...

1

©2009 Google

52 GOOGLE APP ENGINE

USPS *LINUX JOURNAL* (ISSN 1075-3583) (USPS 12854) is published monthly by Belltown Media, Inc., 2211 Norfolk, Ste 514, Houston, TX 77098 USA. Periodicals postage paid at Houston, Texas and at additional mailing offices. Cover price is $5.99 US. Subscription rate is $29.50/year in the United States, $39.50 in Canada and Mexico, $69.50 elsewhere. POSTMASTER: Please send address changes to *Linux Journal*, PO Box 16476, North Hollywood, CA 91615. Subscriptions start with the next issue. Canada Post: Publications Mail Agreement #41549519. Canada Returns to be sent to Bleuchip International, P.O. Box 25542, London, ON N6C 6B2



LINUX JOURNAL

Executive Editor Jill Franklin
jill@linuxjournal.com
Senior Editor Doc Searls
doc@linuxjournal.com
Associate Editor Shawn Powers
shawn@linuxjournal.com
Associate Editor Mitch Frazier
mitch@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

General Manager Rebecca Cassity
rebecca@linuxjournal.com

Senior Print Media Sales Manager Joseph Krack
joseph@linuxjournal.com

Digital Media Sales Manager Michael Beasley
michael@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

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
PHONE: +1 818-487-2089
FAX: +1 818-487-4550
TOLL-FREE: 1-888-66-LINUX
MAIL: PO Box 16476, North Hollywood, CA 91615-9911 USA
Please allow 4–6 weeks for processing address changes and orders
PRINTED IN USA

LINUX is a registered trademark of Linus Torvalds.

PRINTED WITH SOY INK

MPLS for the masses

$39 95

Usually MPLS routers cost more than $1000, but not anymore. MikroTik gives you the ability to use MPLS in any network. No more big box prices for MPLS! A chicken in every pot!

MPLS stands for Multi Protocol Label Switching. It can be used to replace IP routing - packet forwarding decision is no longer based on fields in IP header and routing table, but on labels that are attached to the packet.

MPLS makes it easy to create "virtual links" between nodes on the network, regardless of the protocol of their encapsulated data. It is a highly scalable, protocol agnostic, data-carrying mechanism. MPLS allows one to create end-to-end circuits across any type of transport medium, using any protocol.

Features:

- Label Distribution Protocol for IPv4
- Virtual Private Lan Service
  * VPLS LDP signaling
  * VPLS MP-BGP based autodiscovery and signaling
  * split-horizon bridging
- RSVP TE Tunnels
  * explicit paths
  * CSPF path selection
  * OSPF extensions for TE tunnels
- Virtual Routing and Forwarding
- MP-BGP based MPLS IP VPN
- OSPF and RIP as CE-PE protocols

Benefits:

- higher speed forwarding in network core
- ability to implement transparent L2 and L3 VPNs (VPLS & VRF)
- reduced VPN overhead compared to legacy tunneling solutions
- traffic engineering to implement QoS and optimize network usage
- ability for the ISP to create VPNs without user interaction
- separate tunnels for voice, video, or data

All MikroTik RouterBOARDs support MPLS, including the **RB750** which costs $39.95 The RB750 is a SOHO router with a 400MHz Atheros CPU, five ethernet ports, plastic case and PSU. With MPLS, RB750 is capable of wire speed throughput for 1000byte packets and up, maximum 80000 pps with smaller packets.

MikroTik routerboard

www.mikrotik.com/mpls

SHAWN POWERS

Developing Webs, Even If You're Not a Spider

Ever since the early 1990s, we've been stuck on the Web like a fly visiting a spider. Of course, for us, the Web is a useful medium for information delivery, and no giant spider is coming to eat us (depending on the Web sites we visit, I suppose). Although our passion for the Web hasn't ebbed during the past two decades, the Web itself has changed drastically. This month, we focus on Web development. It's exciting to see how integral the Linux operating system is to the Internet, and as the Web changes, so does the way we develop for it.

Behind most good Web applications, there is a database humming along providing data to the user quickly and efficiently. Reuven M. Lerner shows us Redis, which is a high-speed storage and caching system for databases. It's a bit like memcached on steroids. Be sure to take a look if your database could use a speed boost (and really, what database doesn't?). Thankfully, Daniel Bartholomew follows Reuven with a one-two punch and gives us a review on the Zmanda Recovery Manager. The fastest database in the world is useless if you can't recover its data from a disaster, so you'll want to read Daniel's article before going into production.

Although databases are important for any good Web application, for end users, they're about as exciting as watching paint dry—that's where user interfaces come in. Thankfully, many content management systems exist to do all the heavy-lifting for us. Jerad Bitner and Nate Haug show off Drupal this month. The LinuxJournal.com Web site runs Drupal, so we can attest to how wonderful it is for managing large Web sites. Jerad and Nate explain how Drupal can do the same for your Web site, and they provide some tips and tricks to make it perform well regardless of how big your site might become.

For many developers, simply managing content isn't what they need to accomplish. In that case, we've given you a couple different ways to tackle your specific problem. Paul Barry demonstrates how to use App Engine. App Engine is a way to create webapps on Google's infrastructure, completely free. (Well, if your webapp becomes extremely popular, Google will charge you, but initially it's free, which is a price that's hard to beat.) Google's App Engine is extremely flexible and constantly improving, and Paul shows the ins and outs of this relatively new technology. If its newness or its Googliness turns you off, perhaps Christopher Schultz's article on developing Web applications with Java/JSP will be more what you're looking for in a platform. Java has been around for a long time, but that doesn't mean it's old-fashioned. Christopher shows how to make cutting-edge programs in a time-tested language.

Many of us aren't developers at all. I'm certainly not, and yet I still look forward to the Web development issue because I can point my developer friends at new ways to make my life as an end user more exciting. Rick Rogers, for instance, walks through the process for developing portable Web applications for Internet-enabled devices. Whether you use an Android phone or an iPad tablet, or if you just prefer to run mobile apps on your computer (a little user-agent trickery in your browser usually can help there), Rick's article is one you'll want your developer friends to check out. A beautiful Web page is great, but when you're looking at a three-inch screen, it's nice to have a viewing experience designed for such small real estate.

Finally, this is *Linux Journal*. If this month's issue focus isn't quite your cup of tea, we still have tons of stuff to feed your Linux addiction. Kyle Rankin shows us the ropes with GRUB2, a significant change from the GRUB we all know and love. Dave Taylor teaches us about exit codes to help make our shell scripts a little smarter. Mick Bauer continues his series on transparent firewalls. Even I get into the act with some tips on starting a LUG in your area. Add to that our regular lineup of tech tips, letters to the editor and new product announcements, and you've got an issue bound to inform and entertain. The great thing about Web development with Linux is that no matter how long you stay tangled in this issue, no giant spider will come to eat you. We hope.■

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.

FRONT
REAR
DOUBLE SIDED STORAGE SERVER
80 PLUS GOLD

Zabbix | CouchDB | Eucalyptus | Quickly | MySQL | Coyote Point
LINUX JOURNAL
Since 1994: The Original Magazine of the Linux Community
How-to: Use Zabbix for Server Monitoring
Linux vs. MS Windows in the Enterprise
Get Started with Quickly
Enterprise
Make a CLOUD COMPUTING CLUSTER
MySQL Replication for Data Centers
Is the Enterprise Ready for Open Source?
Database Debate: SQL vs. NoSQL
REVIEWED: Coyote Point E650GX Load-Balancing Appliance

Rant

I just read Dave Taylor's column "Simple Scripts to Sophisticated HTML Forms, Take II" in the July 2010 issue. I don't know how to ask this nicely, so I'll just ask it. If you're writing for *Linux Journal*, why are you writing your scripts on Apple OS X and not a Linux distribution? If you don't want to use Linux as your primary OS, that's fine, but at least do us the service of firing up a virtual machine with Linux. Otherwise, great series.

--
Jason Froebe

Dave Taylor replies: *A fair question! I have long ago ascertained, however, that from a command-line perspective, any *nix that is POSIX-compliant is going to be functionally identical. Although I certainly can—and have in the past—install a variety of different Linuxes on my computers, I instead test with the NetBSD-based Mac command line that's part of Mac OS X and double-check on the FreeBSD server I have running my main Web sites. Are they Linux systems, per se? No. Does it matter? I don't think so. What do you think?*

Thanks for the Nokia N900 Review

I was so impressed with the review Kyle Rankin wrote about the N900 that I jumped onto the Web and started reading about this wonderful product (see the May 2010 issue). The more things I read about it, the more I wanted it. So now thanks to Kyle Rankin, I own one. FYI, the latest update helped the GPS a lot. Also, there is now a turn-by-turn navigation app, Mobile Maps. It's $49.99 euros, and around $61 US. Thank you Kyle Rankin!!

--
jpm1973

Kudos for "Adventures in Scanning"

Thank goodness for Dirk Elmendorf's "Adventures in Scanning" in the June 2010 issue. Like Dirk, I too, have shied away for quite some time from scanners in Linux, and especially multifunction devices. However, I'd been thinking more and more about moving on from my plain-vanilla laser printer, but well, I was still chicken-hearted. Dirk's column has helped bolster my courage a lot! Multifunction printers, onward!

--
Kay Schenk

Using Telnet to Send E-mail

Regarding Torsten's Letter and Kyle Rankin's reply in the July 2010 issue, note that the actual SMTP RFC is RFC-5321 and not 2821. It was updated in 2008. See **www.rfc-editor.org/cgi-bin/rfcsearch.pl?searchwords=5321&opt=Number&num=25**.

--
Wayne Pollock

Plone

I would like *Linux Journal* to print an article on Plone, because one, you haven't printed anything on it, and two, I would like more people to know about it. Also, I bought the June issue of *Linux Journal* at Barnes & Noble, and I am thinking about subscribing to it. Although I am 11, I learned lots about why there are so many distributions and how they are so different. I use Linux Mint 9.

--
Stephen McIntosh

It has been a while since we've had articles on Plone in the magazine, but if you check out our Web site, you'll find some articles from past issues. I'm glad you enjoyed the June issue, and hope we continue to entertain and inform you. Be sure to check out our Web site too, there are many Web-only topics that come out quite regularly. Welcome to the family!—Ed.

Thanks for the *LJ* Archive CD

I have been a subscriber for so long I can't remember, and I've kept all my *LJ* magazines. Unfortunately, due to the preparation of renovating my parent's house and me moving into a smaller apartment, I have no space for my *LJ* collection (amongst other things). I had to, in one fell swoop, throw out all my magazines. Thankfully, you guys came out with the *LJ* CD. Now I can read all the magazines even if I no longer own the printed copies. In any event, thank you very much for *LJ*. I love it. Keep up the great work!

--
Edmund Wong

GASP! Seriously though, my wife often asks why I keep multiple copies of each Linux Journal *issue, and my response is usually something deep and meaningful like, "Because!" She's seldom impressed. Still, I understand your pain, and I'm glad the CD helps soothe the burn a bit. Thanks for kind letter.—Ed.*

Maintaining Automatically Generated Files

In the June 2010 Hack and / column, Kyle Rankin presents a possible nightmare to the next sysadmin after him. The script automatically generates a config file, complete with comments that say "add any extra munin options for each host here". When run in a cron.daily fashion, however, this means someone else may happen across the config file, change it, and then be perplexed the next day as the changes vanished.

Anything that is auto-generated should have such comments replaced with a header comment indicating that it is generated automatically, the source of the generation and a timestamp. Help out the next poor guy who inherits the system (or even yourself), as a year later, you might not remember what you did.

--
David Morton

The Future of Linux: the Average Windows User

An open letter to the Linux community:

I consider myself the above-average Windows user. I've taken and passed A+ Certification and some Windows networking courses. I have a basic, big-picture understanding of the DOS command line and Windows. I don't like anyone, such as Windows or Google, monopolizing control over their clients, so when Linux came along in the 1990s, I wanted it to succeed. I have loved the idea of a free computer operating system for everyone that Linux brought to us. When I say free, I mean freedom from being controlled by Microsoft or its peers. I don't mind paying for an OS that offers freedom from control. So I attempted to use the Red Hat distribution around 2000 to no avail. Now, ten years later, I'm at it again with Ubuntu. I love it. There are so many aspects to Ubuntu that I like better than Windows. One is that there is so much free software for nearly anything you need/want to accomplish. The problem I see is the fact that many hardware devices and software are not compatible.

If Linux is to make headway into the individual consumer market, these compatibility issues *must* be overcome. I would like to ditch Windows totally, but at the present time, this isn't feasible. I have listed some of the current obstacles that I see and objectives the Linux community needs to work toward for it to be viable to the masses:

- Compatibility with *all* Linux software and hardware devices, no matter what Linux distribution the consumer is using. The Linux community needs to reach out as one body to hardware and software manufacturers with uniform standards that apply, no matter what distribution is being used. This leads to the next item.

- The various distributions *must* tear down the walls between their peers and work together for the common good, which is Linux for all.

- A 100% GUI interface with no need for using the terminal, unless the user would like to use it specifically. This, I believe, is the major obstacle we have to scale over together, and it's especially needed for installation.

I hope this gives the Linux community some ideas on how to make Linux better for and easily used by the nontechnical, common user.

--
Christian

Thank you for your letter. I love to see people share my passion for free and open software. Although I largely agree with your points, one of the unavoidable consequences of freedom is the freedom everyone has to do things differently.

I wish there were a common packaging system (I'd pick .deb myself). I wish there were a common GUI interface. I wish there were common configuration tools. I fear, however, that if we "forced" compliance to standards, we'd lose the very freedom we love. It's a wickedly double-edged sword, and I'm not sure I can come up with an ideal answer. Hopefully, your letter and my response will get people thinking about such things.—Ed.

Friends Don't Let Friends Use Windows

My home is 100% Microsoft-free. My primary desktop runs Mint-9-amd64, my old laptop runs Ubuntu 9.04, and my Dell Mini 9 Netbook runs Xubuntu 9.10. I have tried many other distros, but I like these the best. I have a friend who wants to switch to Linux and even tinkers with Ubuntu 10.04 within VMware on his Windows 7 machine, but he just can't make the switch, mostly due to a dependence upon one specific Windows application, but I also suspect it's a healthy fear of the unknown.

I recently invited this friend to join me at an upcoming local Linuxfest, but he declined due to schedule conflicts. I fear that he will never make a "cold-turkey" switch to Linux, so I gave him the advice below. I'm sharing it in this message because other readers may find it useful for helping struggling converts (the rest of the letter is the advice to my friend).

I went back and forth between Windows and Linux over the years. Each time I was driven back to Windows because of a perception that some critical application existed only in Windows and frustration when some things seemed to be harder to do in Linux. Over time, I gradually switched to Windows versions of many of the open-source packages used in Linux and discovered that, with the familiarity of use, I grew comfortable with the open-source applications, such as OpenOffice.org, GIMP, Gramps, GnuCash and Scribus and so on. At some point, I realized I wasn't using Windows for anything but a program launcher for open-source software and that there was really no reason to continue to use Windows. At that point, I switched 100% to Linux and left Microsoft behind forever.

If you really would like to abandon Microsoft, I recommend following a similar path. Start using multiplatform open-source equivalents of your current programs, and after a period of acclimation, you too will realize that you don't really need Microsoft Windows anymore.

Start with OpenOffice.org. It is as good as Microsoft Office, just a little different here and there. The learning curve is not steep, and it's worth the effort. After OpenOffice.org, switch, one at a time, to other open-source solutions. First, read this: **tinyurl.com/2xju2m**.

--
Ed Comer

Great advice! Switching to open-source alternatives is a great way to make the transition. Also, your link supplies great information. Thanks!—Ed.

Rockbox *Isn't* Linux!

I thought I'd better drop a note in case no one else offers a correction to Dan Sawyer's assertion in his "Philosophy and Fancy" article in the June 2010 issue of *Linux Journal*. He says at one point: "No discussion of the different approaches would be complete without mentioning embedded distributions—versions of Linux and derivative operating systems (such as Rockbox and Android) designed to run on handheld devices, in networking appliances, NAS servers and dozens of other gadgets, toys, tools and machines that consumers love to use and hackers love to repurpose."

Unfortunately, Dan makes a mistake that for some reason a lot of people do, which is that Rockbox is in some way based on Linux. It's not—not even remotely. The kernel has been written entirely from scratch, and we don't ship a "distribution" either. We include no GNU tools as they'd be recognized by the GNU Project. (And I say that merely because I can't think of *any* that we ship, but I don't want to claim something that later turns out to be slightly untrue!) I imagine Mr Torvalds would be quite insulted if he ever found out people were of the opinion that he'd written it! About the only thing it has in common with Linux is that it's vaguely POSIX (although not nearly as much as Linux itself is these days).

We at the Rockbox Project are constantly flummoxed at the number of people who seem convinced that Rockbox is based on Linux. We've certainly never said it is! Indeed, if Dan can point us to the source of his own confusion on this matter, we'd love to set whomever straight too!

Other than that, the article was an interesting read, and I thank Dan for the rest of the time he invested in writing it!

--
Bryan

Dan Sawyer replies: *You are, of course, correct about Rockbox. In my tracing of philosophical links and the cross-fertilization among open-source projects, I failed to portray the situation accurately. Thanks very much for the note!*

Dynamic nmap + xymon

Great Hack and / article by Kyle Rankin in the June 2010 issue of *Linux Journal* covering the grepable output function of nmap and culminating in your own dynamic configuration file for Munin. What's so cool about this is it's 1) simple, 2) easy and, most important, 3) brilliantly easy and simple!

I've actually borrowed your idea to do the same thing on the networks I administer at work using xymon (**www.xymon.com/hobbit/help/about.html**).

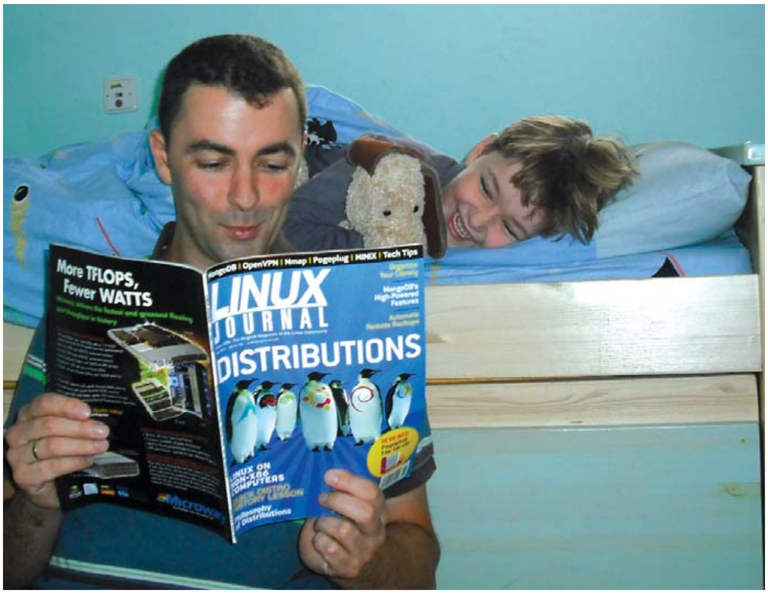
You know as well as I, there can be *a lot* of steps involved in getting servers deployed in any environment (even when it's highly automated), but I sincerely dig shortcut ideas like this to make my life a bit easier in the system administration realm.

Keep the cool tips coming, friend!

--
Adam

PHOTO OF THE MONTH

Have a photo you'd like to share with *LJ* readers? Send your submission to publisher@linuxjournal.com. If we run yours in the magazine, we'll send you a free T-shirt.



Do you think six years old is old enough to switch from *The Wizard of Oz* to really interesting stuff? Submitted by Alexander Sirotkin.

LINUX JOURNAL

At Your Service

MAGAZINE

PRINT SUBSCRIPTIONS: Renewing your subscription, changing your address, paying your invoice, viewing your account details or other subscription inquiries can instantly be done on-line, **www.linuxjournal.com/subs**. Alternatively, within the U.S. and Canada, you may call us toll-free 1-888-66-LINUX (54689), or internationally +1-818-487-2089. E-mail us at subs@linuxjournal.com or reach us via postal mail, Linux Journal, PO Box 16476, North Hollywood, CA 91615-9911 USA. Please remember to include your complete name and address when contacting us.

DIGITAL SUBSCRIPTIONS: Digital subscriptions of *Linux Journal* are now available and delivered as PDFs anywhere in the world for one low cost. Visit **www.linuxjournal.com/digital** for more information or use the contact information above for any digital magazine customer service inquiries.

LETTERS TO THE EDITOR: We welcome your letters and encourage you to submit them at **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.

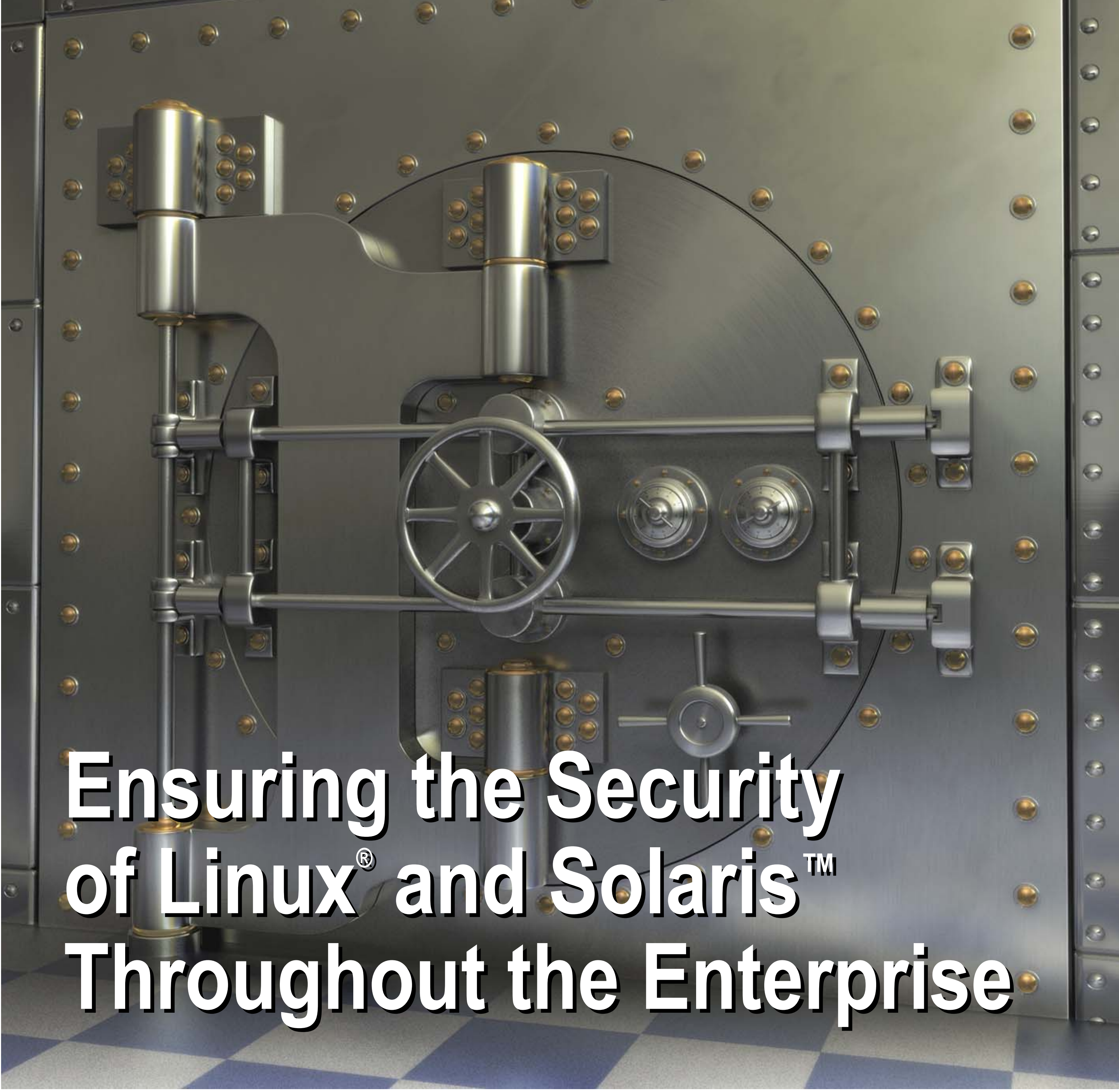
WRITING FOR US: We always are looking for contributed articles, tutorials and real-world stories for the magazine. An author's guide, a list of topics and due dates can be found on-line, **www.linuxjournal.com/author/index**.

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, **www.linuxjournal.com/advertising**. Contact us directly for further information, ads@linuxjournal.com or +1 713-344-1956 ext. 2.

ON-LINE

WEB SITE: Read exclusive on-line-only content on *Linux Journal*'s Web site, **www.linuxjournal.com**. Also, select articles from the print magazine are available on-line. Magazine subscribers, digital or print, receive full access to issue archives; please contact Customer Service for further information, subs@linuxjournal.com.

FREE e-NEWSLETTERS: Each week, *Linux Journal* editors will tell you what's hot in the world of Linux. Receive late-breaking news, technical tips and tricks, and links to in-depth stories featured on **www.linuxjournal.com**. Subscribe for free today, **www.linuxjournal.com/enewsletters**.


Ensuring the Security
of Linux® and Solaris™
Throughout the Enterprise

diff -u

WHAT'S NEW IN KERNEL DEVELOPMENT

Some new documentation is available for **SysFS** and **libudev**. **Alan Ott** couldn't find the docs he wanted, so he wrote some of his own and put them up at **www.signal11.us/oss/udev**. The relationship between those tools is that SysFS presents a filesystem interface to view kernel and hardware status and to edit configuration options, while libudev presents a C library to track SysFS's changes and to modify the various configuration options. Through the years, a lot of attempts have been made to create a consistent interface to present hardware and kernel options to the user. SysFS is one of the most recent, and it seems to be the one that is gradually taking over from **ProcFS**, **ioctls** and all the older mechanisms.

Jesse Barnes has been trying to give **kernel panics** a better chance of presenting screen output when the user has been running the **X Window System**. Typically, running X means a panic just won't produce visible output, which in turn means that a meaningful bug report is much harder to create. Jesse's code improves the situation in some cases, but if X has disabled the display, his patch still won't cause panic output to appear on the screen. There was not much immediate interest displayed in his patches, which could mean that **Linus Torvalds** and **David S. Miller** just haven't had a chance to look at them yet, or it could mean they think he's on the wrong track. It's cool that someone's looking into better panic output though.

The old **GCC 3.x compiler** is having more and more trouble compiling the Linux kernel, and folks like **H. Peter Anvin** are getting less and less enthused about fixing all those problems as they turn up. Recently, there was talk about just dumping support for that compiler, at least for the x86 platform. It seems clear that very few people still use GCC 3.x to compile current Linux releases, although there probably are some. But, even as one group of developers moved more in the direction of deprecating GCC 3.x and eventually abandoning support for it, another group of developers seemed to gain interest in preserving support for GCC 3.x. As **Eric Dumazet** put it, if there's no significant technical reason to drop support, the mere fact of GCC 3.x being "old" didn't seem like a good enough reason, especially as the work involved in maintaining support was not so extreme.

TmpFS has a speed issue, because if multiple threads try to access a mounted TmpFS filesystem, they run into so much lock contention that the filesystem slows down considerably. **Tim Chen** and various other folks implemented a "token jar" to handle the lock contention in TmpFS and saw 270% speed increases on some of their tests. **Andi Kleen** liked the patches and said their token-jar implementation might also be useful elsewhere in the kernel. So, it looks like upcoming kernels will include a much faster TmpFS.

—ZACK BROWN

LinuxJournal.com—Under the Hood

Since this issue focuses on Web development, I'd like to take the opportunity to provide a glimpse behind the scenes at LinuxJournal.com. Many of you already know that LinuxJournal.com is largely made possible by the Drupal platform, aka my favorite open-source project. Many also have asked to learn more about the specifics of our Drupal setup, and to that end, I'd like to share some of my favorite Drupal modules that power LinuxJournal.com:

1. Views: this one is pretty obvious to anyone who has used Drupal at all, but stating the obvious never hurts anyone. Views is *the* absolutely essential query-building module. In my opinion, you cannot build a Drupal site without this module (**drupal.org/project/views**).

2. CCK: again, I state the obvious, but the content construction kit allows you to add fields to content, allowing you to build custom types of content for almost any data imaginable. I can't imagine a Drupal world without CCK (**drupal.org/project/cck**).

3. Flag: the Flag module is brilliant in its simplicity—a simple yes or no, on or off, 1 or 0 to almost anything on your Drupal site. The possibilities are endless (**drupal.org/project/flag**).

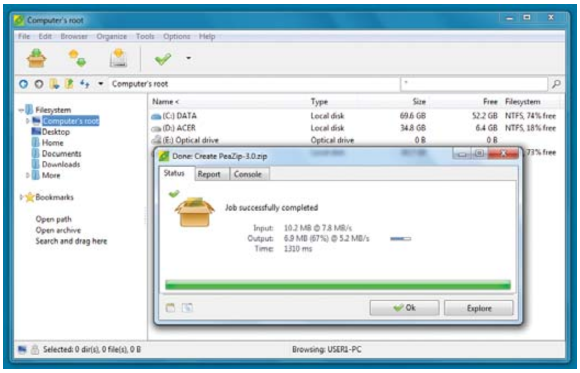
4. Views Attach: what could be better than attaching a list of data to a user or individual piece of content? In my experience, the answer to "How do I display ____?" frequently involves Views Attach. I highly recommend giving this little module a spin to see if it works for you (**drupal.org/project/views_attach**).

5. Mollom: last, but certainly not least, Mollom is the module that keeps me relatively sane. Spam makes LinuxJournal.com less cool, and must, therefore, be destroyed. I hate spam. Mollom gets rid of spam Thanks, Mollom (**drupal.org/project/mollom**)!

These five modules are some of my favorites, but there are lots more where they came from, so I hope you'll visit LinuxJournal.com to read more about these and other great modules, as well as other tidbits from my adventures in Drupaling. Just go to LinuxJournal.com and search "Drupal". See you there!

—KATHERINE DRUCKMAN

NON-LINUX FOSS



Name <	Type	Size	Free	Filesystem
(C:) DATA	Local disk	69.6 GB	52.2 GB	NTFS, 74% free
(D:) ACER	Local disk	34.8 GB	6.4 GB	NTFS, 18% free
(E:) Optical drive	Optical drive	0 B	0 B	
				73% free

Job successfully completed

Input: 10.2 MB @ 7.8 MB/s
Output: 6.9 MB (67%) @ 5.2 MB/s
Time: 1310 ms

Selected: 0 dir(s), 0 file(s), 0 B   Browsing: USER1-PC

PeaZip Zipping PeaZip

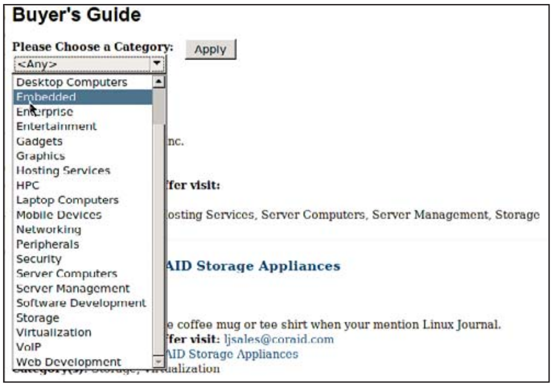
Hardly a day goes by without the need to compress or uncompress something. And, plenty of compression/decompression programs exist, but if you like the idea of using the same tool on multiple platforms and you like open source, you should consider PeaZip. PeaZip runs on Windows and Linux, and because it's written with Free Pascal and Lazarus, the Linux version comes in both a GTK2 (GNOME) flavor and a Qt (KDE) flavor.

PeaZip can create the standard types of compressed files/archives: ZIP, GZ and 7Z. Plus, it creates a few that you don't often see in a desktop GUI tool: BZ2 and TAR. In addition, it creates ARC, PAQ/ZPAQ, PEA, QUAD/BALZ and UPX files/archives. On the decompression side, PeaZip goes ballistic and handles (currently) 123 different archive/file types. In other words, if it's compressed/archived, it's unlikely that PeaZip won't be able to deal with it.

PeaZip is hosted on SourceForge at **peazip.sourceforge.net**. There are installers for Windows (32- and 64-bit). There also are RPMs and DEBs for Linux/GTK2 or Linux/Qt. And, if you're a Lazarus type, you can grab the source. PeaZip also provides localizations for dozens of other languages.

—MITCH FRAZIER

Don't Buy Candy from the Car Salesman



Buyer's Guide

Please Choose a Category:

nc.

ffer visit:

osting Services, Server Computers, Server Management, Storage

AID Storage Appliances

e coffee mug or tee shirt when your mention Linux Journal.
ffer visit: ljsales@coraid.com
AID Storage Appliances
alization

If you want to find a double-chocolate truffle, chances are you would shop at a place that specializes in making candy. Sure, the used-car salesman might have a jar of cheap candies he's giving away, but for serious chocolate-lovers, nothing compares to confections made by experts. The same thing is true with computer equipment. No, there aren't free jars of server blades at the used-car lot, but when you buy hardware or software, you want to buy it from someone who specializes in your operating system—in our case, Linux.

Over at the *Linux Journal* Web site, our very own Joe Krack keeps a handy database of vendors, systems and even sales promotions from Linux-friendly companies. They are vendors we've personally worked with, and they offer deals unique to *Linux Journal* readers. It's not a big list of advertisements; rather, it's a big list of products from companies we trust. Check it out over at **www.linuxjournal.com/buyersguide**. To be fair, I wouldn't recommend buying candy from them either. The "chips" they sell aren't chocolate chips.

—SHAWN POWERS

LJ Index September 2010

1. Millions of active .com domain names: **87.7**

2. Millions of active .net domain names: **13.1**

3. Millions of active .org domain names: **8.5**

4. Millions of active .info domain names: **6.4**

5. Millions of active .biz domain names: **2.1**

6. Millions of active .us domain names: **1.7**

7. Thousands of new .com domains registered per day: **51.6**

8. Thousands of new .net domains registered per day: **7.6**

9. Thousands of new .org domains registered per day: **7.1**

10. Thousands of new .info domains registered per day: **9.9**

11. Thousands of new .biz domains registered per day: **2.2**

12. Thousands of new .us domains registered per day: **2.3**

13. Percent of registered domains managed by top domain registrar (GoDaddy): **30.3**

14. Percent of registered domains managed by 2nd top domain registrar (Enom): **8.3**

15. Percent of registered domains managed by 3rd top domain registrar (TuCows): **6.7**

16. Percent of registered domains managed by 4th top domain registrar (Network Solutions): **5.7**

17. Millions of domains in country with largest number of domains (US): **71.4**

18. Millions of domains in country with 2nd largest number of domains (Germany): **6.2**

19. Millions of domains in country with 3rd largest number of domains (UK): **4.2**

20. Millions of domains in country with 4th largest number of domains (China): **3.9**

Sources:
1–12: domaintools.com / 13–20: webhosting.info

The Web on the Console

Most people think “graphical interfaces” when they think of surfing the Web. And, under X11, there are lots of great programs, like Firefox or Chrome. But, the console isn’t the wasteland it might seem. Lots of utilities are available for surfing the Web and also for downloading or uploading content.

Let’s say you want to surf the Web and find some content. The first utility to look at is also one of the oldest, the venerable Lynx. Lynx actually was my first Web browser, running on a machine that couldn’t handle X11. In its most basic form, you simply run it on the command line and give it a filename or a URL. So, if you wanted to hit Google, you would run:

```
lynx http://www.google.com
```

Lynx then asks you whether you want to accept a cookie Google is trying to set. Once you either accept or reject the cookie, Lynx loads the Web page and renders it. As you will no doubt notice, there are no images. But, all the links and the text box for entering search queries are there. You can navigate from link to link with the arrow keys. Because the layout is very simple and text-based, items are in very different locations on the screen from what you would see when using a graphical browser.

Several options to Lynx might be handy to know. You can hand in more than one URL when you launch Lynx. Lynx adds all of those URLs to the history of your session and renders the last URL and displays it. When you tested loading Google above, Lynx asked about whether or not to accept a cookie. Most sites these days use cookies, so you may not want to hear about every cookie. Use the option `-accept_all_cookies` to avoid those warning messages. You can use Lynx to process Web pages into a readable form with the option `-dump`, which takes the rendered output from Lynx and writes it to standard out. This way, you can process Web pages to a readable format and dump them into a file for later viewing. You can choose what kind of key mapping to use with the options `-vikeys` or `-emacskeys`, so shortcut keys will match your editor of choice.

Lynx does have a few issues. It has a hard time with HTML table rendering, and it doesn’t handle frames. So, let’s look at the Links browser. Links not only works in text mode on the command line, but it also can be compiled to use a graphics display. The graphics systems supported include X11, SVGA and framebuffer. You can select one of these graphics interfaces with the option `-g`. Links also can write the rendered Web pages to standard output with the `-dump` option. If you need to use a proxy, tell Links which to use with the option `-http-proxy host:port`. Links also is able to deal with buggy Web servers. Several Web servers claim to be compliant with a particular HTTP version but aren’t. To compensate for this, use the `-http-bugs.*` options. For example, `-http-bugs.http10 1` forces Links to use HTTP 1.0, even when a server claims to support HTTP 1.1.

If you are looking for a strictly text replacement for the venerable Lynx, there is ELinks. ELinks supports colors, table rendering, frames, background downloading and tabbed browsing. One possibly useful option is `-anonymous 1`. This option disables local file browsing and downloads, among other things. Another interesting option is `-lookup`. When you use this, ELinks prints out all the resolved IP addresses for a given domain name.

Now that you can look at Web content from the command line, how can you interact with the Web? What I really mean is, how do you upload and download from the Web? Say you want an off-line copy of some content from the Web, so you can read it at your leisure by the lake where you don’t have Internet access. You can use curl to do that. curl can transfer data to or from a server on the Internet using HTTP, FTP, SFTP and even LDAP. It can do things like HTTP POST, SSL connections and cookies. You can specify form name/value pairs so that the Web server thinks you are submitting a form by using the option `-F name=value`. One really interesting option is the ability to use multiple URLs through ranges. For example, you can specify multiple hosts with:

```
curl http://site.{one,two,three}.com
```

which hits all three sites. You can go through alphanumeric ranges with square brackets. The command:

```
curl http://www.site.com/text[1-10].html
```

downloads the files text1.html to text10.html.

What if you want a copy of an entire site for off-line browsing? The wget tool can help here. In this case, you likely will want to use the command:

```
wget -k -r -p http://www.site.com
```

The `-r` option recurses through the site’s links starting at http://www.site.com/index.html. The `-k` option rewrites the downloaded files so that links from page to page are all relative, allowing you to navigate correctly through the downloaded pages. The `-p` option downloads all extra content on the page, such as images. This way, you can get a mirror of a site on your desktop. wget also handles proxies, cookies and HTTP authentication, along with many other conditions.

If you’re uploading content to the Web, use wput. wput pushes content up using FTP, with an interface like wget.

Now you should be able to interact with the Internet without ever having to use a graphical interface—yet another reason to keep you on the command line. —JOEY BERNARD

ARE YOU A LONGTIME *LJ* SUBSCRIBER?

The 200th issue of *Linux Journal* is rapidly approaching, and we’d like to take this opportunity for everyone to learn a bit more about some of the people who’ve helped make *LJ* possible for so many years. If you’re a longtime subscriber, please send a message to ljeditor@linuxjournal.com by September 10, 2010, and include the following information (we reserve the right to print your responses):

- **How long you’ve been a subscriber.**
- **Why you subscribe to *LJ* and/or what you like most about *LJ*.**
- **A brief bio of yourself.**
- **A photo of yourself.**
- **Your postal address. (Your address will not be published or used for any purpose other than to send you a T-shirt if you win.)**
- **Your shirt size.**

We will randomly select ten subscribers who participate, and send the “winners” a free T-shirt.

LUG Startup Kit

I live in a very remote area, and the closest active LUG is a several-hour drive away. I figure my situation isn't unique, so while I begin to form a LUG in my area, I thought it would be nice to share some quick tips I've gathered about doing so (mostly from Kyle Rankin, a friend and president of the North Bay Linux Users Group, **www.nblug.org**). Here's my quick list of things to gather when forming a Linux Users Group:

- People: this might seem obvious, but in all the preparations, it's easy to forget that you need at least a half-dozen people or so who are willing to show up regularly.

- Regular time and place: most LUGs meet monthly. There is no rule about this, but monthly meetings seem to be a good regularity, making the LUG feel dedicated, yet occuring not so often that finding speakers (my next bullet point) becomes difficult. If possible, keep a standard meeting place as well. That way, if people miss a meeting, they don't show up at the wrong place next time.

- A reason to come: socializing is great, but having a speaker, a demonstration, a Skype interview or anything unique to gatherings is essential. Why would we leave the comfort of our La-Z-Boy recliners when we could just banter in an IRC channel? Make it worthwhile to put on pants.

- Refreshments: this might be just water, or it might be water and coffee. Perhaps you have donuts. The important thing is for people to have something to hold in their hands, especially during the socialization time of the meeting. Many of us are introverts, and standing in a room full of other introverts is difficult. Put a cup of coffee in people's hands, however, and they have something to do. They're no longer standing awkwardly; they're drinking coffee. Trust me, it helps.

Really, that's about it. If you are starting the LUG, you'll likely need to stand up and talk for a few minutes to welcome everyone and introduce your special guest/video/event. After an hour or hour and a half (try to stick to your scheduled time), you can adjourn the meeting. You're done; you started a LUG.

From there, many other options exist. Most LUGs have a Web site with information about their meetings. Some LUGers go to a bar after the meeting is over. (Many people don't drink alcohol, so don't make it part of your LUG meeting to go for drinks.) Some LUGs host installfests, hackfests or gaming parties. There really aren't any rules for what your group should do. It's a rather open concept. And, if you're in Northern Michigan any time after fall 2010, check out NOMLUG (**www.nomlug.org**). Hopefully, we'll be meeting regularly by then!

—**SHAWN POWERS**

They Said It

Their management made some very bad decisions that damaged their business and allowed us to buy them for a bargain price....The underlying engineering teams are so good, but the direction they got was so astonishingly bad that even they couldn't succeed. Really great blogs do not take the place of great microprocessors. Great blogs do not replace great software. Lots and lots of blogs does [sic] not replace lots and lots of sales.
—Larry Ellison talking about Sun Microsystems

Those days are dead and gone and the eulogy was delivered by Perl.
—Rob Pike in a Slashdot interview responding to a question related to having one tool do one job well.

Eternity is a very long time, especially towards the end.
—Woody Allen

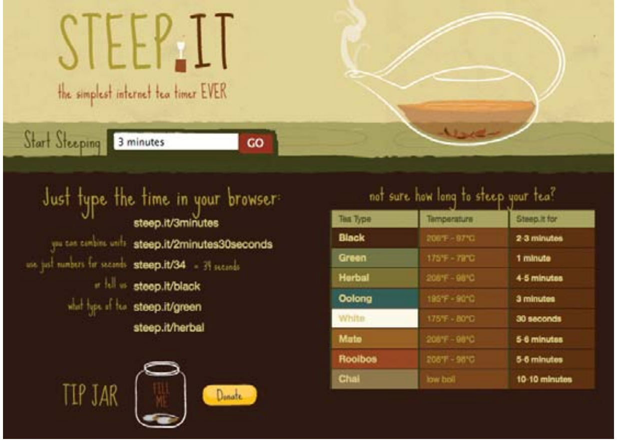
Computers are magnificent tools for the realization of our dreams, but no machine can replace the human spark of spirit, compassion, love, and understanding.
—Louis Gerstner, CEO, IBM

Get your feet off my desk, get out of here, you stink, and we're not going to buy your product.
—Joe Keenan, President of Atari, in 1976 responding to Steve Jobs' offer to sell him rights to the new personal computer he and Steve Wozniak developed.

To be a nemesis, you have to actively try to destroy something, don't you? Really, I'm not out to destroy Microsoft. That will just be a completely unintentional side effect.
—Linus Torvalds

Windows is just DOS in drag.
—Anonymous

Cloudy Tech Tips



STEEP.IT

the simplest internet tea timer EVER

Start Steeping 3 minutes GO

Just type the time in your browser:

steep.it/3minutes

you can combine units steep.it/2minutes30seconds

use just numbers for seconds steep.it/34 = 34 seconds

or tell us steep.it/black

what type of tea steep.it/green

steep.it/herbal

not sure how long to steep your tea?

Tea Type	Temperature	Steep.it for
Black	206°F - 97°C	2-3 minutes
Green	175°F - 79°C	1 minute
Herbal	206°F - 96°C	4-5 minutes
Oolong	195°F - 90°C	3 minutes
White	175°F - 80°C	30 seconds
Mate	206°F - 96°C	5-6 minutes
Rooibos	206°F - 96°C	5-6 minutes
Chai	low boil	10-10 minutes

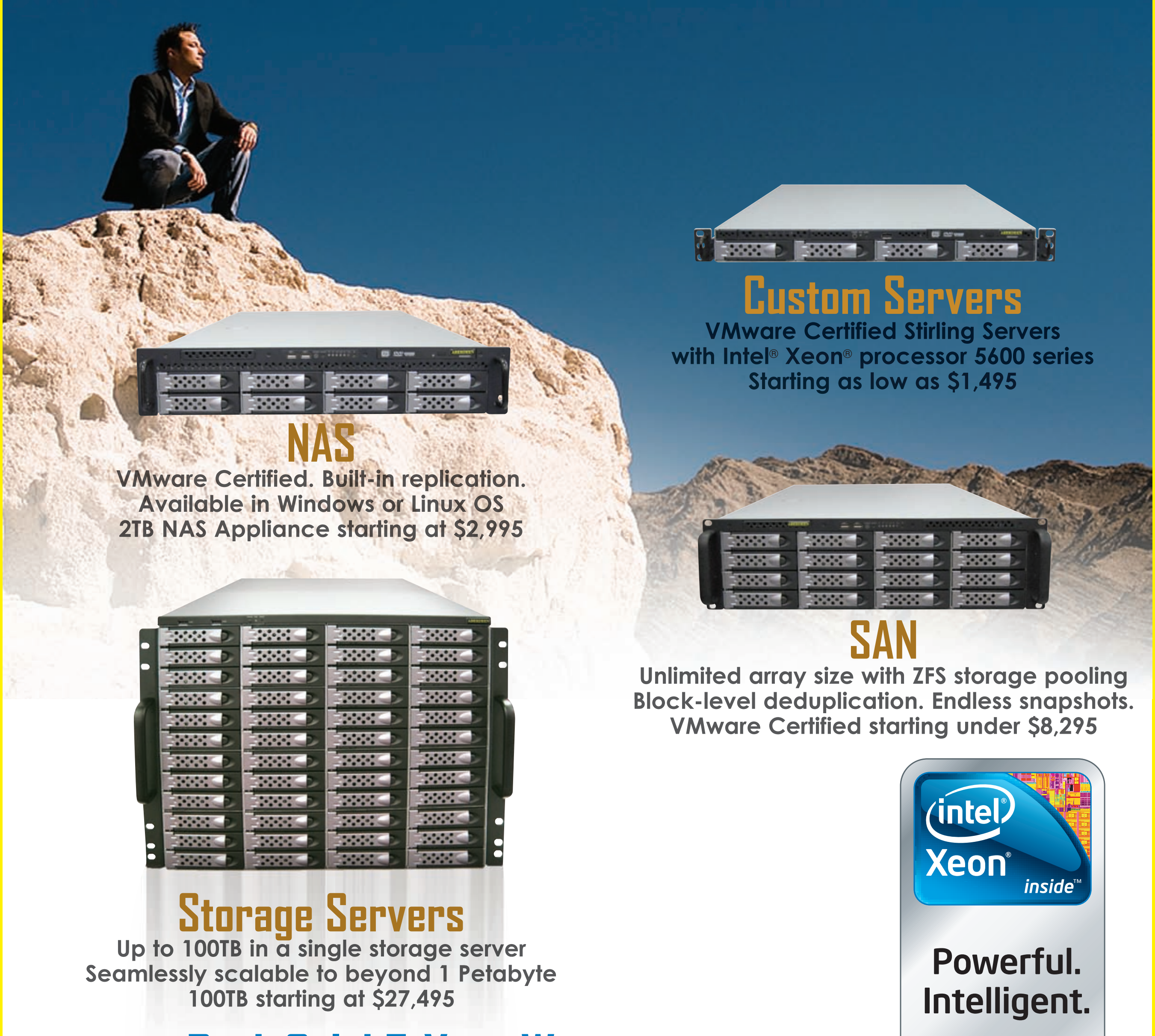
One of the things I love about the Linux community is how willing to share knowledge everyone tends to be. Whether it's a software suggestion, hardware review or just a quick tech tip, we love to share. Most of the tech tips we receive here at *Linux Journal* are command-line tips. Those are great, but what about all those Web tools you might use? Just because a Web-based tip might work for more than only Linux doesn't mean it's not a great tip.

For example, I love the STEEP.IT Web site. Like the old tea-timer applications we have used in the past, this simple Web-based application helps me get perfect green tea instead of over-steeped yuckiness. Just visit **steep.it/green**, and the counter starts for a perfect cup of green tea.

Do you have any handy tech tips you'd like to share? They can be Web-based, GUI-based, command-line-based, or even just tips like, "don't eat yellow snow". Send your tech tips to techtips@linuxjournal.com, and if we print your submission in the magazine, we'll send you a free T-shirt! Be warned, however, it's very unlikely we'll print any tips about eating colored snow.

—SHAWN POWERS

Our very own Shawn Powers models our favorite T-shirt.



Redis

If you need a high-speed storage or caching system that provides everything memcached does and then some, take a look at Redis.

REUVEN M. LERNER

The past few months, I've been covering non-relational databases, sometimes known as NoSQL databases. To hard-core NoSQL proponents, relational databases are no longer the be-all and end-all of data storage. Rather, NoSQL systems, which offer flexibility, easy replication and storage using modern data structures, are the way of the future—and perhaps even of the present.

Most NoSQL adherents aren't quite this extreme, but instead point to NoSQL as a useful solution to relatively new problems, such as those faced by Web sites with massive loads. To them (and me), NoSQL databases offer the storage equivalent of a new data structure. You could build programs with nothing more than integers, strings and arrays, but with the addition of hash tables to your arsenal, your code becomes easier to write and maintain. In the same way, having an additional storage mechanism can improve the quality, efficiency and maintainability of your software.

NoSQL is a catchphrase that has caught on like wildfire in the past year or two, but it's a problematic phrase in that it describes what these databases are not, rather than what they are. And indeed, many different types of NoSQL databases exist. Two that I have explored in this column during the past few months are MongoDB and CouchDB. Both of these are "document" databases—they store collections of name-value pairs, much like a Ruby hash or a Python dictionary.

A different type of NoSQL database is the key-value store. Whereas you can think of a document database as containing multiple hash tables, a key-value store is the equivalent of a single hash table. As you can tell by its name, a key-value store allows for the storage of a single value (which might be an aggregate data structure, such as an array or hash table), identified by a single key.

Whether a document database or a key-value store is more appropriate for your application depends greatly on your needs. I recently rewrote part of my PhD dissertation software, which previously had used PostgreSQL for all back-end storage, to use a combination of PostgreSQL and MongoDB. I chose MongoDB because I will need to retrieve documents using a variety of fields and combinations of fields. A single key for each document would have been insufficient.

In another case, a financial application on which I have been working, I needed fast access to the latest exchange rates for a number of currency pairs. Because I was going to be retrieving data based only on a single, unique key (that is, the six-letter representation of a currency pair), using a document database would result in unnecessary overhead. All I was interested in doing was storing the current exchange rate for a currency pair or retrieving the current rate for that pair, a perfect match for a key-value store.

So, I spent some time investigating key-value stores and decided to use Redis, an open-source key-value store originally developed by Salvatore Sanfilippo, an Italian programmer who was hired by VMware to work on Redis full-time. Redis was released in February 2009, but it quickly has attracted a large following, in no small part because of its amazing speed.

In many ways, Redis resembles memcached, another key-value store that is popular for scaling Web applications. Like memcached, Redis stores keys and values in RAM. Like memcached, Redis is extremely fast. Like memcached, Redis has bindings and clients written in a large number of languages.

However, there are significant differences. Redis can store and manipulate a large number of data structures (such as lists, sets and hashes). Redis stores values in RAM but writes them out to disk, asynchronously, on a regular basis. This means if someone pulls the plug on your computer, you will lose only the items you added since the last time Redis saved. Everything else will be read into RAM and made available in the usual way when you next bring up Redis.

And, have I mentioned that Redis is fast? It's not uncommon to hear people talk about getting tens of thousands of reads and writes per second with Redis.

Downloading and Installing

Now that I have described Redis, let's try to install it. On most modern Linux distributions, you should be able to install Redis (often as the package redis-server) via apt-get or yum. However, pay attention to the version number. My Linux server running Ubuntu 9.10 happily installed a very old version of Redis for me. I uninstalled it and downloaded it from the Redis home page (see Resources).

If you download the source code, you might be surprised to discover that there is no configure script. Rather, you just run `make` to compile Redis. Once it's done, you can install the programs (especially redis-server) manually into an appropriate directory, such as /usr/local/bin. Don't forget to install redis.conf, the Redis configuration file, in an appropriate place, such as /etc. To get things started, say:

```
/usr/local/bin/redis-server /etc/redis.conf
```

This tells Redis to start up and read its configuration from /etc/redis.conf. The configuration file is easy to read and modify, and you should take a look at it when you have a chance. If you're interested in just starting to work with Redis and don't care about fiddling with the controls, you can do that. The default configuration works just fine for most basic purposes.

The configuration setting that probably is of greatest interest is "daemonize", which indicates whether Redis should put itself into the background. I kept Redis in the foreground (and with debug-level logging active) when I first started to use it, but when I finally put it into production, I turned on daemonize, so I wouldn't receive a large number of log and update messages while the system was in use.

The other configuration setting indicates how often Redis should save its state to disk. The default configuration parameters that came with my installation look like this:

```
save 900 1
save 300 10
save 60 10000
```

This means Redis should save its state every 900 seconds if there has been one change, every 300 seconds if there have been ten changes, and every 60 seconds if there have been 10,000 changes. Redis saves to disk asynchronously, so there's no danger of it slowing down substantially when it performs the save operation.

You can change these settings according to your particular application's needs, striking an appropriate balance between how much data you're willing to lose if the server goes down and the need for high performance. A separate program, redis-benchmark, comes with Redis, and it allows you to get a sense of how many reads and writes you can expect to execute per second on your specific hardware, with the configuration options you have put in place.

By default, Redis listens on port 6379. You can connect to it locally via telnet or by using the redis-cli program that comes along with it, which lets you interact with the Redis server.

Working with Redis

Now that you have a Redis server, how do you work with it? One simple way is to use the command-line interface, which comes as a program called redis-cli. If you prefer, you can use a programming language instead, which hides the protocol behind a set of objects and methods, but most of the libraries I have seen use the same method names as the underlying Redis protocol.

The two most basic commands in Redis are GET and SET, which retrieve and set values. SET takes two parameters, a key and a value, while GET takes a single parameter:

```
redis> GET name
(nil)
redis> SET name reuven
OK
redis> GET name
reuven
redis> GET Name
(nil)
```

From this example, you can see several things. First, Redis will return a nil value if you retrieve a key that has not been set. Second, keys are case-sensitive, so "name" is different from "Name". This might be important if you use names or e-mail addresses as the keys in your Redis database, so be careful! Finally, you can see that Redis stores everything as a string, at least when you're storing things in this way, so you don't need to put quotes around your values, unless they contain quotes.

The nature of the protocol means that your keys may not contain space characters. I read somewhere that this restriction may be lifted at some point. Nevertheless, for compatibility with older versions of Redis, you might want to remain conservative in your key-naming conventions. Other than that, you are free to use any character you want in your keys and values.

Additional Features

If this were all Redis could do, you might think of it as a super-memcached that saves its state to disk on a regular basis. After all, memcached also is a key-value store that keeps data in RAM and is extremely fast.

However, Redis offers a number of features on the server that go beyond what memcached offers. For example, there is the setnx command, which sets a new value for a particular key, but only if the value does not yet exist. In other words, this is a test-and-set feature, allowing you to be confident you are not overwriting existing, and important, data. For example:

```
redis> setnx name Kermit
(integer) 0
redis> get name
reuven
```

You also can ask Redis to increment and decrement counters for you. For example:

```
redis> set counter 10
OK
redis> incr counter
(integer) 11
redis> incr counter
(integer) 12
redis> decr counter
(integer) 11
redis> decr counter
(integer) 10
```

Redis also provides a rich set to begin with; it allows you to store and manipulate lists. Lists are stored and retrieved using a separate set of commands from GET and SET, which confused me when I first began to use it, but it has become somewhat more natural over time. You can create, add members to and remove members from a list with a few simple commands:

```
redis> lpush atflist first
OK
redis> lpush atflist next
OK
redis> rpush atflist last
OK
redis> lrange atflist 0 -1
1. next
2. first
3. last
redis> lindex atflist 2
last
```

The lpush and rpush commands add an element to a list (or create the list, if it doesn't yet exist) on the left or right side, respectively. They are similar to the lpop and rpop commands, which remove an element from the stated side of the list. The lrange command allows you to list all the elements of the list from a particular index and until another index. If you give -1 as the ending index, you get the entire list returned to you. Finally, you can retrieve the element at a particular index with lindex.

Although it might not be obvious at this point, Redis is strictly typed. This means trying to retrieve a list as a scalar value, or vice versa, will result in an error:

```
redis> get atflist
(error) ERR Operation against a key holding the wrong kind of value
redis> lpush name lerner
(error) ERR Operation against a key holding the wrong kind of value
```

Thus, it's important, when working with Redis, to remember what the type is of each key-value pair.

Related to lists, but with a distinct purpose, are sets. You add items to a set with `sadd`, get a list of members with `smembers` and find the length ("cardinality") of the set with `scard`:

```
redis> sadd children atara
(integer) 1
redis> smembers children
1. atara
redis> sadd children shikma
(integer) 1
redis> sadd children amotz
(integer) 1
redis> smembers children
1. amotz
2. shikma
3. atara
redis> sadd children amotz
(integer) 0
redis> scard children
(integer) 3
```

As you can see from the above example, adding an element to a set normally results in a response of 1, indicating that the element was added. However, each element of a set must be unique within the set; no duplication is allowed. If you try to re-add an element that already exists in the set, Redis responses with 0, indicating that the element did not need to be added. As with all other parts of Redis, sets are case-sensitive, so if you try to add the same name, but with a different capitalization, the operation will succeed.

Redis provides facilities for working with sets, such as union and intersection. One possible use for this would be in social tags on a Web site. Each URL could be the name of a set, and the set could contain all the social tags applied to that URL. You then could find which tags have been applied to two different URLs without having to retrieve and compute this on your own, at the application level.

Redis also provides sorted sets, which are identical to the sets you have seen until now, but they keep the items in a specific order (or "rank") that can be modified.

The most recent versions of Redis now support hash tables. (By the time you read this, Redis 2.0 likely will have been released, with complete support for such functionality.) This might seem a bit strange, given that you can think of Redis as a large hash table, but it means you can store multiple hash tables within Redis. The hash-table functions all begin with an h and provide the same sorts of setting, getting and testing functionality that you have seen for the main Redis storage mechanism.

Finally, the latest version also provides "multi-exec" functionality, allowing you to execute multiple commands within a single atomic operation. This is not quite the same as transactions as you know them from relational databases, but it goes a long way toward such functionality, making Redis attractive not only for basic key-value operations, but also for more complex ones.

Is Redis Good for Everyone?

I looked at Redis after having read numerous rave reviews, and I was expecting to find serious problems with it. To date, I haven't found any. Indeed, I find myself among its excited proponents. That said, I have grown to enjoy working with Redis because I'm using it in places where it is appropriate. I can handle the loss of data stored since the most recent checkpoint. The data I am storing fits into Redis' data structures quite easily, and the data I am storing fits within my server's available RAM. In addition, there is an excellent

Ruby library for working with Redis, which allows me to integrate it into my work seamlessly and easily.

That said, Redis isn’t a good match for everyone. If you are storing multilevel hash tables, or if you cannot afford to lose even a moment’s data when the server goes down, or if you want to have the data replicated across master servers (as opposed to master-slave, which Redis handles easily), you might want to look at a different solution, such as Cassandra. But I have been impressed and delighted with Redis in my work so far, and from what I can tell, I’m not the only one who feels this way.

Conclusions

If you need a high-speed storage or caching system that provides everything memcached does and then some, you probably should take a look at Redis. It is easy to install, high performance, and it has client libraries in every major programming language. Redis has been in production use with numerous applications, including many Web sites, for more than a year, and its users continue to rave about its functionality and performance. Even if you don’t need a key-value store right now, it might be worth installing and playing with Redis. I wouldn’t be surprised if after a few minutes of experimentation, you will think of some uses for it you hadn’t considered previously.■

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 home page for Redis, as well as the Web site from which you can download the latest source code, is **code.google.com/p/redis**. This page contains a large number of links to tutorials and libraries for Redis users, most of which are worth at least a quick look.

A good introduction to Redis by Kirk Hanes of Engine Yard (a Ruby hosting company) and how you can use it from within your Ruby programs is at **www.engineyard.com/blog/2009/key-value-stores-for-ruby-part-4-to-redis-or-not-to-redis**.

Finally, a helpful cheat sheet for the Redis protocol, including the latest additions, such as hash tables and multi-exec, is available from Mason Jones at his GitHub page: **github.com/masonoise/redis-cheatsheet**. (Appropriate, given that GitHub is a heavy user of Redis.)

DAVE TAYLOR

Understanding Exit Codes

In the real world, things don't always work out as you hoped. That's what exit codes are for—letting you know when things went wrong.

Last month, we looked at signals, the rudimentary mechanism that processes on a Linux box can use to communicate events and state changes. We talked about how each of the signals can be sent manually to a running process with the kill command, and how shell scripts then can catch and respond to specific signals (though not all of them—some cannot be caught because they're actually handled by the operating system itself).

Analogous to signals, exit codes turn out to be an easy way for processes to communicate state back to the calling parent process, in a way that most Linux users just ignore. Not anymore—this month, we're going to take a closer look.

mv Exit Codes

Let's start with a simple Linux command that everyone's probably already mastered: mv, which moves a file or directory from one spot in the filesystem to another (and/or renames it).

As you know, you can generate errors if the target is missing, destination is missing and so on. Here's quick example:

```
$ mv missing ~/missing2
mv: cannot move `missing' to `.../missing2': No such file or directory
```

You see an error; obviously, it didn't work. Ah, but behind the scenes, a numeric "return code" variable has been set in the shell too, something you can test and respond to within a shell script. Check out this sequence:

```
$ echo $?
0
$ mv missing ~/missing2
mv: cannot move `missing' to `.../missing2': No such file or directory
$ echo $?
1
$ echo test me
test me
$ echo $?
0
```

If no error occurs when executing a command, the exit code (which we reference in the shell with the shorthand $?) will have the value of 0: no error. Now, if I run a command that fails, the exit code will have a nonzero value. In the case of the failing mv command above, the error code will have the value of 1. And, if I now run yet another command, one which runs without error, the error code is reset to zero.

Now, let's take a peek at the mv man page, paying particular attention to the latter part of the doc. Close examination reveals: "The mv utility exits 0 on success, and >0 if an error occurs."

That's not so interesting, really. The grep command has more interesting diagnostics, actually: "Normally, exit status is 0 if selected lines are found and 1 otherwise. But the exit status is 2 if an error occurred, unless the -q or --quiet or --silent option is used and a selected line is found."

There is a set of system exit codes that are defined, although it's possible you'll never need the information. Here's a list of the codes and their meanings:

- 1: general errors
- 2: misuse of shell builtins (pretty rare)
- 126: cannot invoke requested command
- 127: command not found error
- 128: invalid argument to "exit"
- 128+n: fatal error signal "n" (for example, kill -9 = 137).
- 130: script terminated by Ctrl-C

I'd never actually seen this list until I started digging into the issue of exit codes, so you can continue on your merry shell-scripting path safely without worrying about the data above.

Utilizing Exit Codes

The most common situation in which you analyze and respond to an exit code is in error handling in a script.

Here's a simple snippet where you want to create a directory. If it fails, you want to output an error message and quit:

```
#!/bin/sh
mkdir /usr
```

```
echo \$? = $?
if [ $? -ne 0 ] ; then
  echo "mkdir /usr failed: we have an exit code of $?"
  exit 1
fi
echo "made the requested directory. Why is '/' world writable?"
exit 0
```

It turns out, there's a nuance to working with the $? that I've already alluded to—one that makes output statements like the first "echo" quite problematic. You can see why in the output:

```
$ ./test.sh
mkdir: /usr: File exists
$? = 1
made the requested directory. Why is '/' world writable?
```

Can you see what happened? The exit code = 1 immediately after the mkdir, which makes sense as /usr already exists, but when we again test the exit code in the conditional, it's *not* a nonzero value!

Why? Because at that point, it indicates the exit code of the echo statement, not the mkdir command. Oops.

You can verify this simply by commenting out the first echo statement, in which case you now see this as the command output:

```
$ !.
./test.sh
mkdir: /usr: File exists
mkdir /usr failed: we have an exit code of 0
```

That makes more sense, doesn't it?

Because this can be tricky, a common thing I see in really bulletproof shell scripts with lots of error handling is something like this:

```
#!/bin/sh
mkdir /usr
error=$?
if [ $error -ne 0 ] ; then
  echo "mkdir /usr failed: we have an exit code of $error"
  exit 1
fi
```

This is one instance where a local variable to hold a system or global variable makes a lot of sense, and it also lets you do things like have an error message show up on-screen *and* be handed off to something like syslog() to ensure that the admin sees it at some point.

Of course, error handling doesn't always just need to print a message and exit. Another scenario might be the following:

```
alternates='
    http://www.example.com/test.pdf
    http://www.example2.com/test.pdf
    http://www.example3.com/test.pdf
    '
gotit=0
for file in $alternates
do
  wget $file
  if [ $? -ne 0 ]; then
    echo "Unable to get $file
  else
    gotit=1
    break
  fi
done
...
```

Here, we try to retrieve a file from one of multiple alternate locations and exit the loop only when we succeed (or when we've run out of possibilities).

Hiding Error Messages

Now that you know how to capture and analyze the exit codes from system commands, what if you want to have the error message be one from your script, not one from the command itself?

That's done with another new shorthand notation: >&, which redirects the stderr/error output stream. Here's how I use that to hide all error messages from the mkdir command being used in our sample scripts:

```
mkdir /usr >& /dev/null
```

You also can use &> or 2>&1 instead of >&.

If you don't test the results of the command, of course, you seriously can hose things up, but this makes the output more elegant for sure:

```
$ ./test.sh
mkdir /usr failed: we have an exit code of 0
```

Hmmm...I'm still getting that false 0. Oh! I haven't added the code to save the exit code value as "error". One slight tweak later and:

```
$ ./test.sh
mkdir /usr failed: we have an exit code of 1
```

That's more like it!

I'm going to call this a wrap for this month. Next month, I'll demonstrate how the exit command lets you send exit codes back to the calling program from procedures and functions, just as if they were separate Linux commands rather than part of the same shell script.■

Dave Taylor has been hacking shell scripts for a really long time, 30 years. He's the author of the popular *Wicked Cool Shell Scripts*, and he can be found on Twitter as @DaveTaylor and more generally at www.DaveTaylorOnline.com.

MICK BAUER

Building a Transparent Firewall with Linux, Part II

Use commodity hardware to build a transparent firewall.

Last month, I kicked off a series of articles on transparent firewalls, beginning with a brief essay on why firewalls are still relevant in an age of Web applications and tunneled traffic. I also explained the difference between a standard, routing firewall and a transparent, bridging firewall.

This month, I begin discussing actually building a transparent firewall. Making a firewall invisible to the network is cool already, but to spice things up even further, I'm going to show how to build a transparent firewall using OpenWrt running on a cheap broadband router. Let's get started!

Possible Topologies

I want to dive right into it, so I'm not going to review very much from last time. Suffice it to say for now that whereas a normal "routing" firewall acts as an IP gateway between the networks it interconnects, a "bridging" firewall acts more like a switch—nothing on either side of the firewall needs to define the firewall explicitly as a route to whatever's on the other side.

One important ramification of this is that with a routing firewall, the networks connected to each firewall interface need to be on different IP subnets. This means if you insert a firewall between different networks, those networks must usually at least be re-subnetted, if not re-IP-addressed altogether.

In contrast, the bridging firewall we're going to build in this series of articles won't require anything on your network to be reconfigured. At worst, you'll need to re-cable things to place the firewall in a "choke point" between the parts of your network you want to isolate from each other.

Suppose you want to use the transparent firewall on a home network to protect it from Internet-based attackers. In that case, you may want only two firewall zones, such as "outside" (the Internet) and "inside" (your home network). Most home users, it's safe to say, connect everything in their network directly to their DSL or cable modem via some flavor of 802.11 Wireless LAN (WLAN), with maybe one or two things connected to Ethernet interfaces on the same modem. Figure 1 shows a typical home network of that type.

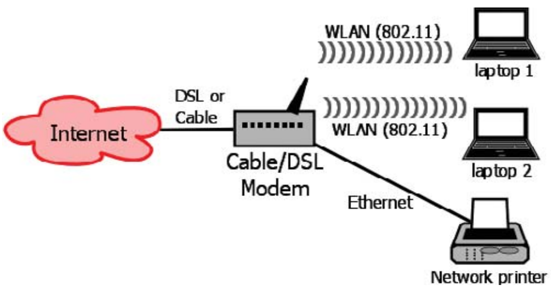


Figure 1. Typical Home Network

If you're such a user, the first step in deploying a transparent firewall is to move everything off the DSL/cable modem (except, of course, the actual DSL or cable connection) and onto either the transparent firewall (if it has enough interfaces), an Ethernet switch (if you don't need WLAN), a "broadband router" (a WLAN access point with built-in Ethernet switch), or onto some combination of those things.

Step two, of course, is placing the transparent firewall between the DSL/cable modem and whatever device (or devices) to which you connected the rest of your network. Despite the list of options in the previous paragraph, there really are only two approaches to this: connecting all the devices in your network to the transparent firewall, which may be perfectly feasible if your firewall has enough interfaces and your network is small enough, or collapsing them back to one or more other network devices that are, in turn, connected to the firewall.

Figure 2 shows the latter approach. In Figure 2, the two wireless laptops and the wired network printer connect to a broadband router, whose "Internet" Ethernet interface is cabled to the "inside" interface of a transparent firewall. The firewall's "outside" interface is cabled to the Ethernet interface of a DSL or cable modem.

(If I was writing this in the 1990s, at this point, I would have to explain crossover cables. But in the modern era, in which pretty much all Ethernet hardware automatically detects "crossed-over" versus "straight-through" connections, all you should need are ordinary patch cables. If you *did* need crossover cables, however, they would be the two cables in

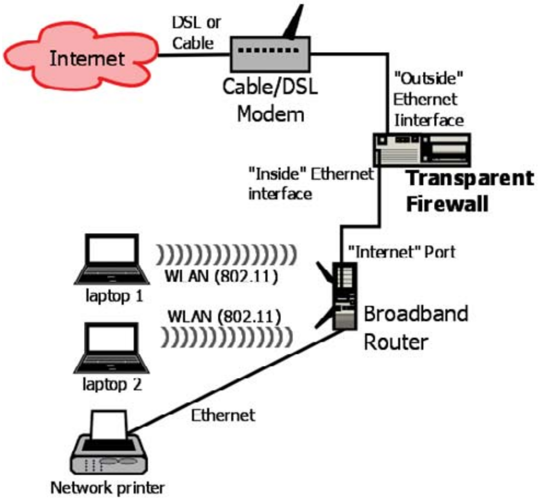


Figure 2. Home Network, Re-Cabled with Firewall

Figure 2 connected to the firewall.)

Even though I'm about to explain why and how I'm using a Linksys WRT54GL broadband router as my transparent firewall platform, which boasts five Ethernet ports plus 802.11g WLAN, for simplicity's sake, I'm going to assume you're using a separate network device like the broadband router in Figure 2, at least for the time being. Although I reserve the right to cover other topologies in later installments of this series, the immediate task will be to build a simple two-interface firewall. (Why? Mainly because it will take too much space to explain how to set up wireless networking on the firewall.)

So, what will our two-port transparent firewall do? Mainly, it will protect the internal network from arbitrary connections from the outside world. In our test scenario of "basic home user", there are no Web servers, SMTP relays or other "bastion hosts". (As with WLAN-on-the-firewall, I may cover adding an "Internet DMZ" zone later on in this series.) The firewall will allow most transactions originating *from* the internal network, with a few exceptions.

First and arguably most important, we're going to configure the firewall to know the IP addresses of our ISP's DNS servers and allow only outbound DNS queries to them. This will protect us against "DNS redirect" attacks (though not *highly* localized attacks that redirect DNS to some other internal system, such as one where a WLAN-connected attacker's evil DNS server is sitting next to the attacker in a van outside your house).

Second, we'll enforce the use of a local Web proxy, such as the one I walked through building in my four-part series "Building a Secure Squid Web Proxy" in the April, May, July and August 2009 issues of *Linux Journal* (see Resources). In other words, our firewall policy will allow Web transactions to the outside world only if they originate from the IP address of our Web proxy. This will allow us to enforce blacklists against prohibited or known dangerous sites, and also to block the activity of any non-proxy-aware malware that may end up infiltrating our internal network.

Finally, we'll restrict outbound SMTP e-mail traffic to our ISP's SMTP servers, blocking any SMTP destined elsewhere. This also will provide a small hedge against malware activity.

Why not, you may wonder, allow all internally originated traffic through for simplicity's sake? That is a valid option and a fairly popular one at that. But, it contradicts Ranum's dictum: that which has not been expressly permitted is denied. Put another way, assume that the unexpected is also undesirable.

There's some simple math behind this dictum. Bad traffic can take an infinite range of different forms. "Known-good" traffic, for most organizations, tends to constitute a manageably short list. If you allow only the transactions you expect, and if you've done your homework on identifying and predicting everything you *should* expect, then other transactions are unnecessary, evil or both.

And, what on the inside, which is supposedly "trusted", could cause unexpected transactions?

Making a firewall invisible to the network is cool already, but to spice things up even further, I'm going to show how to build a transparent firewall using OpenWrt running on a cheap broadband router.

Statistically speaking, probably malware—worms, trojans and viruses. Worms propagate themselves across networks, so by definition, they create lots of traffic. Trojans and viruses don't propagate themselves, but after they make landfall on a victim system (typically from an e-mail attachment, hostile Web site or by being hidden in some other application the user's been tricked into installing), they typically "phone home" in order to allow the malware's author to control the infected system from afar.

Traditionally, botnet agents used for spam propagation and Distributed Denial of Service (DDoS) attacks use the IRC protocol for command and control functions. That alone is a good reason to block all outbound IRC, but because IRC can use practically any TCP or UDP port, it isn't good enough to block TCP/UDP ports 194, 529 and 994 (its "assigned" ports). Besides, the malware could just as easily use some non-IRC protocol, again over completely arbitrary ports.

What if malware authors are clever enough to anticipate possible firewall restrictions, such that their code checks infected systems' local SMTP and Web-proxy settings? You still could block that malware if it tries to initiate Web transactions with some "known-evil" site on your Web proxy's blacklist. Regardless, security is never absolute. Good security involves taking reasonable measures to maximize the amount of effort attackers have to expend in order to succeed. Sadly, attackers will *always* succeed with enough effort, inside information and luck. (The good news is most attackers are opportunistic and lazy!)

Our firewall, therefore, won't allow us to be lazy about keeping our internal systems fully patched, educating our users against installing software from untrusted sources or visiting potentially nasty Web sites and so forth. But it *will* provide an important layer in our "security onion" that will make our network a less obvious target to attackers doing mass port scans against our ISP, and it will make it harder for any weirdness that *does* slip through to connect back out.

The last thing I'm going to say for now about our firewall design is that we won't have to worry about Network Address Translation (NAT) or DHCP. This, in fact, is one of the benefits of a transparent firewall! Whatever was providing NAT and DHCP services before (probably the DSL or cable modem, in our home-use scenario) can continue to do so, and if we place our firewall correctly, NAT and DHCP should continue working exactly the same as before.

Why not, you may wonder, allow all internally originated traffic through for simplicity's sake?

Hardware Considerations

Now that you understand how this setup will look, before and after firewalling, let's talk about firewall hardware. This article series isn't the first time I've tinkered with transparent Linux firewalls. Years ago when I started researching passive network monitoring, I set up several "white-box" PCs that each had multiple network interfaces and could monitor and restrict network traffic transparently.

When I began researching this new series, my first thought was to resuscitate one of those old systems or build a new one. That seemed like a waste of electricity, however. Why deal with case and CPU fans, hard drives and so forth, for something usually handled by optimized network appliances?

This line of thinking brought me to the idea of industrial/embedded platforms—small, diskless computers running an Atom or ARM processor. But the cost of these, especially models with multiple network interfaces, is similar to that of PCs, and I wanted to spend as little as possible.

Then it dawned on me that this is exactly what OpenWrt was designed for! In case you're unfamiliar with it, OpenWrt is a free Linux distribution designed to run on commodity WLAN gateways and broadband routers, such as Linksys' venerable WRT54G series. On the one hand, I'm not much interested in covering WLAN firewalling in this series (although once it's configured properly, a firewall with a WLAN interface can treat it just the same as any other network interface). But on the other hand, the WRT54G is basically a small computer with five network interfaces plus WLAN. Small memory and slow CPU aside, it should make an ideal Linux firewall platform.

This is how I settled on the Linksys WRT54GL wireless-G broadband router, which cost me only $58, as the test platform for my transparent Linux firewall experiments. How well does it perform and scale, and how stable is it? Time will tell. I would guess the short answer is "good enough for home use, but not quite Fortune-500-ready". Besides, it's bright blue, cheap and cool.

If this sort of hardware hacking isn't quite your cup of tea, I hope you'll stay with me through the series anyhow, because most of the real iptables magic we'll be working in building our transparent firewalling examples is applicable to *any* Linux system with multiple network interfaces.

One last note on hardware selection. As a Linux firewall platform, a laptop computer makes a nice middle ground between broadband routers and desktop PCs with respect to cost and power consumption, and you easily can add network interfaces to one via USB. Although even a used laptop will cost more than an OpenWrt-compatible broadband router, it will be able to run practically any mainstream Linux distribution, giving you access to a much wider range of software than you can run on OpenWrt.

If you opt for the laptop approach, be sure to select USB Ethernet interfaces that support USB 2.0 (which is necessary for anything approaching acceptable performance—USB 2.0 operates at 480Mbps, but USB 1.1 is only 12Mbps, and 1.0 is a tiny 1.5Mbps!) and, of course, that are Linux-compatible!

I've had good luck with the D-Link DUB-E100, a USB 2.0, Fast Ethernet (100Mbps) interface. It's supported under Linux by the usbnet and asix kernel modules. (My Ubuntu system automatically detects my DUB-E100 interfaces and loads both modules.)

Installing OpenWrt on a Linksys WRT54GL

Back to my OpenWrt adventure, indulge me for a few more paragraphs (plus a few more next month) before we tackle firewall configuration proper. The first step in choosing hardware to use with OpenWrt is consulting the OpenWrt Web site to see what's supported by current versions of OpenWrt (see Resources).

If you choose a Linksys device, which probably is a good choice given that the OpenWrt Project began around the Linksys WRT54G product line, be sure to choose a model whose number ends in L, which indicates "Linux-compatible". As I mentioned earlier, I chose the Linksys WRT54GL, still available at the time of this writing from various on-line retailers.

The OpenWrt Table of Hardware provides links to other OpenWrt pages giving detailed instructions on installing and configuring OpenWrt on each supported device. In the case of my Linksys WRT54GL, I followed these steps to install OpenWrt:

1. I downloaded the image file openwrt-wrt54g-squashfs.bin from the OpenWrt Web site (**downloads.openwrt.org/kamikaze/8.09.2/brcm-2.4**).

2. I powered on the WRT54GL with its factory-installed firmware.

3. I connected to the WRT54GL by typing the URL http://192.168.1.1/Upgrade.asp in the browser of a laptop connected to one of the WRT54GL's Ethernet ports, not its "Internet" port. Note that my laptop's network interface was configured to use DHCP and actually pulled its IP address via DHCP from the WRT54GL. Hence, it was assigned an IP in the subnet 192.168.1.0/24, which the WRT54GL uses by default.

4. I "upgraded" the WRT54GL's firmware with the file openwrt-54g-squashfs.bin and waited a few minutes for the upload to complete and for the WRT54GL to reboot with the new firmware.

5. Finally, from my laptop, I ran the command `telnet 192.168.1.1` to connect to the WRT54GL, and I was greeted with this message and prompt:

```
=== IMPORTANT ============================
 Use 'passwd' to set your login password
 this will disable telnet and enable SSH
------------------------------------------


BusyBox v1.11.2 (2009-12-02 06:19:32 UTC) built-in shell (ash)
Enter 'help' for a list of built-in commands.

  _______                     ________        __
 |       |.-----.-----.-----.|  |  |  |.----.|  |_
 |   -   ||  _  |  -__|     ||  |  |  ||   _||   _|
 |_______||   __|_____|__|__||________||__|  |____|
          |__| W I R E L E S S   F R E E D O M
 KAMIKAZE (8.09.2, r18961) -------------------------
  * 10 oz Vodka       Shake well with ice and strain
  * 10 oz Triple sec  mixture into 10 shot glasses.
  * 10 oz lime juice  Salute!
 ---------------------------------------------------
root@OpenWrt:/#
```

Success! Not only had I successfully turned my inexpensive broadband router into a five-Ethernet-interfaced Linux computer, I'd also learned the recipe for a refreshing cocktail, the Kamikaze. Looking around, I was pleased to discover a fairly ordinary Linux environment.

The only thing missing was a Linux 2.6 kernel. I had one more task before proceeding to turning this blue beastie into a firewall—upgrading its kernel. According to the OpenWrt Wiki, you can do so only after first installing a 2.4 kernel (which I'd just done) and changing some NVRAM settings like so via telnet:

```
root@OpenWrt:/# nvram set boot_wait=on
root@OpenWrt:/# nvram set boot_time=10
root@OpenWrt:/# nvram commit && reboot
```

This done, on my laptop, I downloaded the latest Backfire version of OpenWrt (v. 10.03 at the time of this writing) from **downloads.openwrt.org/backfire/10.03/brcm47xx**. The file I downloaded for my WRT54GL was openwrt-wrt54g-squashfs.bin.

To flash it to my WRT54GL, I opened a command window on my Windows laptop, navigated to the directory to which I'd just downloaded my new firmware image, and without pressing Enter just yet, typed the following command:

I've had good luck with the D-Link DUB-E100, a USB 2.0, Fast Ethernet (100Mbps) interface.

```
tftp -i 192.168.1.1 PUT openwrt-wrt54g-squashfs.bin
```

Before pressing Enter, I unplugged my WRT54GL, waited a few seconds, plugged it back in, and immediately pressed Enter in my Windows laptop's command window to execute that tftp command. After a few seconds, I got a "Transfer successful" message. The router decompressed the new firmware image, and it rebooted itself to Backfire. When I telneted back in to the router, I was greeted with a new banner:

```
BusyBox v1.15.3 (2010-04-06 04:08:20 CEST) built-in shell (ash)
Enter 'help' for a list of built-in commands.

  _______                     ________        __
 |       |.-----.-----.-----.|  |  |  |.----.|  |_
 |   -   ||  _  |  -__|     ||  |  |  ||   _||   _|
 |_______||   __|_____|__|__||________||__|  |____|
          |__| W I R E L E S S   F R E E D O M
 Backfire (10.03, r20728) --------------------------
  * 1/3 shot Kahlua    In a shot glass, layer Kahlua
  * 1/3 shot Bailey's  on the bottom, then Bailey's,
  * 1/3 shot Vodka     then Vodka.
 ---------------------------------------------------
root@OpenWrt:/# which tftp
```

Again, success! Now, not only is my WRT54GL broadband router running Linux, it's also running a reasonably current 2.6 kernel. I'm definitely ready to start configuring this machine for its new stealth firewall duties.

Conclusion

But, this is as far as I can take you this month. Next time, I'll begin showing how to configure networking, bridging and, of course, iptables on our transparent firewall.

If you can't wait until then, see the OpenWrt home page for more information, or if you're really adventurous, search the Web for other tutorials on creating transparent firewalls with OpenWrt. But, if I say so myself, we're off to a good start!■

Mick Bauer (darth.elmo@wiremonkeys.org) is Network Security Architect for one of the US's largest banks. He is the author of the O'Reilly book *Linux Server Security*, 2nd edition (formerly called *Building Secure Servers With Linux*), an occasional presenter at information security conferences and composer of the "Network Engineering Polka".

Resources

Home Page for the OpenWrt Project: **www.openwrt.org**

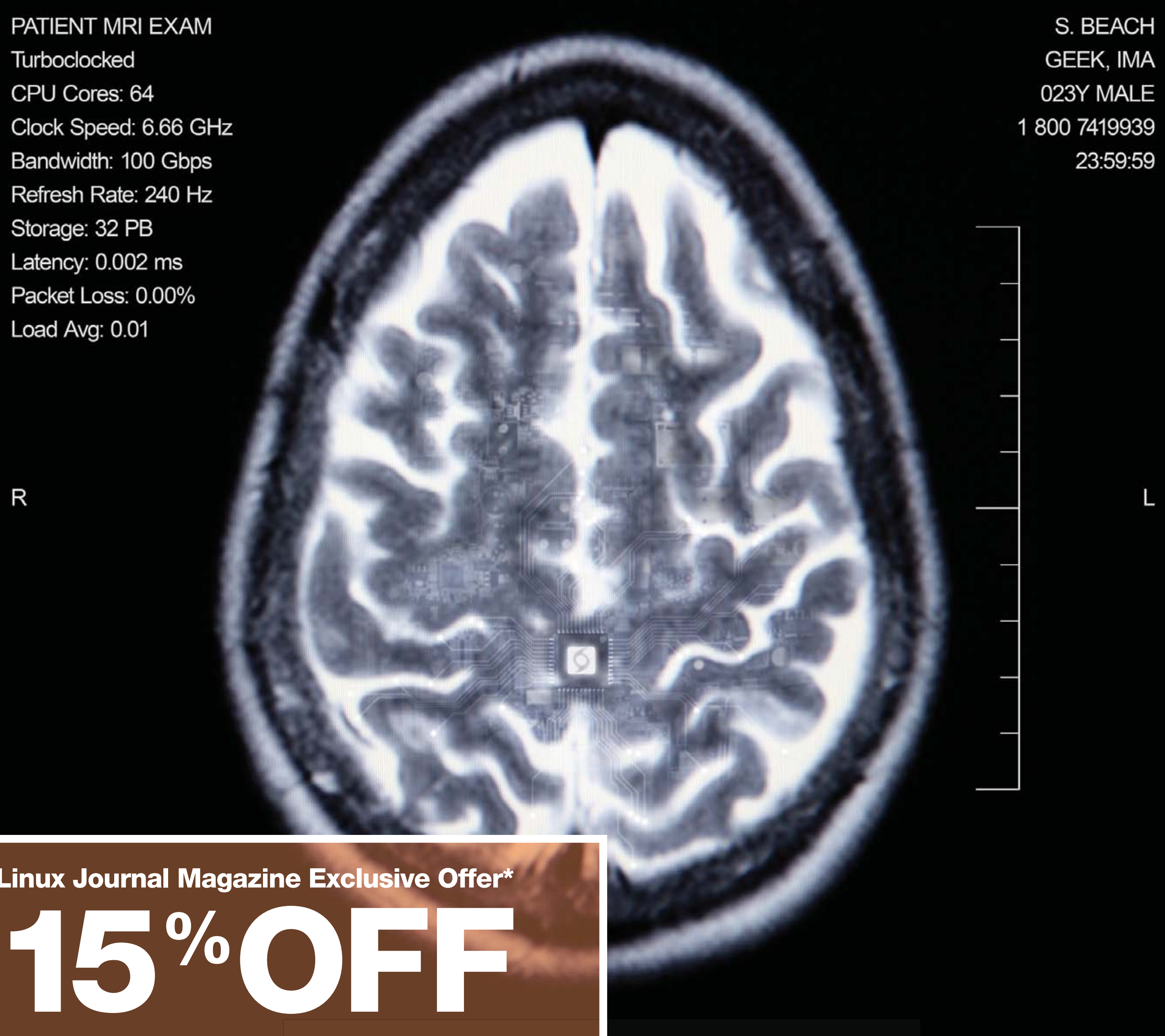
OpenWrt's Table of (Supported) Hardware: **oldwiki.openwrt.org/TableOfHardware.html**

Mick Bauer's "Building a Secure Squid Web Proxy, Part I", *LJ*, April 2009: **www.linuxjournal.com/article/10407**

Mick Bauer's "Building a Secure Squid Web Proxy, Part II", *LJ*, May 2009: **www.linuxjournal.com/article/10433**

Mick Bauer's "Building a Secure Squid Web Proxy, Part III", *LJ*, July 2009: **www.linuxjournal.com/article/10488**

Mick Bauer's "Building a Secure Squid Web Proxy, Part IV", *LJ*, August 2009: **www.linuxjournal.com/article/10513**


PATIENT MRI EXAM
Turboclocked
CPU Cores: 64
Clock Speed: 6.66 GHz
Bandwidth: 100 Gbps
Refresh Rate: 240 Hz
Storage: 32 PB
Latency: 0.002 ms
Packet Loss: 0.00%
Load Avg: 0.01
S. BEACH
GEEK, IMA
023Y MALE
1 800 7419939
23:59:59
R
L

KYLE RANKIN

Break In Your Boots

New boots, new bootloaders—both take some time to get used to. In this column, find out all the uncomfortable changes in the new GRUB2 bootloader.

A few months ago, I had to replace my favorite pair of shoes: black, suede Converse One Stars (the classic style with no white rubber toe cap, thank you). I had worn the shoes for years, but although most of the shoe held up fine, I had completely worn down the heel. Now, I'm not one to throw away a comfortable pair of shoes. This pair was on its ninth or tenth Shoe-Goo repair, but it finally became hopeless. They had to be replaced. It seemed like a simple task—after all, these shoes had been available all through my adult life, but wouldn't you know, the moment I need another pair, Converse discontinued the model and replaced it with a canvas version with the Chuck Taylor-esque white rubber toe. I had to find a new shoe.

Let me tell you, once you have found the perfect sneaker, it's impossible to find a replacement. Everything I looked at was held up to the standard of the shoe I couldn't have. After a month or two, I finally found shoes that were up to the task, and although I like them, I still miss my old shoes (oh, wouldn't you know it, now that I bought a replacement, Converse has re-released the One Stars how I like them).

I really should be used to this feeling. It seems every few years some open-source project decides to throw away an entire codebase and start from scratch. Although GNOME and KDE have stirred the pot the most with this, I've also lived through the same thing with the Enlightenment Project, the SysV init to Upstart transition, the LILO bootloader being phased out for GRUB, and now GRUB being replaced by GRUB2. For those of you who thought the difference between GRUB2 and GRUB1 was "one", you are: good at subtraction, a bit of a smart aleck and in for a rude awakening. In this article, I'm going to help you break in your new GRUB2 bootloader, so hopefully some day, it will be as comfortable to you as the original GRUB.

Why?

The first question you might ask is why we need a new bootloader at all? What is wrong with GRUB? The answer, according to the GRUB2 developers, is that the original GRUB codebase was rather old and had become unmaintainable. The software continued to get new feature requests (such as supporting new hardware and platforms) that ultimately were beyond the scope of the original code, so the decision was made to scrap it and start from scratch. Because it was a complete rewrite, the developers decided to take the opportunity to make a clean break and redesign the layout and syntax of the configuration files. Along with these changes, GRUB2 has been able to add new features, such as a rescue mode, enhanced graphical menu and splash screen support, full support for UUIDs, and support for non-x86 platforms, such as PowerPC.

Old Boots

Before I go into how GRUB2 has changed things, I'm going to give a quick overview of GRUB1 (or GRUB Legacy, as they are calling it now) to help highlight the changes for those of you who might be unfamiliar with either bootloader. GRUB (and LILO before it) has been the standard bootloader used by the majority of Linux distributions. When you boot your computer and see a menu that lets you choose between different Linux kernels, or between different versions of Linux and Windows in a dual-boot scenario, you probably are using GRUB. GRUB's job is to allow you to choose between one or more operating systems at boot time and then either load the respective kernel and initrd into memory and start the rest of the boot process, or launch the boot code for some other operating system, such as Windows.

GRUB is quite configurable and organizes itself into a few core programs and directories:

- /boot/grub/menu.lst: this is the default configuration file for GRUB, although on some distributions it is a symlink to /boot/grub/grub.conf. All of GRUB's configuration is in this file, and users edit this file directly to change any GRUB options.

- /usr/sbin/grub: this is the core GRUB binary that you can use (if you learned all of the syntax) to install GRUB onto your system. The syntax is a bit tricky though, so ultimately, other programs appeared to help automate the process.

- /usr/sbin/grub-install: this program acts as the front end to /usr/sbin/grub and makes it much simpler to install GRUB to your hard drive.

- /usr/sbin/update-grub: this script helps automate configuration of the menu.lst file. Instead of having to add new kernels to menu.lst manually, you can run this script, and it will detect kernels available on your system and build the menu.lst for you. In addition, this script can read special configuration options in the comments of menu.lst and further automate the process of providing rescue modes, memtest86+ and other customizations of the file.

Another great feature of GRUB is the fact that even with all of this configuration, if you make a mistake, GRUB allows you to change essentially any configuration option from the boot prompt. At the GRUB menu, you can press the Esc key to change boot options on the fly.

New Boots

Now that you have learned (or refreshed your memory) about GRUB, you may promptly forget it, because much of what I said above has changed. For starters (and this will drive you crazy), the default key to edit GRUB2 options at boot is Shift, not Esc. Second, the main configuration file has been changed from /boot/grub/menu.lst to /boot/grub/grub.cfg. Not only has the filename changed, but also the syntax inside the file is quite different from what you'd find in menu.lst.

While I'm on the subject of syntax changes, a crucial syntax change that GRUB2 makes is in how it numbers partitions. Where in GRUB your partitions were counted starting from zero, now the count starts with one. To make it more confusing, disk devices still are being counted starting from zero. Confused yet? In short:

- GRUB1: `/dev/sda1 = (hd0,0)`
- GRUB2: `/dev/sda1 = (hd0,1)`

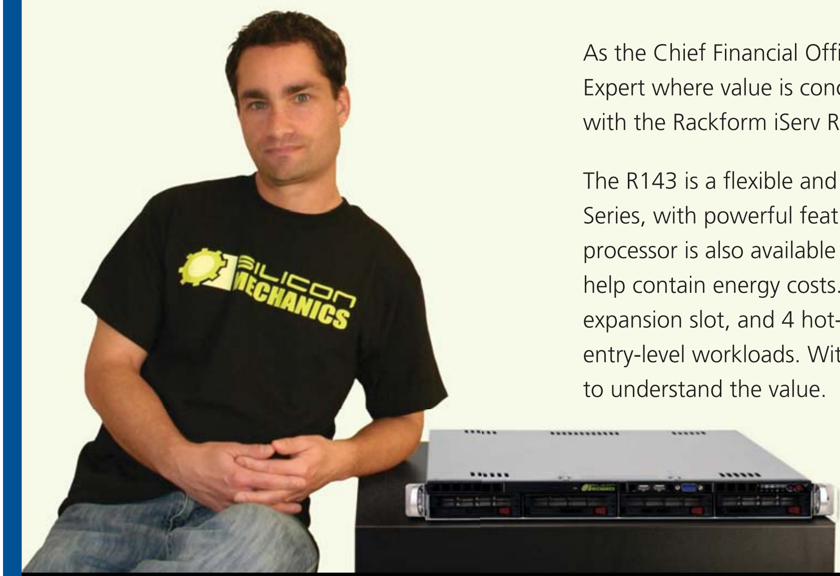
Here's a sample stanza from GRUB's menu.lst and a similar stanza from GRUB2's grub.cfg, so you can compare their syntax:

GRUB:

```
title    Ubuntu karmic (development branch), kernel 2.6.31-14-generic
uuid     c7b6836f-ac57-47ed-9e7c-b16adbf8abed
kernel   /boot/vmlinuz-2.6.31-14-generic root=UUID=
         ➥c7b6836f-ac57-47ed-9e7c-b16adbf8abed ro quiet splash
initrd   /boot/initrd.img-2.6.31-14-generic
quiet
```

GRUB2:

```
menuentry 'Ubuntu, with Linux 2.6.31-20-generic' --class ubuntu
 ➥--class gnu-linux --class gnu --class os {
        recordfail
        insmod ext2
        set root='(hd0,2)'
        search --no-floppy --fs-uuid --set
```



```
        ➥c7b6836f-ac57-47ed-9e7c-b16adbf8abed
        linux    /boot/vmlinuz-2.6.31-20-generic
        ➥root=UUID=c7b6836f-ac57-47ed-9e7c-b16adbf8abed
        ➥ro    quiet splash
        initrd   /boot/initrd.img-2.6.31-20-generic
}
```

Before you sit down and study the new syntax, I should point out that you are actively discouraged from editing grub.cfg directly. This file is generated from a series of scripts and configuration files I will talk about later, so any changes you make will be overwritten the next time any package updates that would trigger the GRUB2 update.

Like with GRUB, here are the core files and programs involved in GRUB2 configuration:

- /boot/grub/grub.cfg: this is the core GRUB2 configuration file but is not to be edited directly.

- /etc/default/grub: this is the main configuration file for end users to edit. In this file, you can configure a limited subset of GRUB2 options, such as timeouts, basic kernel boot options and whether to use a graphical console or UUIDs. Every time you make a change to this file, you must run `/usr/sbin/update-grub` for the changes to be reflected. Here are some sample lines from the file to give you some idea of the syntax:

```
GRUB_DEFAULT=0
GRUB_HIDDEN_TIMEOUT=0
GRUB_HIDDEN_TIMEOUT_QUIET=true
GRUB_TIMEOUT=5
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"
GRUB_CMDLINE_LINUX=""
```

- /etc/grub.d/: this directory contains a series of scripts that are executed in numerical order by the grub-mkconfig program and will configure different parts of grub.cfg. On a default Ubuntu Lucid install, for instance, you would find the following files:

```
$ ls /etc/grub.d/
00_header        10_linux        30_os-prober  README
05_debian_theme  20_memtest86+   40_custom
```

  The order in which configuration options appear in grub.cfg is governed by the order its script appears in this directory. So if, for instance, you wanted to have a different OS appear before the Linux options in the menu, you could name the script 01_otheros. Although the scripts that are currently there do a few complicated things, essentially your script needs to output the configuration information you want in the proper grub.cfg syntax, so I suppose it even could be a series of echo statements in a shell script. If you want to create a custom configuration script though, Ubuntu has provided the 40_custom script for you to use that will not risk being overwritten.

- /usr/sbin/grub-install: like with GRUB, the GRUB2 grub-install program is the recommended way to install GRUB2 onto a device. It calls a number of other scripts that perform various system checks, device probes and everything else that's necessary to install GRUB2 to a boot device.

- /usr/sbin/update-grub: this script still exists and is still the recommended way to update GRUB2's configuration file, but now this is a very short shell script that executes grub-mkconfig. Whenever you edit a configuration file or script, run this command with no arguments to rebuild the grub.cfg file.

- /usr/sbin/grub-mkconfig: this program does the real heavy-lifting to build your grub.cfg file. It is the program that executes the various scripts in /etc/grub.d.

In addition to the above changes, here are a few extra things that are different in GRUB2:

- GRUB2 no longer has stage 1.5 in the boot process.

- On a new Ubuntu install when no other OSes are present, GRUB2 will not display a menu at boot time and will instead boot directly into the Ubuntu install.

- To reiterate, hold Shift instead of Esc to change GRUB2 boot options.

If you are like I was when I first discovered all of these changes, right about now you are feeling like the ground has been moved out from under you. I felt much like I did when I couldn't buy a new pair of black suede One Stars. All I can hope for is that over time, like my new shoes, the uncomfortable parts of GRUB2 will break in, and I will feel comfortable with them and maybe some day even like them as much as GRUB1.■

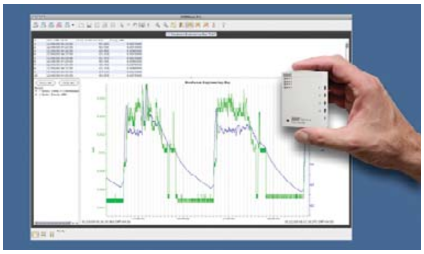
Kyle Rankin is a Systems Architect in the San Francisco Bay Area and the author of a number of books, including *The Official Ubuntu Server Book*, *Knoppix Hacks* and *Ubuntu Hacks*. He is currently the president of the North Bay Linux Users' Group.

Resources

GRUB Wiki: **grub.enbug.org**

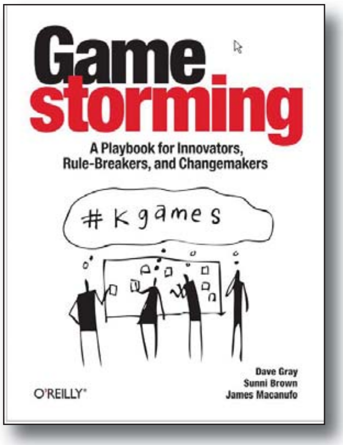
Ubuntu GRUB Community Documentation: **https://help.ubuntu.com/community/Grub2**

Onset's HOBO U12 Data Loggers



The data-logger expert Onset has expanded the capabilities to its HOBO U12 line, which now can measure and record kilowatts, air velocity, gauge pressure, differential pressure, DC current and other energy and environmental parameters. Onset attributes this new functionality to a new, compact power adapter, which enables energy and building management professionals to power external sensors that require 12-volt A/C excitation power conveniently. The new functionality augments existing measurement parameters, such as air temperature, relative humidity, light intensity, AC current and AC voltage. HOBO U12 Data Loggers also can record data unattended for up to months at a time, storing up to 43,000 measurements. Using a USB connection, HOBO U12 data loggers offer convenient, high-speed data offload directly to a computer or to a HOBO U-Shuttle data transport device.

www.onsetcomp.com



Dave Gray, Sunni Brown and James Macanufo's *Gamestorming* (O'Reilly)

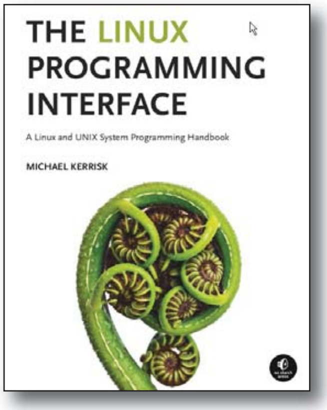
The authorial threesome Dave Gray, Sunni Brown and James Macanufo have just released an interesting new book, *Gamestorming: A Playbook for Innovators, Rule-Breakers, and Changemakers*. The subtitle indicates that the book is targeted squarely at us—that is, "people who want to design the future, to change the world, to make, break and innovate." The book's premise is that 200 years of industrial habits are embedded in our workplaces, our schools and our system of government, and certain strategies are required to make the changes necessary to "win in the 21st Century". *Gamestorming* is full of practical solutions that help one engage people in a project, to get better traction and move more quickly with groups, to make things happen and get better, faster decisions and results.

www.oreilly.com

Michael Kerrisk's *The Linux Programming Interface* (No Starch)

No Starch Press, publisher of Michael Kerrisk's 1,500-page book *The Linux Programming Interface*, bills the title as the "authoritative work" and "definitive guide to the Linux and UNIX programming interface". Kerrisk, who is the maintainer of the Linux man pages project, presents detailed descriptions of the system calls and library functions that one needs in order to master the craft of system programming. He accompanies his explanations with clear, complete example programs. Some key topics include using signals, clocks and timers; creating processes and executing programs; writing multithreaded programs using POSIX threads; building and using shared libraries; performing interprocess communication using pipes, message queues, shared memory and semaphores; and writing network applications with the sockets API.

www.nostarch.com

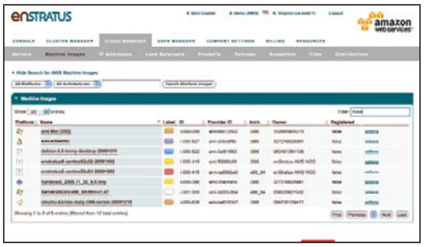

THE LINUX PROGRAMMING INTERFACE
A Linux and UNIX System Programming Handbook
MICHAEL KERRISK

SUGARCRM. THE CLOUD IS OPEN

SugarCRM's Sugar 6

SugarCRM hopes to (warning of sugar metaphor ahead) sweeten up the CRM space with Sugar 6, the latest edition of the company's flagship CRM system. The buzz around Sugar 6 involves its integration of social-media tools, such as Twitter, Facebook and LinkedIn, directly within the user interface. Users now can listen, monitor and aggregate social data and tie it to their existing customer information in a simple yet highly structured manner.

www.sugarcrm.com



enStratus

The recent brewing going on at enStratus has resulted in a new edition of its self-titled suite of tools for managing cloud infrastructure that now includes VMware's vSphere. With this expanded support, customers can leverage a unified solution to manage vSphere as well as private and public cloud infrastructures. Features that customers now can leverage in a vSphere deployment include self-provisioning, advanced user management, financial controls and automation. In addition to supporting vSphere and vCloud Express from VMware, enStratus also supports leading cloud infrastructure platforms from Amazon Web Services, Eucalyptus, GoGrid, Rackspace, Cloud.com, ReliaCloud, Terremark and Windows Azure.

www.enstratus.com

Axigen's Messaging Platform

The word from Axigen is that the new v. 7.4 of the Axigen integrated e-mail, calendaring and collaboration platform is available and replete with many fine new features. Hundreds of additions have been added to the Axigen Mail Server, the Webmail interface, the Outlook Connector and the Active Directory Connector. Support for FreeBSD 8.x has been added. In addition, Axigen's application now offers a revamped licensing model that includes free e-mail users.

www.axigen.com

Blancco's Mobile Edition

Bond would be up a creek without a gadget should his arch-rivals get their hands on Blancco Mobile Edition, an application designed to eliminate the risk of inadvertent data leaks by erasing retired smartphones that may contain sensitive business and/or personal information. Capable of erasing up to 150 such devices per day, Blancco Mobile Edition helps IT security managers set and enforce end-of-service policy related to smartphones, an area frequently overlooked as a source of data breach. The application was developed for erasure of smartphones running major platforms like Symbian, RIM for BlackBerry, and Microsoft; support for Android-based platforms is slated for late 2010.

www.blancco.com

xTuple's Web Portal

The role of xTuple's new Drupal-based Web Portal is to extend the firm's open-source CRM, accounting and ERP applications to the Web. The xTuple Web Portal, which can be hosted on-premise or through an xTuple Partner, enables companies to improve customer service, establish an internal help desk, build deeper relationships with partners or suppliers and engage end users of the company's products. Once a conversation is initiated in the xTuple Web Portal, an incident is created automatically that users can categorize, prioritize and assign. In addition, users can create opportunities, to-do lists or even full projects from that initial incident. One can utilize all three xTuple editions on the Web Portal, namely the free xTuple PostBooks, xTuple Standard and xTuple Manufacturing.

www.xtuple.com

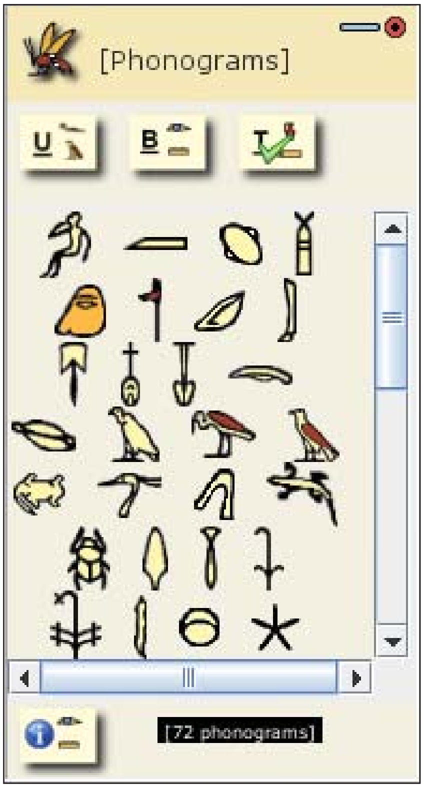
Please send information about releases of Linux-related products to newproducts@linuxjournal.com or New Products c/o *Linux Journal*, PO Box 980985, Houston, TX 77098. Submissions are edited for length and content.

Fresh from the Labs

Kemet—Computing Tools for Hieroglyphs

web.me.com/fabricemaupin/Kemet/Home_Page.html

For anyone interested in the translation of hieroglyphics—or should I say, transliteration (I'm thinking archaeologists and *Tomb Raider* fans)—the Kemet Project might be a great place to start.



Kemet comes with a series of "phonogram" hieroglyphs ready to use.

Installation As far as system requirements go, aside from X, the only big dependency I could find was Java. The rest of the installation process isn't quite as simple. As the project is undertaking a large and daunting task, it has been split into several parts: an API, an LF (which stands for something—I'm not sure) and a Kemet application (which at the time of writing was "coming soon"). Thankfully, a demo has been provided under the KemetLF package, so head to the downloads section and grab the latest package.

Extract the new package and look inside the new folder. There should be a file called KemetLFDEMO.zip. Extract this into a new subdirectory of its own and look in the new directory. You should see a file called KemtLFDEMO.jar; flag this as executable to run it. If you're using a file manager, you should be able to do something along the lines of right-clicking and checking "executable" in the permissions section. If you're using a terminal, you can flag the file as executable with this command:

```
$ chmod u+x KemetLFDEMO.jar
```

From here, if you're lucky, you can run it by clicking on the file or entering the command:

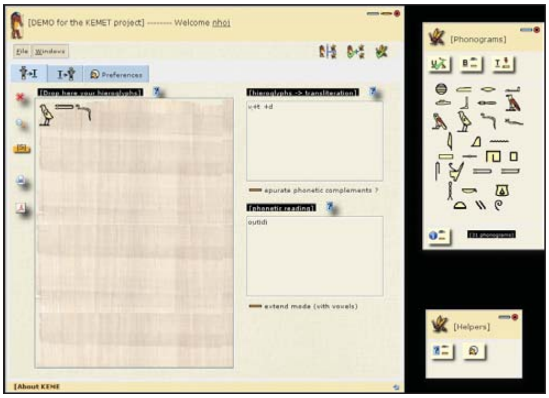
```
$ ./KemetLFDEMO.jar
```

Usage First, as the information window at the start points out, this is a demo—only some of the final features are implemented. Three windows will open: the main window, a Phonograms window that contains hieroglyphs to drag and drop onto the main window, as well as a small help window to explain some of the basic features.

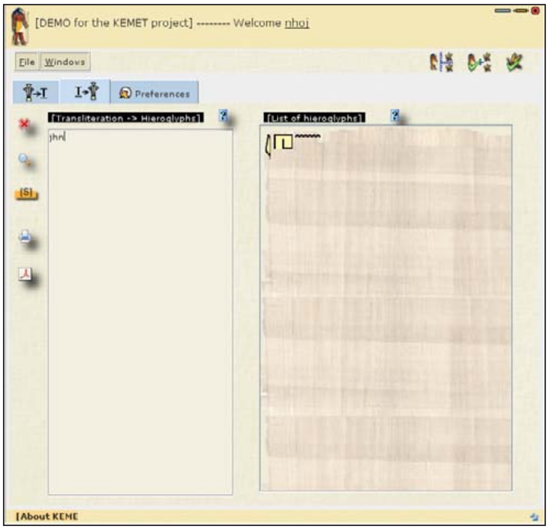
The main window is broken into three tabs: the last is a Preferences tab (self-explanatory), and the first two tabs are for Hieroglyphs to Transliteration and Transliteration to Hieroglyphs, respectively.

The first tab is for dragging selected phonograms from their window into the big writing space on the left. Once dragged, any symbols will appear in the right two panes in both transliterated format as well as a phonetic reading.

Open the second tab, and in the left-hand pane, you can enter your own text in the Latin alphabet and have it appear in hieroglyphs on the right. Pretty neat, huh? Something that alarmed me, but will amuse anyone who's already well versed on the subject, is that there are no vowels, only consonants. When I entered my name, it came up as "jhn", but at least I could see my name in hieroglyphs on the right.


[DEMO for the KEMET project] -------- Welcome nhoj
File
Windows
Preferences
[Drop here your hieroglyphs]
[hieroglyphs -> transliteration]
u4t +d
apurate phonetic complements ?
[phonetic reading]
outidi
extend mode (with vowels)
[About KEME
[Phonograms]
[21 phonograms]
[Helpers]

Once hieroglyphs have been dragged into place, you can get a reading in both transliteration and phonetic form.


[DEMO for the KEMET project] -------- Welcome nhoj
File
Windows
Preferences
[Transliteration -> Hieroglyphs]
jhn
[List of hieroglyphs]
[S]
[About KEME

Type a word (minus vowels) in the left and get a series of hieroglyphs on the right. Cool!

Once again, I'm covering a topic I know nearly nothing about, so I'm sure I've probably made some silly errors and given some misinformation. Still, this project charmed the pants off me—from its enormous scope to the small details in making the GUI elements appear as hieroglyphic-looking characters to a sandy background for the windows and text boxes. I hope this project makes it to full release. Only open source could give the public a free tool like this.

VirtMus—the Modern Music Stand

virtmus.com

I'm actually not the first person here at *LJ* to cover this program. Dave Phillips wrote about it a few months back in his excellent "News In The Linux Audio World" column on LinuxJournal.com (great stuff by the way, I love Dave's work). I was so impressed with the idea, I thought it deserved some further coverage here.

As any musician who deals with sheet music can attest, there comes the annoying time when you have to turn the page, which means you either have to find a way to take one hand off your instrument and keep playing, or you simply have to stop playing and resume once the page is turned. Either way, it's a dilemma that has existed since classical times, and it has caused so many problems through the years that people often are employed just to turn the page at the appropriate time (there's even a French film called *The Page Turner*).

According to the VirtMus Web site:

> VirtMus (virtual music) is a free application that allows the user to display sheet music and turn pages without removing the hands (or feet) from the instrument the music is performed on. This feature is very useful during concerts and practice sessions as it allows the musician to focus on performing the music without interruption. The software also allows the users to store and organize their entire sheet music collection on a laptop, making it fully portable and available at a click of a button.

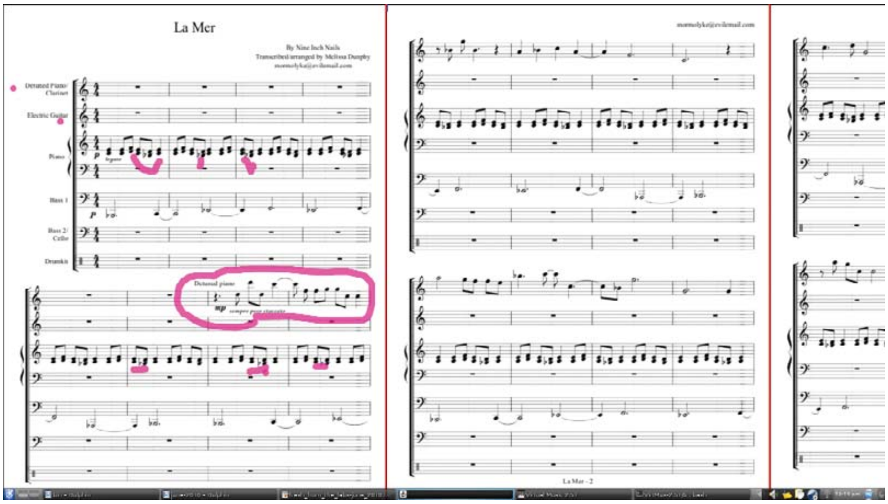
Installation VirtMus is a cross-platform, Java-based program, and as such, the only real requirement is a working installation of Java Runtime Environment (JRE) 6 SE. As for the program itself, head to the download section at the Web site and grab the latest .zip file. Extract it, and make your way to the bin folder inside the main VirtMus directory. Here, you'll see executables for Linux and Windows, with the Linux binary simply named virtmus.

Depending on your file manager, you may be lucky enough to click on the file and have it run, but don't panic if it doesn't. Simply open a terminal inside the bin folder and enter the command:

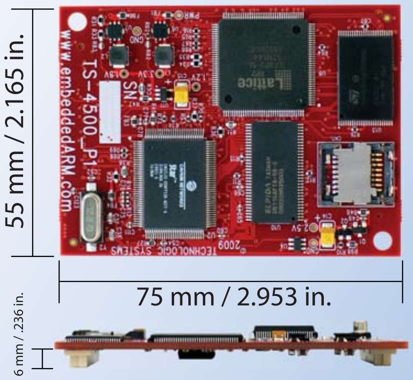
```
$ ./virtmus
```

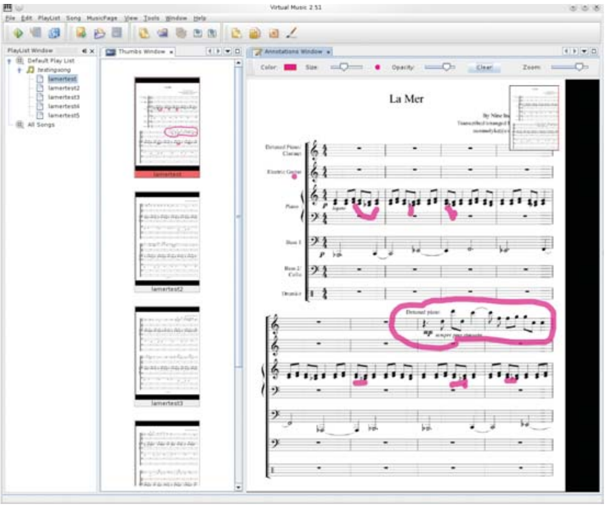
Usage Once you're inside the program, you need to begin by adding a new song. From the menu, choose Song→New Song, and you'll be prompted for a new project filename. Enter a name and click save. Now that you have a new song underway, you need to add some musical pages.

Look in the PlayList Window on the left, and under Default Play List in the drop-down menu will be the name of your song. To add some musical pages, from the menu, choose MusicPage→Add Pages. Here, you can add either PDFs or image files. I had some problems with the musical characters in the PDFs being rendered properly (probably just my system though),



VirtMus lets you turn musical pages easily with one button push (ideally by foot), while you continue playing your instrument.





VirtMus also lets you make your own notation on the pages before you play (I made those weird pink marks).

so I converted each page to an image file and imported them separately.

Now that you have loaded some pages, if you look inside the Thumbs Window, you'll see a preview of the order in which pages will appear in a thumbnail image format, which you then can modify in the playlist on the left.

The annotations window on the right is yet another piece of genius. Here, you can make your own personal notes on the page itself (with the added benefit of leaving the original intact), highlighting difficult sections or notes you should pay attention to while you play. Click on a page of notation from the playlist on the left, and the page will appear in the Annotations Window on the right.

And, now we reach the entire point of this program, the "Go Live" feature. Either choose View→Go Live from the menu, or press F5, and suddenly, you'll be transported into a full-screen view of your pages, with the first two pages you're currently viewing featured prominently on-screen, and the next page trailing just off to the right.

To begin page turning, click the left-mouse button to move ahead, or click the right-mouse button to go back. Numerous other buttons on the keyboard will move the page forward, including Space, Enter, the arrow keys, Ctrl and so on, but the only one I found for going backward was Page Up, although I bet there's another key somewhere that I missed in my late-night testing.

Of course, clicking a mouse or pressing something on the keyboard still leaves you with much the same problem—you have to take a hand off the instrument to turn the page—although it is at least in a less clumsy format than the traditional hand-and-paper method. But fear not! The Web site recommends using a USB footswitch, removing the need to let go of your instrument.

However, I'm not sure how common or cheap USB footswitches are. Being the cheapskate that I am, I'd plug in a second USB keyboard (the cheapest I could get), remove some of the keys around the spacebar, and then tap the spacebar with my toes. It'd look a little funny, but a tightwad like myself wouldn't really care!

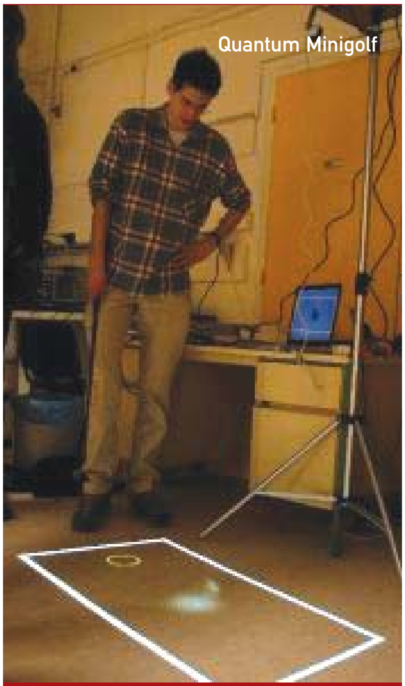
Unfortunately, VirtMus development seems to have dried up. The last time there was any development (at least at the time of this writing) was in 2009. One motive in writing this piece was to spur the project on to completion. I know a great many musical institutions that could benefit from this program and to have it reach full maturity would be fantastic.

I'm going to leave you now with the words of my fellow *Linux Journal* colleague, Dave Phillips:

> Alas, development of VirtMus appears to be stalled. The concept is cool—an affordable alternative to dedicated digital music displays—and the source code is freely available, so there's hope for the project's revival. And, if any of my readers happen to step up to the challenge, please let me know. I'd be happy to update the status of the VirtMus Project.■

John Knight is a 26-year-old, drumming- and climbing-obsessed maniac from the world's most isolated city—Perth, Western Australia. He can usually be found either buried in an Audacity screen or thrashing a kick-drum beyond recognition.

Brewing something fresh, innovative or mind-bending? Send e-mail to newprojects@linuxjournal.com.



Quantum Minigolf

Project at a Glance

Quantum Minigolf

quantumminigolf.sourceforge.net

I'm working alongside German developer Friedemann Reinhard to bring you this project next month. *Quantum Minigolf* is nearly the same as the game *Minigolf*, except that the ball obeys the laws of quantum mechanics. For instance, a ball can be at several places at once. It can diffract around obstacles and interfere with itself. Apart from that, the rules are the same. You can play on various tracks involving various obstacles. You hit the ball with a club and try to kick it into a hole on the other side of the track. I've played it already, and the concept is pretty wacky, I can tell you!

SharePoint in Boston!

SPTechCon
The SharePoint Technology Conference

Oct. 20-22, 2010 → Boston

Hyatt Regency Cambridge

The best SharePoint education is at SPTechCon Boston!

Planning a quick migration to SharePoint 2010? Seeking to maximize your productivity on SharePoint 2007? Sign up for the SharePoint Technology Conference in Boston, where Microsoft engineers and Microsoft MVPs teach the industry's best workshops and classes for IT Pros, Software Developers and Business Users. Come to SPTechCon to learn, network and go deep with SharePoint. See you in Boston!

Download the Complete Class Listing at www.sptechcon.com!

Learn from the most experienced SharePoint experts in the industry!

Joel Oleson, Brett Lonsdale, Michael Noel, Shane Young, Paul Stork, Todd Klindt, Robert Bogue, Andrew Connell, Heather Solomon

John Ross, Scott Jamison, Laura Rogers, Scott Hillier, Gary Lapointe, Rob Windsor, Brian Culver, Maurice Prather, Randy Drisgill

Fabian Williams, Errin O'Connor, Scott Peterson, Daniel Antion, Paul Swider, Dux Raymond Sy, Jennifer Mason, Chris McNulty, Matt Passannante

Keynote Speakers

Steve Fox
Microsoft
Sr. Evangelism
Manager, Developer
and Platform Group

Bill English
SharePoint MVP,
Noted author and
speaker, Founder,
Mindsharp

Michael Doyle, Ruven Gotz, Marcel Meth, Peter Serzo, Joshua Haebets, Mark Miller

Brian Jackett, Dustin Miller, Marc Anderson, Mark Rackley, Mike Watson, Geoff Varosky

DIAMOND SPONSOR
AvePoint® Unleashing the Power of SharePoint™

PLATINUM SPONSORS
commvault solving forward
SharePoint Hosting Pioneer fpweb.net
QUEST SOFTWARE Smart Systems Management
ESRI
epcgroup.net
hp

GOLD SPONSORS
susQtech
StoragePoint
Syntergy
AXceler
Atalasoft
idera TOOLS FOR SQL SERVER
CorasWorks
BURNTSAND
adlibsoftware
TITUS LABS
boldbrick
MacroView
SHAREPOINT911 THE SHAREPOINT AUTHORITY

SILVER SPONSORS
KnowledgeLake
rssbus
Double-Take
HiSoftware
NINTEX
CubisOne
SharePoint Engine
Optimal IdM

A BZ Media Event

Follow us at twitter.com/SPTechCon

www.sptechcon.com

SOFTWARE

Easy Database Backups with Zmanda Recovery Manager for MySQL

Zmanda Recovery Manager makes it easy to dump your database and that homegrown backup solution you've been using and meaning to replace. DANIEL BARTHOLOMEW

Recently, I had a chance to test the community edition of Zmanda Recovery Manager for MySQL. I was partly testing to make sure it worked with MariaDB, Monty Program's drop-in replacement for MySQL, but I also was testing to see whether it would work well for our own database backups.

Monty Program has arguably the most in-depth knowledge of the MySQL codebase on the planet. But apart from some large servers we use for performance and other testing, our actual database usage and needs are similar to many other small- to medium-size companies. For our Web sites, we need only a couple small database servers. The datasets for each database are not large, a couple hundred megabytes each. But, we still don't want to lose any of it, so backups are a must.

I've long used a homegrown set of shell scripts to manage backing up our databases. They're simple and work well enough to get the job done. They lack some features that I've never gotten around to implementing, such as automated retention periods and easy restores. The setup process also is more involved than I would prefer. They get the job done, but I've always wanted something a little better, and I've never had the time to do it myself.

This is where Zmanda Recovery Manager for MySQL enters the picture. ZRM Enterprise edition was reviewed by Alolita Sharma in the September 2008 issue of *Linux Journal*, but I'm never very interested in Enterprise editions. They always have proprietary bits, and I've never trusted GUI tools as much as their command-line counterparts. Luckily, where there is an "enterprise" version there almost always is a "community" version lurking somewhere in the shadows.

Like many other community editions, Zmanda Recovery Manager for MySQL, Community Edition (let's just call it ZRM) lacks the "flashy bits". Things like the graphical "console" application, Windows compatibility, 24x7 support and other high-profile features of its Enterprise sibling are missing in the Community version. But the essentials are there, and it has one big feature I like: it is fully open source (GPL v2). The key metric, however, is does it do what I need it to do?

To find out, I set up a small test environment (I didn't want to test on live, production databases) and gave it a spin. See the Setting Up a Test Environment sidebar for details on what I did prior to installing and testing ZRM.

Installing and Using ZRM

Zmanda offers packages for Debian, RPM and Solaris/OpenSolaris systems and their derivatives on its Web site. A source package also is available. Because I'm using Ubuntu 10.04, I downloaded the latest stable Debian package (mysql-zrm_2.2.0_all.deb at the time of this writing) from the ZRM download page.

ZRM requires the libxml-parser-perl and libdbi-perl packages, the mariadb-client or mysql-client package, and something that allows it to send e-mail messages (for notifying the correct people of a backup's success or failure). If you are running ZRM on the same server as your database, the Perl and client packages likely already will be installed. If you elect to do what I did and run ZRM from a dedicated backup server, these will need to be installed:

```
apt-get install libxml-parser-perl libdbi-perl \
                mariadb-client bsd-mailx
```

When mailx is installed on Ubuntu, it also will install postfix (unless you already have a different MTA installed), but other MTAs (mail transport agents) may be the default on your distribution. During the installation of the postfix package, I chose the basic "Internet site" setting, which provides just enough of a configuration to allow the server to send e-mail.

The ZRM package expects a user named "mysql" to exist. This user typically is created when MySQL or MariaDB is installed, but because my backup server has only the mariadb-client package installed, the mysql user didn't exist, so I needed to create it. I also chose to give the new user the same home directory that the user would have had if the user had been created as part of an Ubuntu mariadb-server installation:

```
sudo adduser --system --group --home="/var/lib/mysql" mysql
```

With dependencies finally out of the way, I was ready to install Zmanda Recovery Manager. I installed it like so:

```
dpkg -i mysql-zrm_2.2.0_all.deb
```

The installation itself is pretty boring, and it looks no different from any other package install:

```
me@backuphost:~$ sudo dpkg -i mysql-zrm_2.2.0_all.deb
```

SETTING UP A TEST ENVIRONMENT

For testing and evaluating ZRM, I set up three virtual servers running Ubuntu 10.04 LTS, installed MariaDB on them, and then downloaded the example "employees" test database from **launchpad.net/test-db**.

Installing MariaDB is easy on Debian, Ubuntu and CentOS because of some MariaDB repositories maintained by OurDelta.org. The site has instructions on how to configure your system to use the repositories. For Ubuntu 10.04, I did the following:

1. Added the following lines to /etc/apt/sources.list.d/mariadb.list:

```
# MariaDB OurDelta repository for Ubuntu 10.04 "Lucid Lynx"
deb http://mirror.ourdelta.org/deb lucid mariadb-ourdelta
deb-src http://mirror.ourdelta.org/deb lucid mariadb-ourdelta
```

2. Added the repository key to apt:

```
apt-key adv --recv-keys \
          --keyserver keyserver.ubuntu.com 3EA4DFD8A29A9ED6
```

3. Updated apt and installed mariadb-server:

```
apt-get update
apt-get install mariadb-server
```

Installing mariadb-server looks and acts just like installing mysql-server.

With the database server installed, I loaded the test database. To load the employees test database into MariaDB, I first downloaded and untarred it and then used the mysql command-line program to load it into MariaDB like so:

```
tar -jxvf employees_db-full-1.0.6.tar.bz2
cd employees_db/
mysql -u root -p -t < employees.sql
```

The employees test database uses a couple hundred megabytes of disk space. This is in line with the size of our "real" databases. But more important than the comparative size, the employees test database comes with a handy verification script that lets me test that the data is correct. Verifying the data is as simple as this:

```
mysql -u root -p -t < test_employees_sha.sql
```

With the test database servers set up, I then created a fourth virtual machine with a base Ubuntu Server install on it to act as my backup server. Now I was ready to test using ZRM for backup and recovery with the ability to verify that the recovery was successful.

```
Selecting previously deselected package mysql-zrm.
(Reading database ...
    42938 files and directories currently installed.)
Unpacking mysql-zrm (from mysql-zrm_2.2.0_all.deb) ...
Setting up mysql-zrm (2.2.0) ...
Updating ownership of previously backed up data sets

Processing triggers for man-db ...
me@backuphost:~$
```

So what did the package install? A look at the output of `dpkg -L mysql-zrm` reveals that the package installs several Perl scripts into the /usr/bin/ folder and creates the following directories:

- /usr/share/mysql-zrm — a "plugins" folder with several Perl scripts inside.
- /usr/share/doc/mysql-zrm — various docs and README files.
- /usr/lib/mysql-zrm — various Perl modules.
- /etc/mysql-zrm — configuration files.
- /var/log/mysql-zrm — empty directory for log files.
- /var/lib/mysql-zrm — the folder where backups go (initially empty).

The package also installs man pages for the scripts and config files, and xinetd and logrotate config files.

Now I was ready to set up some backups. ZRM uses the concept of "backup sets" to refer to backup settings for a single server or backup job. To create a new backup set, you create a

If disaster strikes and you need to restore a backup to a server, the first thing is to determine the location of the most recent successful backup, and then to use it.

new directory under /etc/mysql-zrm/ and copy the default configuration file into the new directory, like so:

```
cd /etc/mysql-zrm
mkdir -v backupsetname
cp -vi mysql-zrm.conf backupsetname/
```

The folder can have any name you want. The mysql-zrm.conf file is, by default, completely commented out. The file has inline documentation for each configuration directive, and it is pretty easy to read. For my project, I wanted compressed and encrypted logical backups, so the lines I customized and uncommented were these:

```
backup-mode=logical
backup-type=regular
retention-policy=30D
compress=1
compress-plugin=/bin/gzip
encrypt=1
encrypt-plugin="/usr/share/mysql-zrm/plugins/encrypt.pl"
all-databases=1
user="backup-user"
password="examplepassword"
host="db1.example.org"
mailto="my-email@example.org"
```

The `user` and `password` in the set of variables above is a MariaDB database user, not a system user. This user is created like other database users using the mysql command-line tool and a GRANT statement. Here's the GRANT statement Zmanda recommends:

```
GRANT select, show view, create view, insert, update,
      create, drop, reload, shutdown, alter, super, lock tables,
      replication client
          on *.*
          to 'backup-user'@'backuphost'
          identified by 'examplepassword';
```

If you set up ZRM on the host it is backing up, `backuphost` in the above statement would be changed to `localhost`. At this point, I also needed to configure one of our database servers to allow remote logins. This is done by setting the bind-address variable in the /etc/mysql/my.cnf file to the IP address of the database server and then restarting mysqld.

Backups can be either "raw" or "logical". Raw backups are actual copies of the database files. Logical backups are a dump (using mysqldump) of the contents of your database in SQL. Raw backups can be restored only to a server running the same version of MariaDB or MySQL. Logical backups do not have this restriction and can be loaded successfully onto servers running older or newer MariaDB/MySQL versions (depending on whether the new server to which you're restoring supports the same features that the old one did).

Backup types are "regular" and "quick". The quick type applies only to raw backups and only if your database is stored on an LVM logical volume. A raw+regular backup is a copy of your MariaDB/MySQL data files made using mysqlhotcopy. A raw+quick backup is an LVM snapshot of those data files. If you are doing a logical backup, the quick backup type is not available.

The retention-policy variable tells ZRM how long you want to keep backups. The default is 10W, which stands for ten weeks. Other suffixes you can specify include D for days, M for months or Y for years.

ZRM uses "plugins" to extend its functionality. Several plugins come with ZRM, including a couple that can be used to copy backups from a remote database server to the server running ZRM, and a plugin to encrypt backups. Some plugins are just wrapper scripts, like the encryption plugin, which is a wrapper around GPG. Other plugins are just system binaries. For example, the default "compress" plugin is just the gzip program, no wrapper script required. Any or all of these can be replaced with your own preferred solutions.

Configuration and setup varies per plugin. The encryption plugin, for example, requires the creation of a file named .passphrase in the /etc/mysql-zrm/ folder. This file contains the password used when encrypting backups. The steps I followed when creating this file are:

```
touch /etc/mysql-zrm/.passphrase
echo 'mysupercoolhardtoguesspassword' > /etc/mysql-zrm/.passphrase
chmod -v 700 /etc/mysql-zrm/.passphrase
```

Furthermore, because the encryption plugin uses GPG, the .gnupg folder needs to be present in the root user's home directory (the backups are spawned by root). It wasn't present for me, so I created it:

```
mkdir -v /root/.gnupg
chmod -v 600 /root/.gnupg
```

Finally, I was ready to perform some backups. Running a manual backup is pretty easy:

```
mysql-zrm-scheduler --backup-set backupsetname \
                    --backup-level 0 --now
```

Scheduling backups also is easy. Like running a manual backup, to schedule backups, you use the mysql-zrm-scheduler script, but instead of having the backup start "now" you set an interval and a start time, like so:

```
sudo mysql-zrm-scheduler --add --backup-set backupsetname \
                         --backup-level 0 \
                         --interval daily --start-time 01:00
```

The above backup will run every day starting at 1am. You can view the schedule with `mysql-zrm-scheduler --query`, or because ZRM schedules backups using cron, you simply can query the root crontab with `crontab -l` (running the command as root).

When you add your first schedule, ZRM also will add a cronjob for running the "purge" action for removing backups that are older than the retention period.

To check that your backup data hasn't been corrupted since the backup was made, use the mysql-zrm script with the verify-backup action:

```
mysql-zrm --action verify-backup --backup-set backupsetname
```

To view stats on recent backups, the mysql-zrm-reporter can help:

```
mysql-zrm-reporter --show backup-performance-info
```

If disaster strikes and you need to restore a backup to a server, the first thing is to determine the location of the most recent successful backup, and then to use it. The mysql-zrm-reporter script is an easy way to reveal the location:

```
mysql-zrm-reporter  --show restore-info \
                    --where backup-set=backupsetname
```

In the output, look for the backup_directory of the most recent backup where the backup_status is "Backup succeeded". The backup_directory path will look something like this:

```
/var/lib/mysql-zrm/backupsetname/20100607141122
```

With this information, you can perform a restore, like so:

```
backup_dir=/var/lib/mysql-zrm/backupsetname/20100607141122
mysql-zrm-restore --backup-set backupsetname \
                  --source-directory $backup_dir
```

Expect restores to take a while, depending on the size of your database. In my testing setup, after the restore completed, I verified the data as described in the Setting up a Test Environment sidebar, and everything checked out.

Conclusion

At the end of my evaluation, I decided to use ZRM for our database backups. My use case is logical backups over the network, and for those, the Open Source community edition of ZRM works very well.

I like how easy scheduling new backups and creating new backup jobs is. With Zmanda, I can configure backups for a new database server effortlessly, something that could not be said about my homegrown solution. Restores also are easy, which will be appreciated if the unthinkable happens and I need to restore from a backup. And, thanks to ZRM's use of standard tools, even if I can't restore using ZRM, the backup contains a file that I can load into the database manually either as-is (if I'm not encrypting or compressing my backups) or after a little processing using the standard gunzip and GPG tools.

Zmanda Recovery Manager for MySQL is not perfect. During my testing, I was never able to get raw backups working properly over the network, for example. Another issue, though minor, is that the man pages have formatting issues that make them hard to read. Some of the error messages are not the most informative as well, and the documentation could be improved and expanded. But, the software is built using solid open-source tools, it doesn't try to re-invent the wheel at every turn, and it works for the backups I want to do.

In the end, the thing that tipped the scales for me was that ZRM offers several things that my homegrown scripts do not. These include the automatically creating a checksum for verifying that a backup is still good, faster and very customizable setup for new database servers, and easy restores. I could add all of these to my scripts, given time. But it's time I don't have at the moment, and I never seem to have enough (if you know where to find some, let me know). So despite some rough edges, I've found Zmanda Recovery Manager for MySQL, Community Edition to be a good backup solution for all my MariaDB servers.■

Daniel Bartholomew works for Monty Program (montyprogram.com) as a technical writer and system administrator. He lives with his wife and children in North Carolina and can often be found hanging out on #maria and #linuxjournal on Freenode IRC.

Resources

Zmanda Recovery Manager for MySQL: **zmanda.com/backup-mysql.html**

Community Downloads Page: **www.zmanda.com/download-zrm.php**

Large-Scale Web Site Infrastructure and Drupal

Setting up a Drupal Web site is pretty simple these days, until it gets popular, then you need to bring out the big guns and start finding and fixing the performance bottlenecks. In this article, we show some of the techniques that can allow your Drupal Web site to scale to the grandiose levels you originally hoped for.

JERAD BITNER and NATE HAUG

When Twitter experiences an outage, users see the infamous "fail whale" error message, an illustration of twit-birds struggling to hoist a sleeping cartoon whale into the air along with the words "Too many tweets! Please wait a moment and try again." It happens so often, Twitter has a much-heralded illustration for it. Not too long ago, many readers may remember Facebook going down for days at a time. True, those sites are dealing with extraordinary levels of traffic, but smaller sites often face the same problems. How come? First, Web sites are no longer a collection of static pages. Nowadays, Web sites combine social-networking features with highly customized content for individual users, meaning most pages have to be assembled on the fly. Second, content is changing—rich media, on-line advertising, video, telephony. There's more than text forcing its way through the pipe, and network traffic only continues to grow. Addressing this tandem of complexity and load is the bane of many growing social-media Web sites' existence. What follows are some clever ways to address this whale of a problem.

Surprisingly, the solutions to most scaling problems are frequently the same, regardless of the technology upon which the site was built. Lullabot (the parent company of this article's authors) is a Drupal development company, meaning that most of our experience is centered around the typical LAMP stack (Linux, Apache, MySQL and PHP), although most techniques are universal, and some of the most advanced performance software is platform-neutral.

Server Infrastructure

One of the main factors in scaling a Web site is, of course, the hardware (Figure 1). System administrators always can throw more hardware at a problem and solve it at least temporarily, if they have the resources to do so. Quite a few services can be put in place before this needs to be done, and developers can selectively optimize the application by reducing or optimizing queries. Nevertheless, when it comes to sheer numbers of users and bandwidth over a short amount of time, there almost always comes a point where it's necessary to include hardware in the mix. That's why it is important to have your hardware infrastructure planned in a way that it rapidly can scale upward on a traffic spike, and back down when your traffic recedes.

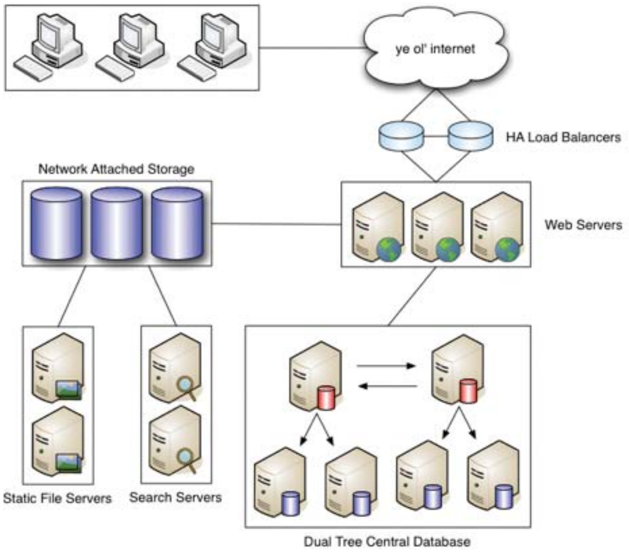


Figure 1. Hardware Stack

A typical setup, whether virtual or dedicated, usually includes multiple Web servers, multiple database servers and sometimes even separate caching servers, all behind a load balancer that distributes traffic between machines. Depending on its processor speed and the amount of available memory, a Web server or database often can double as the caching server, because caching services usually require less resources than Apache or MySQL.

Although distributing traffic across multiple Web servers, or Web heads, can be a quick win, it can introduce problems with managing file uploads. If requests are being distributed round-robin by the load balancer, a user may upload a file on one server but then be switched to a different Web server after the upload, which doesn't have the newly uploaded file. To solve this problem, a file server also is added into the mix. The file server is usually some form of NAS (Network Attached Storage) or an NFS (Network File System) mount that allows the application to share files between machines. Each Web head will have a copy of the application stored in the Web root, but when it comes to the files that are uploaded or changed often by the users of the application, an NFS mount connects all the servers to a shared file location.

Cache Techniques

The other main factor in scaling a Web site is, of course, the software (Figure 2). To scale effectively, high-traffic Web sites require some flavor or flavors of caching. Caching mechanisms are not mutually exclusive, and most high-profile sites combine several. Most types of caching seek to reduce the amount of disk access necessary to render a page or compile higher-level languages into bytecode so they're faster to run—the closer to machine language the better.

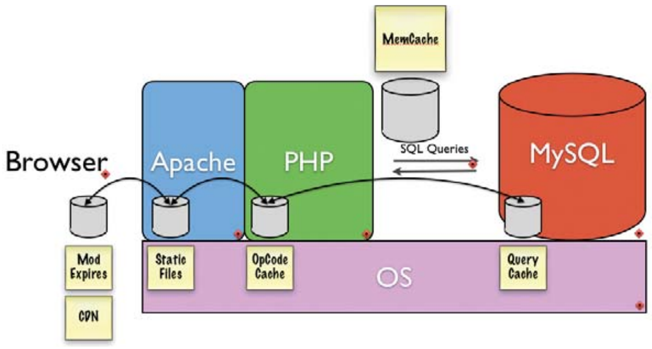


Figure 2. Software Stack

APC (Alternative PHP Cache) and other opcode caches save the Web server from having to read, parse and compile PHP files on every request. APC is a free, open-source opcode cache and is pretty much the standard. It will come built in with PHP 6, but there are many different ones that perform differently.

Modern content management systems, like Drupal, can make a plethora of database calls on every page request. Because calls to the database hit the disk, it is often a bottleneck. Memcached is a service that allows entire database tables to be stored in memory, dramatically speeding up queries to those tables and alleviating strain on the database. It behaves as though it were a giant

If **Apache** were a physician, **Varnish** would be the triage nurse.

hash table and serves this data out of memory. Memcached is free, open source and in use by a ton of high-traffic sites. Memcached is installed alongside MySQL on the database server in most typical setups. However, the database server needs to have a lot of RAM available if Memcached and MySQL are sharing this critical resource. There are occasions when Memcached is actually placed on its own server, completely decoupled from the database server, which precludes Memcached from using too much of the database server's memory.

Varnish is an excellent high-performance, HTTP accelerator. The technical term for Varnish is a "reverse proxy cache", meaning that it handles the requests when you visit a Web site. If Apache were a physician, Varnish would be the triage nurse. After each anonymous page request is made, Varnish makes a copy of the page in an ultra-fast storage so that the next time the page is requested, it returns it immediately, circumventing a bootstrap of Apache, PHP, MySQL, Memcached or any other technologies your Web site may require to serve pages. If Varnish doesn't have a copy of the file or page being requested, it will send the request on to Apache. And, it's really a huge win if you're going to be serving static content.

Outsource Search

Search is resource-intensive. Optimizing search will contribute to overall site performance and is a great process to outsource to another box. Solr can help ease an over-burdened Web server. Solr is a project from the Apache Foundation that takes the power of Lucene, a fantastic indexer and searcher, and exposes it as a Web service. Using HTTP POST and GET requests, you can feed documents to Solr for indexing and issue queries for searching. In Drupal, the Views module serves as a visual query-builder and handles search. With Views 3, in Drupal, you can plug in Solr to handle the search heavy-lifting instead of having Drupal hit MySQL for this, alleviating a load on your database server best left to a document indexer like Lucene.

Tune Apache

Apache's MaxClients setting is a limit on the number of simultaneous requests that can be served. If this limit is reached, users have to wait until a child process is freed up until they can connect. If this number is increased too much, however, there is a risk that the Web head will run out of memory. There's a standard formula for figuring out what this setting should be based upon the RAM available to the machine:

- formula: RAM/Average Apache Memory Size in Use = # max clients

- example: 2GB/20MB = 100 MaxClients

Apache's mod_expires setting controls the HTTP header information for anything served through Apache to your machine. If a resource has been cached on a user's computer, this setting can tell any subsequent request to that resource if it has expired and needs to be downloaded again. It's a good idea to have this turned on for text/HTML header types:

```
<IfModule mod_expires.c>
  ExpiresActive On
  ExpiresDefault A1209600
  ExpiresByType text/html A1
</IfModule>
```

The KeepAlive setting is a way to tell Apache to keep an HTTP connection alive for a period of time so that it can be reused. This has been shown to result in an almost 50% speed increase in latency times for HTML documents with many images. Turn this on and set the KeepAliveTimeout to 2 seconds:

```
KeepAlive On
KeepAliveTimeout 2
```

Optimize MySQL

MySQL is the most widely used database for Drupal, although Drupal 6 also supports Postgres. Drupal 7 has an object-oriented database abstraction layer that allows drivers to be written for many other database systems. There are some key things to keep in mind within MySQL's configuration that can help optimize your application for performance.

MySQL has a built-in query cache that is turned on by default. Make sure to afford a liberal amount of memory to this cache:

```
[mysqld] query_cache_size=32M
```

Once your application is built, it's a good idea to log slow queries for a short amount of time to get a list of queries that are taking a long time and can be examined with an EXPLAIN and then optimized:

```
log-slow-queries = /var/log/slow_query.log
long_query_time = 5
#log-queries-not-using-indexes
```

MySQL's EXPLAIN command is a great way to find out exactly what a particular query is doing in order to get some clues as to why it may be taking a long time to evaluate and return a result. One of the key things to look at is the number of rows that EXPLAIN tells you it had to search through. This may indicate that one of your tables, bursting at the seams, is a good candidate for a new index.

Taking a look at the following query, we see there are three fields that could have an index placed upon them in order to reduce the number of rows that a query has to search through in order to find the desired result:

```
...
FROM node node
WHERE node.status = 1
AND node.type IN ('story')
ORDER BY node.created DESC
```

The status, type and created fields are key to this query's result and can be indexed so that they are seen as a group:

```
mysql> ALTER TABLE node ADD INDEX (status, type, created);
```

Table locking can be a performance headache. By default,

7th Annual

High Performance Computing on Wall Street Show and Conference

September 20, 2010 (Monday) **Roosevelt Hotel, NYC**
Madison Ave and 45th St, next to Grand Central Station

Low cost Conference $295. Save $100. The Show is free. Register today.

Need for Speed, Low Latency, Cost Savings: Wall Street and the financial markets are looking for ways for traders to speed their orders to market.

High Performance Computing, Linux, Low Latency, Virtualization, Cloud, Grid, Blade, Cluster – the largest meeting of High Performance Computing in New York in 2010.

This major HPC networking opportunity will assemble 800 Wall Street IT professionals at one time and one place in New York in September 2010.

This HPC show will focus on speed, Low Latency, lowering computer costs. Catch the wave at our convenient and affordable New York marketplace.

This show will cover High Performance Computing, High Frequency Trading, Low Latency, Virtualization, Cloud, Grid, Blade, Cluster and new strategies to put these technologies to work.

Our Show is an efficient one-day showcase and networking opportunity.

Leading companies will be showing their newest products on-the-show floor live.

Register in advance for the full conference program which includes general sessions, drill down sessions, an industry luncheon, coffee breaks, exclusive viewing times in the exhibits, and more. Save $100. $295 in advance. $395 on site.

Don't have time for the full Conference? Attend the free Show. Register in advance at: www.flaggmgmt.com/hpc

Wall Street IT speakers and Gold Sponsors will lead drill-down sessions in the Grand Ballroom program.

This Show is a networking opportunity for the entire IT community.

Show:	Mon, Sept 20	8:00 - 4:00
Conference:		8:30 - 4:50

2010 Sponsors

Microsoft
IBM
hp
intel
redhat
CISCO
Corvil
FUJITSU
Solace Systems
BLADE NETWORK TECHNOLOGIES
sgi
APPRO HPC Cluster Solutions
FULCRUM microsystems
SUPERMICRO
ScaleMP
ARISTA
LINUX JOURNAL
waters
SD Times
Global Investment Technology
LINUX PRO MAGAZINE
HPCwire
SECURITIES INDUSTRY news
Inside Market Data
DWT
WINDOWS IN FINANCIAL SERVICES
bobsguide
WSTA Wall Street Technology Association

Show & Conference:
Flagg Management Inc
353 Lexington Avenue
New York, NY10016
(212) 286 0333
flaggmgmt@msn.com

Visit: www.flaggmgmt.com/hpc

Drupal's MySQL database tables are all set to MyISAM. Because MyISAM locks the entire table down during a query, high traffic may cause MySQL errors when a certain table is unavailable or locked. If you start seeing these errors, look at which tables are giving the error and evaluate whether they should be set to InnoDB instead. InnoDB does row locking instead of table locking. When evaluating, look to see if the table has any auto_increment fields, and keep in mind that converting this table may cause slow-downs on INSERTs, as InnoDB does a full table lock on INSERTs to avoid key duplication.

PHP

Static variable caching is a quick-and-easy win in PHP. Here is an example of a simple function with a simple query to the database:

```
function taxonomy_get_term($tid) {
  return db_fetch_object(
    db_query('SELECT * FROM {term_data} WHERE tid = %d', $tid)
  );
}
```

This function can be given a simple static variable, so that if this function happens to be called more than once on a page load, it can skip over the call to the database and serve the result out of this static cache:

```
function taxonomy_get_term($tid) {
  static $terms = array();
  if (!isset($terms[$tid])) {
    $terms[$tid] = db_fetch_object(
      db_query('SELECT * from {term_data} WHERE tid = %d', $tid)
    );
  }
  return $terms[$tid];
}
```

Application (Drupal)

Drupal is a content management framework that Lullabot uses to build high-performance Web sites on top of this infrastructure. Drupal is built with PHP as its primary programming language and has a ton of user-contributed modules freely available to extend its functionality. It has been compared to LEGOs because of this, and because the quality of modules vary, it's a good idea to do full code reviews of any modules that are selected for inclusion into any platform build. If an existing module already does mostly what is needed, it should be reviewed to make sure static variable caching is utilized, queries are optimized and general coding standards are being used.

Regularly contribute patches back to modules when a module is found lacking in any of these areas or if any general bugs are found through the module's issue queue, which can be found on the same page where you download the module. Performance reviews also are a good idea once a site is built to ensure that queries are optimized and not run more than once per page load. The Devel module is a great resource for this, as it will give you stats on page load times, memory usage and can display every query executed on any given page load.

Beyond the regular LAMP configuration optimizations, caching techniques, and hardware infrastructure are some general Web development best practices available within Drupal that not only can reduce loads on various servers, but also make it easy to have some of your data structures in code that can be version-controlled to keep track of changes and to help with the deployment process of said changes. The first, and relatively new, paradigm of "exportables" is twofold, in that it gives you a way to read a data structure from code instead of the database, and it also can be deployed to different environments and reused.

Exportables started with the Views module by Earl (merlinofchaos) Miles who wanted a way to help debug the problems that his module users might encounter. So, he created a way for users to export the view they created into a readable data structure that he then could put on his own machine to help him debug. This not only had the awesome side effect of being able to share these "view recipes" with other users, but it also evolved into a method where the structure could replace what was read from the database and help increase the performance. Exportables then was extrapolated into a library dubbed Ctools (for Chaos Tools) and used for the Panels module. Other people started catching on and implementing exportables for their modules, and now there are a whole slew of modules that use the Ctools Exportables for this purpose.

This eventually led to a module called Features that provides a UI to choose the various exportable data structures within a Drupal installation and wrap them up into a custom "feature" module, which then can be shared. These features can be simple configuration options or complex features requiring many other contributed modules in order to provide feature-rich enhancements for any Drupal Web site. Not only can it be used to share such features, but it also has become an important part of the deployment process in creating modern Drupal Web sites.

Another tool that has recently matured and become a necessity to any professional Drupalite is Drush. Drush stands for Drupal Shell and is a way to control your Drupal Web sites through the command line. Not only does it provide powerful commands to manipulate your Web site quickly, but other modules can provide integration with Drush as well, creating their own commands related to working with their particular module. For example: the Features module provides commands to Drush that allow you to list, update and revert any feature modules quickly that are part of a Drupal installation's codebase. The Backup and Migrate module provides integration to allow you to create SQL backups of your Web site quickly with a simple command. Some modules even provide commands to work with Drupal and Git! So, not only does Drush allow you to work with your Drupal site quickly, but you also don't have to load a huge page through Apache to do so.

And, of course, no professional Web site would be complete without revision control. Lullabot has used CVS (Concurrent Versions System), SVN (Subversion) and, most recently, made the move to Git. But no matter what you use, it's important to have a backup of your work and versioning for teams working on the same project. The merits of versioning your code are many. Working on a high-performance Web site usually takes many people, so version control becomes a necessity.■

Jerad Bitner has been using Drupal since the nightmare upgrades from 4.6 to 4.7 (that's early 2005, if you're asking). He started out as a Technical Illustrator with C/S Group and worked for three years with Photoshop, Illustrator, AutoCad and Macromedia products as well as PHP. When it came time to replicate a platform across the different locations of the company, Jerad found Drupal and hasn't looked back since.

Nate Haug adds a dash of design to Lullabot. He received degrees in both a Fine Arts and Computer Science from Truman State University, creating the perfect bridge between the technical and aesthetic. Detail is his obsession, so if you know what you want, Nate will deliver your desire.

Doing IT the App Engine Way

Using Google's App Engine, you can develop Web applications in Python or Java and deploy them on Google's infrastructure for free—until you hit five million page views per month.

Paul Barry

Google first announced its App Engine technology in 2008. The Internet's search superstar will host your Web application (webapp) on its infrastructure, initially at no charge. Only when your webapp gets "busy" will Google start charging you. By "busy", Google means something in the range of five million "page views" per month. Hit that threshold, and Google will come looking for your credit-card details.

Apps for Google's App Engine are written in Python (with Java recently added to the mix). As most of you know, Python's creator (Guido van Rossum) works for Google and spends a reported 50% of his time working on Python's ongoing improvement and the developer ecosystem that surrounds this increasingly popular general-purpose programming and scripting technology.

Unlike most other webapp development technologies and frameworks that require you to host your webapp yourself (or find a friendly cost-effective ISP to host your webapp and its dependent technologies for you), Google's App Engine abstracts away the hosting part. Simply build your webapp to the App Engine standard, upload it to Google, and it's then deployed in the Google "cloud". Google handles backups, load balancing, spikes in access, deployment, caches and the like. All you have to worry about is your code, as there's no more deployment distractions. And, when it comes to App Engine, it's all about the code, which is just how we programmers like things, isn't it?

In this article, I explain how to build a very simple App Engine project using the Python API. By the time you've worked through to the end, you should know enough to be in a better position to decide if Google's App Engine is something you want to spend more time learning.

App Engine and the MVC Pattern

Most modern webapp technologies are organized around the Model-View-Controller pattern (MVC). In essence, an MVC-compliant webapp is divided into three parts: the model handles interactions with your webapp's data, the view looks after the user interface, and the controller provides the application or business logic that glues everything together. In theory, changes to how your webapp looks (the view) should not significantly affect how your webapp stores its data (the model), or at least any effects should be minimal and localized. Of course, the MVC pattern can be applied to any application domain, but it is particularly appropriate within the Web context, as each component is (typically) physically separate from the others. For instance, your view code runs within a browser on your client PC. Your controller code runs on your Web server, and your model may be deployed on some sort of datastore, which may or may not run on its own hardware.

App Engine conforms to the MVC pattern, and what's nice about App Engine are that each of the MVC components are realized in code files that are easy to keep separate from one another. You can put your code into one big source code file if you like, but to be honest, all but the most trivial webapps benefit from splitting out each MVC component into its own source file.

Of course, App Engine doesn't really care how you structure your code (only that it's correct), nor does App Engine care how your webapp operates. If you want to write a basic CGI webapp, you can. You can use Python's WSGI standard too, which is the standard we'll use here. So, let's learn what's involved by building a simple MVC-patterned webapp with Google App Engine.

Tools

To build an App Engine webapp, you need two things: Google's App Engine SDK and release 2.5 of Python.

Although you'll deploy your App Engine webapp to Google's cloud, during development, it is possible to test your webapp on your local machine. All you need is a copy of the App Engine SDK, which is easy to get. Go to the App Engine download page (see Resources) and click on the link for the Python SDK. From the page displayed, select the Linux download (which is a ZIP file at the time of this writing). The current App Engine documentation as well as a Google Plugin for Eclipse also are available for download (should such things interest you).

With the download complete, installation is a breeze. Simply unzip the downloaded file into a directory of your choosing. I unzipped into my HOME directory, and a new directory called google_appengine was created.

Why Python 2.5?

Since the end of 2008, Python comes in three distinct flavors: the standard 2.5 release (which maintains backward-compatibility with the installed base), the 2.6 release (which is designed to bridge the move from the 2.5 release to future releases) and release 3 (which is the new, backward-incompatible Python). App Engine targets a customized and optimized version of the 2.5 release of Python that runs in Google's cloud. You can try to run your webapp under 2.6, but my experience has been less than satisfactory, although it can sometimes work. The 3.1 release of Python currently is not supported, so don't waste your time trying to use it with App Engine.

My Development Platform: Fedora 12

For this article, I'm using Fedora 12. I used to eat, sleep and breathe Red Hat Linux (then Fedora), but like a lot of other Linux users, I was tempted away by the promise and then delivery of a friendlier desktop experience with Ubuntu. As luck would have it, I recently received the latest edition of Mark G. Sobell's book (see Resources), which includes Fedora 12 on DVD. Installation, as expected, was straightforward. However, Fedora 12 comes with Python 2.6 pre-installed, and I needed the 2.5 release.

A feature of Python that I love is that multiple versions of the interpreter can happily co-exist on your system. On my systems (desktops and laptops), I have releases 2.5, 2.6 and 3.1 installed. Only one of these releases is symbolically linked to the /usr/bin/python (usually the 2.6 release), but I can invoke the other releases using one of these command lines:

```
python3
python2.5
```

Deploying release 2.5 of Python on Fedora wasn't an issue. It's the usual `tar -zxvf`, `configure`, `make` and `make-install` four-step. As expected, the Fedora DVD pre-installed all the development tools required to let me build Python 2.5 from source without any issues.

Configuration

Unlike technologies, such as Ruby on Rails or Django, which provide a collection of helper scripts to get you up and running quickly, App Engine forces you to do all the work yourself. Thankfully, this is not a huge effort. To demonstrate, let's create a new project with the rather imaginative name, myapp. Create the initial directories and files required in your HOME directory with these commands:

```
mkdir myapp && cd myapp
mkdir templates && touch app.yaml
```

The above commands create some required directories (more on these later) as well as the main App Engine configuration file: app.yaml. You edit this file to tell App Engine all about your webapp. Add the following configuration directives to your YAML file:

```
application: myapp
version: 1
runtime: python
api-version: 1

handlers:
- url: /.*
  script: myapp.py
```

The `application` line identifies your webapp, and the value needs to match the name of the directory you just created to house your project. Use the `version` value to indicate the release of your webapp to which this YAML file refers (this value also is used by Google's cloud to refer to different versions of your webapp, should they exist). The `runtime` line tells the App Engine for which platform you are coding, and the `api-version` value indicates what version of the API you are using.

The remaining three lines tell App Engine what to do with any Web requests destined for your webapp. It is useful to think of these handlers in the YAML file as high-level, application "routing directives". The bit after `url` is a regular expression that (as all regex gurus will tell you) matches anything that starts with a /, followed by any string (or nothing at all). What's happening here is that any URL received by App Engine on behalf of your webapp is going to be redirected to the script identified on the `script>` line, which in this case is called myapp.py. At the moment, no such script exists, so let's fix that.

Writing a Request-Handler Script

To demonstrate how an App Engine webapp is put together, let's build a simple Web page that lets users submit their e-mail addresses with a message. These two pieces of data are stored in the Google cloud. A second page displays the e-mail addresses and messages on-demand. Granted, this isn't a hugely exciting webapp, but it's enough to demonstrate the basics of the technology and for you to get started with something "real". Of course, it is possible to build this using App Engine's CGI mechanism (which works exactly as you would expect), but as this application is destined for greatness, let's code to Python's WSGI standard instead. Let's also build the webapp to conform to the MVC pattern.

Defining the Model

As you plan to store some data in this webapp, you need somewhere to put it, which means you need a model. App Engine provides an API to Google's "cloud" datastore. All you need to do is define a Python class that inherits from db.Model, then create the required data fields. For the webapp, you need a field for the e-mail address and the associated message. To keep things manageable, let's put your model code in its own file, called myappDB.py:

```
from google.appengine.ext import db

class UserComment(db.Model):
    cust_email   = db.StringProperty()
    cust_message = db.TextProperty()
```

There's not much to this model code. It simply imports the db module from App Engine, creates a new class called UserComment and creates class instance variables for each data field. As App Engine's StringProperty type is limited to 500 bytes, you need to specify TextProperty for the user message, just in case someone has a lot to say.

Defining Your Controller Code

With the model defined, you can create the rest of your code. Create a file called myapp.py and pop the following code into it. Note that the code in this section is all contained in one file, but it's split up here so I can describe its function to you. Start with your imports:

```
import wsgiref.handlers

from google.appengine.ext import webapp
from google.appengine.ext import db
from google.appengine.ext.webapp import template

import myappDB
```

After importing the Python-standard WSGI reference implementation, three libraries are imported from App Engine: `webapp` provides a simple Web framework, `db` provides access to the App Engine datastore, and `template` provides access to App Engine's standard templating system (which is based on and built from Django's). Note that you've also imported the just-created myappDB module, which brings your model definitions into this program.

Every webapp needs to be told what to do when a user sends a default request from a browser to a server. Typically, this is requesting the index or home page. The first request handler provides that functionality:

```
class IndexHandler(webapp.RequestHandler):
    def get(self):
        html = template.render(
            'templates/index.html', {}
        )
        self.response.out.write(html)
```

You've created a new class called `IndexHandler` that's inherited from webapp's `RequestHandler`. Within the class, a Python method called `get` is invoked whenever a request for the default page is processed. The method renders a template called index.html within the templates directory and assigns the rendered page to the html variable, which then is sent to standard output as the HTTP response, which eventually makes its way to the browser. The second parameter to the render() method is an empty hash (or "dictionary" to use the correct Python terminology). It's possible to send data (template variables) to the template engine, but you don't have to in this instance. The convention is to send an empty hash when there's no data for the template engine to process. I explain how to create templates later in this article. For now, note that any HTML used by your webapp is stored in the templates directory, which helps segregate view code from controller code.

The functionality required to leave a message has two parts. The first presents a small form that allows users to enter their e-mail addresses and messages. The second takes the submitted form data and stores it in the Google cloud. Here's a request handler class that implements both parts:

```
class LeaveCommentHandler(webapp.RequestHandler):
    def get(self):
        html = template.render(
            'templates/comment.html', {}
        )
        self.response.out.write(html)

    def post(self):
        comment = myappDB.UserComment(
            cust_email = self.request.get("c_email"),
            cust_message = self.request.get("c_message")
        )
        comment.put()
        self.redirect("/comments")
```

The `get` method in this class is essentially identical to the `get` method in the previous class, except in this method, you are rendering a different template called comment.html. The `post` method (which you didn't have in the last class) responds to a POST request sent from a browser. In other words, when users fill in the form rendered by the `get` method and then click the submit button, the data on the form is delivered to this request handler's `post` method. The code in your `post` method first creates a new instance of the model data by extracting two named form fields from the posted data (referred to as c_email and c_message in the HTML form). The submitted data is assigned to the data fields defined in the model and then stored in Google's cloud with a call to `comment.put()`. With that done, your code immediately redirects to a different URL (/comments), which causes another request handler's code to activate. That is, of course, assuming you have the "/comments" handler code written. And, here it is:

```
class DisplayCommentsHandler(webapp.RequestHandler):
    def get(self):
        comments_query = myappDB.UserComment.all()
        comments = comments_query.fetch(1000)
        html = template.render(
            'templates/comments.html',
            {'comments': comments}
        )
        self.response.out.write(html)
```

Your `DisplayCommentsHandler` code provides only GET functionality, as that's all that's required. Using App Engine's functional interface to the datastore, your `get` method first creates a query that asks for all the `UserComment` data, before fetching the first 1,000 comment-pairs from the query results. Your code then renders the comments.html template, passing in the (up to 1,000) comment-pairs to the templating engine. The rendered HTML returned from the template system then is sent to standard output as the HTTP response.

The limit of 1,000 is imposed by App Engine on all running webapps and is designed to limit the potential harm a "rogue" webapp could do to the App Engine infrastructure if left unchecked. By limiting the number of rows of data that can be fetched at once, App Engine can attempt to ensure that no one webapp hogs all of its resources. It's not really much of restriction. How many Web pages attempt to display more than a few hundred database records at a time? Obviously, if you have more than 1,000 rows in your datastore, you need to write some extra code to cycle through your data, 1,000 rows at a time, until you've exhausted it all.

Unlike the previous two request handlers, this latest one sends data to the templating system. The comment query results (a collection of e-mail addresses and messages) are passed to the template for further processing.

You may have noticed that there's no SQL used in the `get` method within the `DisplayCommentsHandler` request handler. Instead, you've used App Engine's API to request all the data from the datastore, from which you've then fetched 1,000 rows of data. Google's datastore technology, an integral component of App Engine, doesn't support SQL. It turns out that the datastore is not a relational database. Instead, it's based on Google's BigTable technology, which is a different beast altogether. Google does provide a query language of sorts in the guise of GQL, which looks a lot like SQL but doesn't do everything you are used to being able to do with SQL. For instance, there are no joins in GQL. Check out the App Engine datastore documentation for all the details to see if no SQL will be a deal-breaker for your app.

With the request handlers defined, all that's left to do is connect your application's URLs to the request handlers. Recall that the app.yaml file already has arranged for every URL request directed to your App Engine webapp to be handled by your myapp.py program. So, the question is, when the request gets to your program, what happens next? The answer is that your myapp.py program handles it, of course! When you imported webapp, you inherited functionality that allows you to link your URLs with your request handlers. Here's the code you need:

```
def main():
    app = webapp.WSGIApplication(
        [ ('/comment', LeaveCommentHandler),
          ('/comments', DisplayCommentsHandler),
          ('/.*', IndexHandler)],
          debug = True
    )
    wsgiref.handlers.CGIHandler().run(app)
```

The `WSGIApplication` constructor takes a list of tuples detailing which URL invokes which request handler. Each of the URLs in the tuple list are regex patterns that are checked for a match to the URL that has been delivered to your program. If a match is found, the request handler is invoked. Note that in this code, you need a "catchall" regex to do something sensible when a delivered URL does not match one of the URL patterns. Here, you arrange to invoke the `IndexHandler` when no match is found. You've also switched debugging information on, which is useful during development, but it should be switched off when you deploy. With the URL routes ready, a call to the `run` method provided by `wsgiref.handlers.CGIHandler` starts your webapp.

The URL-routing and application-starting code is contained within the `main` function. This is deliberate, as App Engine looks for this function when initially loading and reloading your webapp. Its existence allows App Engine to optimize and cache your webapp. It also allows App Engine to avoid having to load your application every time a new request occurs, a criticism that continues to haunt CGI to this day. Of course, to test your webapp locally, you need to tell Python how to start it, which is accomplished with the classic Python idiom:

```
if __name__ == '__main__':
    main()
```

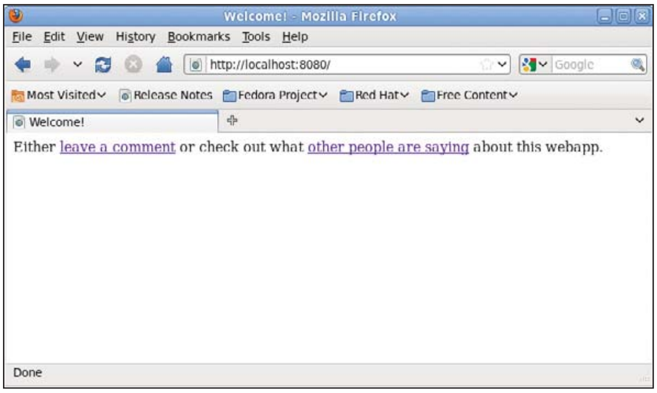
And, that's it for your controller code. You've created code that does what you require when a URL request is sent to your webapp. All that's left is to create your view code, which, in this simple app, is a small set of HTML templates.

Defining Your Views

The first template is rendered when no URL or the default URL is matched during the routing process. It's rendered by `IndexHandler`, is stored in your templates directory, is called index.html and looks like this:

```
<html>
<head><title>Welcome!</title></head>

<body>
Either <a href="/comment">leave a comment</a>
or check out what
<a href="/comments">other people are saying</a>
about this webapp.
</body>
</html>
```

As you may notice, the index.html template is plain HTML. It displays a welcome page that offers your site's users the option to leave a new message or see all the messages already submitted (Figure 1). Granted, this HTML is trivial, but critically, it is stored separately from your code, which maintains the goal of the MVC pattern.



Either leave a comment or check out what other people are saying about this webapp.

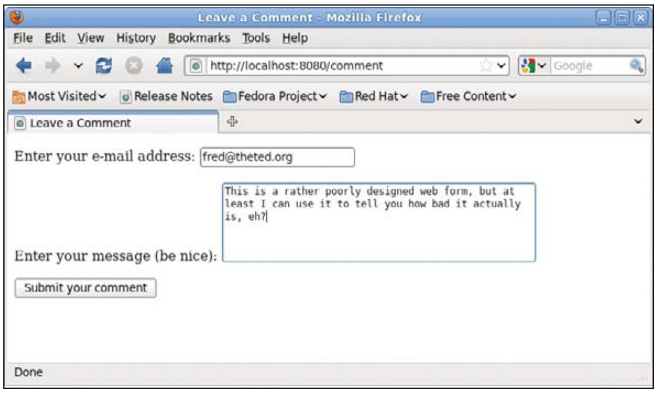
Figure 1. The Index Request Handler Page

When users click the leave a comment link, a request routes to `LeaveCommentHandler`'s `get` method, which renders another template called comment.html, which, again, resides in the templates directory. Here's this template's HTML:

```
<html>
<head><title>Leave a Comment</title></head>

<body>
<form action="/comment" method="POST">
<p>Enter your e-mail address:
<input type="text" name="c_email"></p>

<p>Enter your message (be nice):
<textarea name="c_message" rows="5" cols="50"></textarea></p>

<p><input type="Submit" value="Submit your comment"></p>
</body>
</html>
```

Again, this is standard HTML. A form is rendered to the screen (Figure 2), and the form solicits an e-mail address and message from the user. Note the names assigned to each interface element in the HTML. Also, note that the form's action tag is directed back to the `/comment` URL. When users submit the form, the data is posted to App Engine, which results in the `post` method executing within your `LeaveCommentHandler` request handler. This code creates a new row of data from the submitted data (note how the names match) and saves it to the datastore. The code then redirects to the `/comments` URL.

When App Engine sees the `/comments` request URL, it invokes

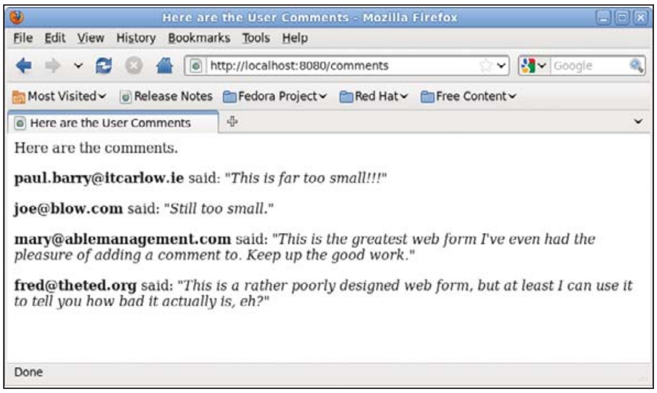

Leave a Comment - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:8080/comment
Google
Most Visited Release Notes Fedora Project Red Hat Free Content
Leave a Comment
Done

Enter your e-mail address: fred@theted.org

Enter your message (be nice): This is a rather poorly designed web form, but at least I can use it to tell you how bad it actually is, eh?

Submit your comment

Figure 2. The Leave a Comment Form


Here are the User Comments - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:8080/comments
Google
Most Visited Release Notes Fedora Project Red Hat Free Content
Here are the User Comments
Done

Here are the comments.

paul.barry@itcarlow.ie said: *"This is far too small!!!"*

joe@blow.com said: *"Still too small."*

mary@ablemanagement.com said: *"This is the greatest web form I've even had the pleasure of adding a comment to. Keep up the good work."*

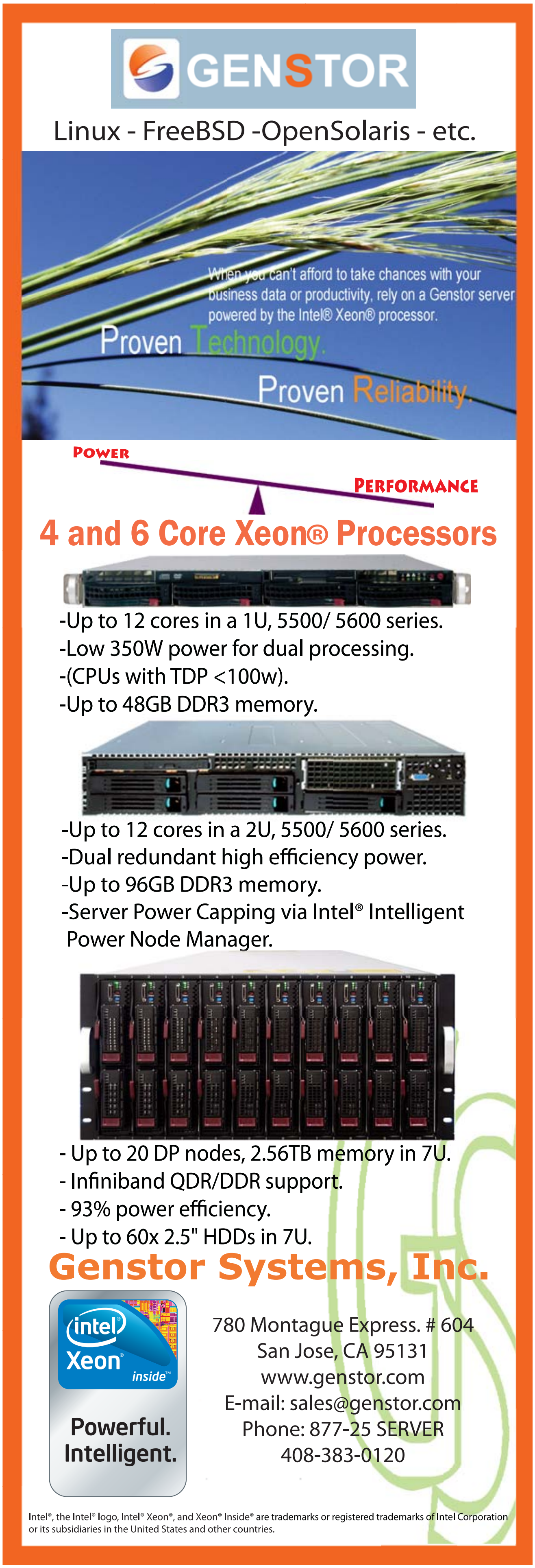
fred@theted.org said: *"This is a rather poorly designed web form, but at least I can use it to tell you how bad it actually is, eh?"*

Figure 3. The Displayed List of Comments

the `DisplayCommentsHandler` request handler's `get` method, which fetches 1,000 rows of data from the datastore and then sends the data to the comments.html template for rendering. This final template looks a bit more like a "real" template:

```
<html>
<head><title>Here are the User Comments</title></head>

<body>
<p>Here are the comments.</p>

{% for c in comments %}
    <p><b>{{ c.cust_email }}</b> said:
    <i>"{{ c.cust_message }}"</i></p>
{% endfor %}

</body>
</html>
```

This is the final template that contains templating instructions, which are included to process the data that the request handler sent to the template engine when the template was invoked in your controller code. Anything enclosed within {% and %} and within {{ and }} is template code; everything else is standard HTML. App Engine's templating technology is based on Django's, which is a popular Python-based Web application framework. Code found within {% and %} is executed, whereas code found within {{ and



}} is a value substitution. This template takes the comment query results passed to the template and displays each row within some custom HTML. The rendered results are shown in Figure 3.

Testing Your webapp

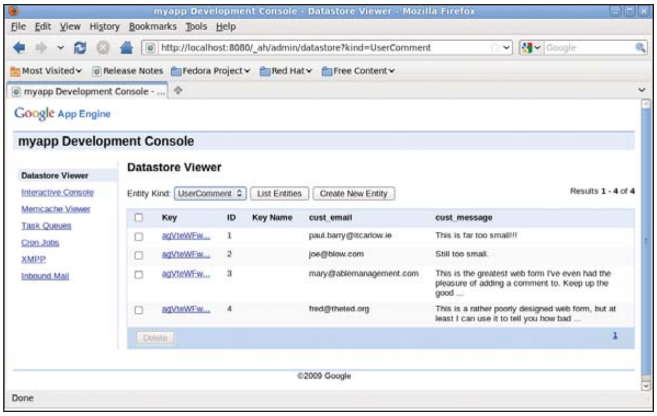
Testing your application locally is complicated (only slightly) by having to ensure that you use the correct version of Python, namely the 2.5 release. If you have been following along, you should have App Engine installed in its own directory in your HOME directory, as well as your webapp code in a directory called myapp. To start your webapp from your HOME directory, open a shell and use this command:

```
python2.5 google_appengine/dev_appserver.py myapp/
```

A bunch of status messages will appear in the shell, and if all is well, the development server will inform you that your webapp is up and running on http://localhost:8080. Fire up your favorite browser, surf to your webapp and give it a spin. You should see behaviors similar to those shown in Figures 1, 2 and 3.

The Admin Interface

App Engine includes an administrator interface to your webapp that can be accessed via the Web. To see it, point your browser at http://localhost:8080/_ah/admin. Figure 4 shows the interface to the datastore entries I created. The interface lets you inspect and edit each of your entries in the datastore (as well as create new ones). Each entry has been assigned a unique Key and ID value by the datastore automatically. These values are often used to retrieve a specific entry from the datastore.



http://localhost:8080/_ah/admin/datastore?kind=UserComment

myapp Development Console

Datastore Viewer

Entity Kind: UserComment

Results 1 - 4 of 4

□	Key	ID	Key Name	cust_email	cust_message
□	agVteWFw...	1		paul.barry@itcarlow.ie	This is far too small!!!
□	agVteWFw...	2		joe@blow.com	Still too small.
□	agVteWFw...	3		mary@ablemanagement.com	This is the greatest web form I've even had the pleasure of adding a comment to. Keep up the good ...
□	agVteWFw...	4		fred@theted.org	This is a rather poorly designed web form, but at least I can use it to tell you how bad ...

Figure 4. The App Engine Admin Interface

Deployment

With your testing complete, you're ready for deployment. To do this, follow the deployment instructions on the App Engine Web site. This involves signing up for a Google ID (you already have one if you use Gmail or Wave), selecting a unique name for your webapp and requesting a seven-digit Google App Engine Code (which you need to activate your webapp and which is sent by SMS to your cell phone). With all of that in place, upload your code to Google's cloud from your HOME directory using this command:

```
google_appengin/appcfg.py update myapp/
```

Learning More

Of course, it doesn't end there. App Engine has so much more, including integration with Google's user management and login system, security enhancements, memcached integration and validation technologies, among other things. I recommend reading *Using Google App Engine* and *Programming Google App Engine*, both from O'Reilly Media (see Resources). The former is an extended tutorial introduction to App Engine using Python, and the latter is a reference that targets both the Python and Java APIs. At the time of this writing, the other technical publishers have App Engine books at an advanced stage of development (most notably Manning). Apress also has a series of Google books. Another project worth keeping an eye out for is the upcoming Google App Engine video tutorials (again) from O'Reilly Media.

Is App Engine Really Free?

As I mentioned earlier, Google lets you get started with App Engine for free. When your site becomes popular, Google asks you to pay for the hosting services it provides. The busier your site, the more you pay, and costs are pretty much in line with what you'd expect from a reasonable-size ISP. If your site traffic remains modest, you may never have to pay for App Engine's hosting service. But, do you pay in other ways? Consider the following: once your code is uploaded to App Engine, you can't retrieve it. You *can* update it, but you had better keep a local copy as your own backup should you wish to transfer the business logic you've embedded in your webapp to another platform. Then, there's your data. It lives in the Google cloud, and what that means really depends on whom you ask. App Engine keeps your data away from others, but you are trusting Google to mind it for you.

App Engine is built on top of open-source Linux, with Python and Java APIs, which also are both open technologies. But, these facts alone do not make App Engine open. Far from it, this is as vertically closed a system as Apple's iPad. Be aware of what you are giving up when you decide to develop for this particular "free" Google platform. If you're okay with vendor lock-in, and if you trust Google with your data and your application, Google App Engine may be for you.■

Paul Barry (paul.barry@itcarlow.ie) lectures at The Institute of Technology, Carlow in Ireland. He recently completed *Head First Programming*, which he cowrote with David Griffiths. As he's a sucker for punishment, he's now working on *Head First Python*, to be published by O'Reilly Media in late 2010.

Resources

The Google App Engine Download Page: **code.google.com/appengine/downloads.html**

A Practical Guide to Fedora and Red Hat Enterprise Linux, 5th ed., by Mark G. Sobell, Prentice-Hall, PTR, 2010: **www.pearsonhighered.com/educator/product/Practical-Guide-to-Fedora-and-Red-Hat-Enterprise-Linux-A/9780137060887.page**

Using Google App Engine, by Charles Severance, O'Reilly Media, 2009: **oreilly.com/catalog/9780596801601**

Programming Google App Engine, by Dan Sanderson, O'Reilly Media, 2009: **oreilly.com/catalog/9780596522735**

Developing ***Portable Mobile Web Applications***

What if you could write **iPhone and iPad applications** on Linux? And, what if those applications also ran on **Android phones**? Well, you can, by writing your application as a portable mobile Web application. Take advantage of HTML5 and JavaScript libraries to write rich Web applications that users can't tell from native applications. You even can have them install from iTunes and Android Market. Plus, you can write them in a fraction of the time it takes to write a native application.

Rick Rogers

Mobile applications for iPhone and Android smartphones are where the excitement is today in application development. There are plenty of customers (Canalys says there were almost 8 million Android phones and more than 25 million iPhones sold in 2009), and those users regularly load applications on their phones (AdMob says Android and iPhone users average around nine downloads a month, and iTouch users average around 12). The demand for mobile applications is hot.

How can a Linux developer tackle this market? Native applications for the iPhone must be developed with the Apple iPhone SDK, which runs only on Mac OS X. Android development is supported on Linux with the Android SDK, but ideally, you'd like to develop apps on Linux that run on both iPhone and Android.

Mobile Web applications provide a solution. Web applications use the browser as a common runtime environment across different platforms. Applications are written in HTML, JavaScript and CSS, and run on the platform's native browser. This idea is not new, but historically, there have been a few issues with Web applications:

- Browser security prevented the storage of local data.
- Platform features like geolocation were not accessible from HTML/JavaScript.

- Users had to start the browser to use a Web app, which then didn't match the native UI.

- Browsers themselves were fragmented—different browsers interpreted JavaScript in different ways.

These problems are being addressed, and fortunately, Android and iPhone both selected WebKit as the layout engine for their respective browsers. HTML5 extends what a Web application can do, and creative people have devised JavaScript libraries that minimize the look-and-feel issues. The solution is an ongoing process, and it isn't complete.

HTML5 and related specifications add features like canvas, video, local storage, Web workers, off-line applications and geolocation to HTML, and WebKit is rapidly integrating these into the layout engine.

Specialized runtimes and libraries exist to allow JavaScript access to underlying phone functions, such as location, acceleration, sound, contacts, battery, camera, telephony and calendar. Some of these (for example JIL, BONDI and WAC) are industry-led, involve custom Web runtimes and aim to provide a universal widget environment across platforms. Others (such as PhoneGap, Titanium and Rhomobile) focus on iPhone and Android, with ties to the native SDKs that extend the capabilities of Web apps. As HTML5 implements similar features, these libraries generally conform their APIs to the HTML5 APIs.

JavaScript libraries have been developed to address the look-and-feel problem, including:

- iUi: a small extensible library that mimics the iPhone user interface.

- iWebKit: another framework for iPhone-style applications.

- jQTouch: a plugin for the popular jQuery JavaScript library, which provides an iPhone style and a more general jQTouch style. jQTouch has the advantage of using jQuery to hide browser differences.

Android and iPhone both use WebKit as their layout engine, but there are still differences, partly due to the selection of different WebKit releases:

- Android 1.6 (HTC G1) uses WebKit 525.20 and implements only Canvas, Canvas Text and Geolocation.

- Android 2.1 (Motorola Droid) uses WebKit 530.17 and adds the rest of HTML5 (video, audio, local storage, Web Workers and off-line applications).

- iPhone 3GS and iTouch use WebKit 528.16 and .18 and include all features except for Web Workers (multithreading).

- iPad uses WebKit 531.21 and includes all features.

So What's the Catch?

Developing Web applications for smartphones is pretty straightforward, but there are some things you need to know:

- It's not like C: if you're used to developing Linux applications in C or C++ or Java or Perl, this is different. Web application development is a little closer to the Android development environment, where screen layouts are in XML files, and functionality is written in Java, but mostly, it's like Web development.

- Native apps require native SDKs: if you want users to be able to load your application from iTunes and Android Market, you have to enable that with the appropriate SDKs. There are workarounds that I talk about later.

Development Tools and Libraries

All you need to develop Web applications is a text editor to write JavaScript, CSS and HTML, and a browser to test the results. The job is a bit easier using a Web-oriented IDE, a JavaScript debugger and the Safari browser, along with an assortment of mobile devices for testing. Safari has a number of features that simplify development. You can select which User Agent the browser emulates (from the Developer menu), and the Web Inspector

allows you to inspect and debug Web elements, including client-side databases. Safari isn't supported on Linux, but it runs just fine under VirtualBox. The code for this article was developed using the following:

- Ubuntu 9.10 and the gedit editor.
- Safari browser on Windows XP on VirtualBox.
- Apache httpd Web server.
- GIMP for icon graphics.
- jQTouch and jQuery libraries.
- iPhone, iTouch, iPad and Android devices.

Installation of the tools is well documented elsewhere. The Resources section for this article gives pointers to the download URLs. To install jQTouch, just put the JavaScript and CSS files in the directory tree of your Web application, and point to them from your HTML <head> element. jQTouch comes with minimized versions of the files and a minimized jQuery library.

Example Application

As an example, let's look at a simple notes application that I'll call Webnotes. With it, a user could write notes, and view, edit or delete notes later. A note will consist of a title and an arbitrary-length string. The notes will be stored locally on the smartphone, using HTML5 client-side database APIs, and we'll test it running on a variety of Apple and Android devices. When we're done, we'll compare this to a similar Android sample application that ships with the Android SDK. Because we're using client-side database features that are part of HTML5, we'd expect it to work fine on the iPhone, the Droid and the iPad, and not to work on the HTC G1 (it does not support local storage).

Our app has three screens:

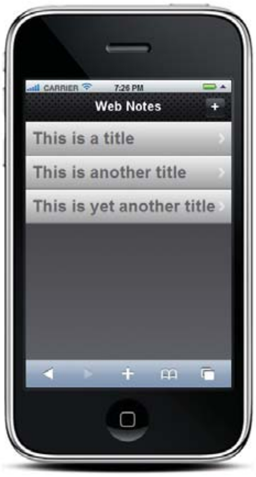
- The opening screen will display a list of existing notes, listed by title, in order by the date they were last edited. Touching a title will select that note for edit. Touching a + button will add a new note (Figure 1).
- An edit screen will allow viewing, editing or deletion of a note. (Figure 2).
- An add screen will create a new note and store it in the database (Figure 3).

Listing 1. index.html

```
<!DOCTYPE HTML PUBLIC>
 <head>
  <title>WebNotes</title>
  <link type="text/css" rel="stylesheet"
        media="screen" href="jqtouch/jqtouch.min.css">
  <link type="text/css" rel="stylesheet"
        media="screen" href="themes/jqt/theme.min.css">
  <script type="text/javascript"
          src="jqtouch/jquery.1.3.2.min.js"></script>
  <script type="text/javascript"
          src="jqtouch/jqtouch.min.js"></script>
  <script type="text/javascript"
          src="javascript/webnotes.js"></script>
 </head>
 <body>
  <div id="home">
   <div class="toolbar">
    <h1>Web Notes</h1>
    <a class="button add slideup"
       href="#addNote">+</a>
   </div>
   <ul class="metal">
    <li id="noteTemplate" class="arrow"
        style="display:none">
     <span class="title">Title</span>
    </li>
   </ul>
  </div>

  <div id="addNote">
   <div class="toolbar">
    <h1>Add Note</h1>
    <a class="button cancel" href="#">Cancel</a>
   </div>
   <form method="post">
    <ul>
     <li><input type="text" class="title" /></li>
     <li><textarea class="note" ></textarea></li>
     <li>
      <input type="submit" class="submit"
             name="action" value="Save Note" /></li>
    </ul>
   </form>
  </div>

  <div id="editNote">
   <div class="toolbar">
    <h1>Edit Note</h1>
    <a class="button cancel" href="#">Cancel</a>
    <a class="button"
       onclick="deleteNoteById()">Delete</a>
   </div>
   <form method="post">
    <ul>
     <li><input type="text" class="title" /></li>
     <li><textarea class="note" ></textarea></li>
     <li>
      <input type="submit" class="submit"
             name="action" value="Save Note" /></li>
    </ul>
   </form>
  </div>
 </body>
</html>
```

Listing 2. webnotes.js

```
var jQT = $.jQTouch({
   icon: 'icon.png',
});
var db;
var currId;
$(document).ready(function(){
 $('#addNote form').submit(addNote);
 $('#editNote form').submit(replaceNoteById);
 db = openDatabase('WebNotes', '1.0', 'WebNotes',
                  524288);
 db.transaction(
  function(transaction) {
   transaction.executeSql(
    'CREATE TABLE IF NOT EXISTS notes ' +
    ' (id INTEGER NOT NULL PRIMARY KEY ' +
    '     AUTOINCREMENT, ' +
    ' date DATE NOT NULL, title TEXT NOT NULL, ' +
    ' note TEXT NOT NULL);'
   );
  }
 );
 refreshNotes();
});

function addNote(){
 var now = new Date();
 var title = $('#addNote .title').val();
 var note = $('#addNote .note').val();
 db.transaction(
  function(transaction) {
   transaction.executeSql(
    'INSERT INTO notes (date, title, note) VALUES' +
    ' (?,?,?)', [now, title, note],
    function(){
     $('#addNote .title').attr('value', "");
     $('#addNote .note').text("");
     refreshNotes();
     jQT.goBack();
    },
    errorHandler);
  }
 );
 return false;
}

function errorHandler(transaction, err){
 alert('SQL err: '+err.message+' ('+err.code+')');
 return true;
}

function refreshNotes() {
 $('#home ul li:gt(0)').remove();
 db.transaction(
  function(transaction) {
   transaction.executeSql(
    'SELECT * from notes ORDER BY date;', null,
    function(transaction, result) {
     for (var i=0; i < result.rows.length; i++) {
      var row = result.rows.item(i);
      var newNote = $('#noteTemplate').clone();
      newNote.removeAttr('id');
      newNote.removeAttr('style');
      newNote.data('noteId', row.id);
      newNote.appendTo('#home ul');
      newNote.find('.title').text(row.title);
      newNote.click(function(){
       editNoteById($(this).data('noteId'));
       jQT.goTo('#editNote', 'swap');
      });
     }
    },
    errorHandler);
  }
 );
}

function replaceNoteById() {
 db.transaction(
  function(transaction) {
   transaction.executeSql(
    'UPDATE notes SET title=?, note=? WHERE id=?',
     [$('#editNote .title').val(),
      $('#editNote .note').val(),
      currId],
    function(transaction, result) {
     refreshNotes();
     jQT.goTo('#home', 'swap');
    },
    errorHandler);
  }
 );
}

function editNoteById(id) {
 db.transaction(
  function(transaction) {
   transaction.executeSql(
    'SELECT * from notes WHERE id=?;', [id],
    function(transaction, result) {
     var res = result.rows.item(0);
     currId = res.id;
     $('#editNote .title').attr('value', res.title);
     $('#editNote .note').text(res.note);
    });
  }, errorHandler);
}

function deleteNoteById() {
 db.transaction(
  function(transaction) {
   transaction.executeSql(
    'DELETE FROM notes WHERE id=?;', [currId],
    function(transaction, result) {
     alert('Note deleted');
     refreshNotes();
     jQT.goBack();
    }, errorHandler);
  }
 );
}
```

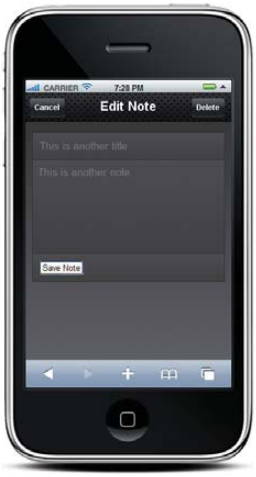


This is a title

This is another title

This is yet another title

Figure 1. Home Screen



This is another title

This is another note

Figure 2. Edit Screen

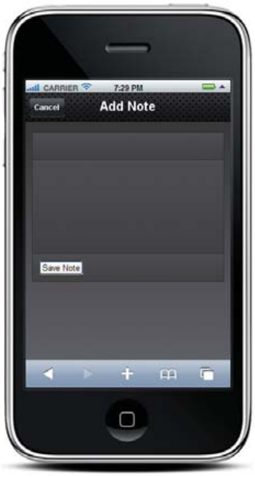


Figure 3. Add Screen

Listing 1 is the HTML file, index.html, primarily concerned with layout. Listing 2 is a JavaScript file, webnotes.js, that has the logic we need. Let's go through the HTML first.

After the HTML declaration, there is the <head> section of the document. The <title> element is the HTML title for the page. The iPhone and Android browsers display it as the window title until we make the application full screen. The next two <link> elements tell the browser where to find CSS files referenced in the application. Two styles come with jQTouch: "/themes/apple" and "themes/jqt". We've chosen the latter here, to be a little more device-independent. The next three elements are <script> references—the first two for jQuery and jQTouch, and the next for webnotes.js. The order is important—the <script> element for jQuery must precede the one for jQTouch, and they must both precede any scripts that use them.

After the header, the <body> element of Listing 1 contains three first-level <div> elements, one for each screen. The top levels of the three screens are very similar. Each has a unique id attribute that we use to refer to the screen from JavaScript. Each also includes an inner <div> with `class="toolbar"` that defines the bar at the top of the screen. The <h1> element in these <div>s is the screen title. The toolbars also each have an anchor element of `class="button cancel"`. This jQTouch class defines the arrow-shaped cancel button, and the href says clicking on it takes us to the "home" screen. The anchor also defines text ("Cancel") that appears in the button. On the "home" screen, we've included a + button to add a note. We've specified the "slideup" animation for that button's action, so the "addNote" screen will slide into view from the bottom of the screen.

The "home" screen also contains an inner <div> with a list containing one list item. That item is a template that defines the display of note titles. You'll see how we use it below.

In the "addNote" screen, after the "toolbar" <div> there is another <div> that contains a <form>. This <form> contains a list that has three list items:

- Text input for the title of the note.
- Text area for the contents of the note.
- Button to submit the <form>.

We've given the list items class names to make them easy to find with jQTouch.

The "editNote" screen looks like "addNote", with one addition. There's an <input> of `class="button"` in the toolbar to give the user a way to delete a note. The onclick attribute for this button tells the browser to call a JavaScript routine `deleteNoteById()`, which we define in webnotes.js.

Client-Side Database

WebKit uses SQLite to implement the client-side database APIs for HTML5. The implementation is remarkably complete, including support for transactions with rollback if the transaction does not succeed. Let's look at the JavaScript in Listing 2, webnotes.js.

The first four lines of the file initialize jQTouch and assign an instance to the variable jQT. The parameter ".icon" is one of many that can be defined for jQTouch. It points to a 57x57 pixel icon for the application (Figure 4).

On line five, we declare a variable "db" for the database instance. The block of code that starts `$(document).ready` is a jQuery function that executes when the browser has finished loading the DOM, even though the page contents may still be loading. The anonymous function first redirects the submit buttons from the "addNote" and "editNote" forms, pointing them at JavaScript functions. Then we use openDatabase() to do just that, passing it four parameters:

- Short name for the database.
- Database version number.
- Display name for the database.
- Maximum size of the database.

SQLite creates the database if it does not exist. The anonymous function executes an SQLite transaction that creates or opens a table called "notes". Each row of that table represents a note with the following columns:

Figure 4. Application Icon

- `INTEGER id KEY`: unique identifier auto-incremented by SQLite.
- `DATE lastedit`: date the note was last edited.
- `TEXT title`: title of the note.
- `TEXT note`: contents of the note.

Once the table is opened, we call a function, refreshNotes(), described below, that will update the list displayed on the "home" screen.

We next define the "addNote()" function, which gets invoked when the user touches Save Note on the "addNote" screen. The user already has entered the text for title and note, so we now want to insert a suitable record into the "notes" table. We get the date from the Date() function, and use jQuery to locate the title and note input elements. If you're not familiar with jQuery, it uses a CSS-like syntax to identify DOM elements. In this case, it finds the elements with classes "title" and "note", and the .val() function assigns their values to JavaScript variables "title" and "note", respectively. Using the client-side database API to start a transaction, transaction.executeSql() takes four parameters:

- SQL string: template for the SQL to be executed.
- Array of parameters whose values replace the ? marks in the SQL template.
- Function that executes if the operation was successful—in this case, an anonymous function.
- Function that executes if there was an error—in this case, errorHandler().

If successful, we clear out the values in the input elements (ready for the next add), refresh the list of notes so the new one will show up and use a jQTouch function, jQT.goBack() to return to the "home" screen. Since we used "slideup" to show the "addNote" screen, jQTouch is smart enough to slide it down to

Listing 3. webnotes.manifest

```
CACHE MANIFEST
index.html
icon.png
jqtouch/jqtouch.min.css
themes/jqt/theme.min.css
jqtouch/jquery.1.3.2.min.js
jqtouch/jqtouch.min.js
javascript/webnotes.js
```

Advertiser Index

CHECK OUT OUR BUYER'S GUIDE ON-LINE.
Go to www.linuxjournal.com/buyersguide where you can learn more about our advertisers or link directly to their Web sites.

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

Advertiser	Page #	Advertiser	Page #
1&1 Internet, Inc. www.oneandone.com	1	Linux Foundation www.linuxfoundation.org	75
Aberdeen, LLC www.aberdeeninc.com	19	Logic Supply, Inc. www.logicsupply.com	29
ApacheCon www.us.apachecon.com/us2008	59	Lullabot www.lullabot.com	51
Archie McPhee www.mcphee.com	78	Microway, Inc. www.microway.com	C2, C4
ASA Computers, Inc. www.asacomputers.com	55	Mikro Tik www.routerboard.com	7
Cari.net www.cari.net	61	Ohio Linux Fest www.ohiolinux.org	67
Codero www.codero.com	5	Polywell Computers, Inc. www.polywell.com	78, 79
Digi-Key Corporation www.digi-key.com	78	Saint Arnold Brewing Company www.saintarnold.com	78
Emac, Inc. www.emacinc.com	43	ServerBeach www.serverbeach.com	31
EmperorLinux www.emperorlinux.com	23	Servers Direct www.serversdirect.com	9
Gecad Technologies/Axigen www.axigen.com	79	SharePoint www.sptechcon.com	41
Genstor Systems, Inc. www.genstor.com	57	Silicon Mechanics www.siliconmechanics.com	17, 33
Gutsy Geeks www.gutsygeeks.com	45	Technologic Systems www.embeddedx86.com	39
Hacker Halted www.hackerhalted.com	77	Trusted Computer Solutions www.trustedcs.com/SecurityBlanket	13, 79
High Performance Computing on Wall Street www.flaggmgmt.com/hpc	49	USENIX Association www.usenix.org/osdi10/lja	35
Intel www.intel.com	3	Utilikilts www.utilikilts.com	79
iXsystems, Inc. www.ixsystems.com	C3		

Table 1. Comparing Lines of Code

	NOTEPAD	WEBNOTES
Java	672	
XML	50	
HTML		62
JavaScript		121
Source Lines	722	183

return to "home". We then return "false" to the browser, as we don't need it to continue.

We now define the errorHandler() function, which we reuse for all database transactions in the script. It displays an alert box with the error message and returns.

Next is the refreshNotes() function. We use jQuery to find all the <li> elements on the "home" screen and remove them. Then, we execute a database transaction to find all the note records, and use the "home" screen template to create a list item for each note and insert it into the "home" screen. We add a .click function to each list item that will take users to the "editNote" screen when they click on a note title.

All you need to develop Web applications is a text editor to write JavaScript, CSS and HTML, and a browser to test the results.

The replaceNoteById(), editNoteById() and deleteNoteById() functions are all very similar to addNote(), with the SQL template changed appropriately.

Running the App

The application can be run from the browser on an iPhone or Android device. As expected from looking at HTML5 features on different phones, Webnotes works fine on the iPhone, iTouch, iPad and Droid. It does not work on the G1, because that phone doesn't support client-side database transactions.

Packaging the App for Distribution

If you want to package a Web application for distribution on iTunes or Android Market, one way is to use the appropriate SDK and write a small wrapper application. The application creates a browser Intent (Android) or a UIWebView (iPhone) and gives it the location of index.html. We don't have room to go into the SDKs here, but the applications are literally a few lines of code.

Or, for iTunes, you can let a package like PhoneGap do the work for you. You still need the iPhone SDK, so you have to create the package on a Mac, but PhoneGap makes the process simpler. Once it's created, you can upload it to iTunes like any other iPhone app.

If you don't care about iTunes or Android Market, there's another way—package your application as an HTML5 Offline Application. Listing 3 is a manifest file, webnotes.manifest, that you put in the home directory of your application. You also need to add an attribute to the <html> element in index.html:

```
manifest="webnotes.manifest"
```

One more thing—if you're serving the files from an Apache Web server, the .htaccess file in your Web directory needs a line like:

```
AddType text/cache-manifest .manifest
```

This tells Apache to serve .manifest files with the correct MIME type. When a user first goes to the Web site, the server will download the files listed in the manifest and keep them on the device. On subsequent visits, the file will be reloaded if the manifest changes—even if the change is in a comment field.

On Apple devices, when users go to your Web site, they can touch + at the bottom of the browser to put an icon for that URL on their homescreen (remember that .icon attribute when we initialized jQTouch?). If you've created an off-line application, it loads and executes from local storage, much like a native application.

Summary

We defined Webnotes to be similar to an example application that comes with the Android SDK called NotePad. See Table 1 for a comparison of lines of code.

If the effort to write a line of code is about the same in any language, it would take about a fourth of the time to write the application as a Web application—and it runs on most mobile WebKit-based browsers. That's worth considering as you plan your next mobile application development.■

Rick Rogers has been a professional embedded software developer for more than 30 years (FORTRAN to JavaScript). He's currently a Mobile Solutions Architect at Wind River Systems. He welcomes feedback on this article at portmobileapps@gmail.com.

Resources

Canalys 2009 Smartphone Market Analysis: **www.canalys.com/pr/2010/r2010021.html**

AdMob Mobile Metrics for January 2010: **metrics.admob.com/wp-content/uploads/2010/02/AdMob-Mobile-Metrics-Jan-10.pdf**

Comparison of Layout Engines (HTML5): **en.wikipedia.org/wiki/Comparison_of_layout_engines_(HTML5)#cite_note-114**

Dive into HTML5's Site to Check for HTML5 Features in Browsers: **diveintohtml5.org/past.html**

What's My User Agent?: **whatsmyuseragent.com**

jQTouch: **www.jqtouch.com**

PhoneGap: **phonegap.com**

iui: **code.google.com/p/iui**

iWebkit: **iwebkit.net**

Jon Stark's excellent book on Building iPhone Apps with HTML, CSS and JavaScript: **building-iphone-apps.labs.oreilly.com**

SEPTEMBER 11-13 2010
COLUMBUS CONVENTION CENTER
OHIOLINUX.ORG

OHIO LINUX FEST

HOW
WILL
FREE
CHANGE
THE
WORLD?

WEB APPLICATIONS WITH Java/JSP

LINUX'S HISTORY AS AN ENTHUSIAST'S PLAYGROUND HAS ALWAYS MADE IT A FUN PLACE TO WORK FOR PROGRAMMERS. COMBINE THE FUN OF LINUX WITH THE POWER OF JAVA AND JSP, AND QUICKLY BUILD SECURE MULTI-TIER WEB APPLICATIONS USING THE LATEST TECHNOLOGIES.

All the cool new programming languages, like Ruby, always have compilers/interpreters and tools for Linux, and the old UNIX standbys like Tcl/Tk are still around when you need them. Why, then, is Java not a ubiquitous player in the Linux arena?

Linux and Java really do have a lot to offer each other. Both are rock-solid and scalable server-class software systems, and most college and university graduates with software-related degrees are familiar with them, making for a powerful combination. In this article, I introduce you to Java Web applications through the Java Servlet Specification, the Java programming language itself and Java Server Pages. These three tools can help you get a Web application running in a lot less time than you think.

CHRISTOPHER SCHULTZ

The Java Servlet Specification

The Java Servlet Specification defines a Servlet Container, a Web application and the Servlet API, which is the glue that holds these pieces together.

A Servlet Container is analogous to a Web server, but it also knows how to deploy and manage Web applications, and so it often is known as an Application Server. The Servlet Container provides services that support the Servlet API, which is used by the Web application to interact with HTTP requests and responses.

Java Web Applications

A Java Web application is a self-contained collection of configuration files, static and dynamic resources, compiled classes and support libraries that are all treated as a cohesive unit by the Servlet Container. They are somewhat different from standard LAMP-style Web applications, which are more like collections of associated programs or scripts than formally defined, self-contained units. To demonstrate a Java Web application, I have developed a simple "timesheet" featuring some of the standard Java libraries that helped me write it.

Typically, a Web application is packaged in a WAR (Web ARchive) file, which is just a ZIP file with a special directory structure and configuration file layout. The directory structure of the Web application logically and physically separates these types of files. The WEB-INF directory contains all the configuration files, a lib directory contains all libraries (packaged in JAR, or Java ARchive files), and a classes directory contains the application's compiled code. Listing 1 shows the file layout of the Web application for reference.

Listing 1. Contents of timesheet.war

```
index.jsp
tasks.jsp
WEB-INF/web.xml
WEB-INF/lib/jstl-impl-1.2.jar
WEB-INF/lib/jstl-api-1.2.jar
WEB-INF/classes/lj/timesheet/Task.class
WEB-INF/classes/lj/timesheet/GetTasksServlet.class
WEB-INF/classes/lj/timesheet/BaseServlet.class
WEB-INF/classes/lj/timesheet/Client.class
WEB-INF/classes/lj/timesheet/SaveTaskServlet.class
WEB-INF/classes/ApplicationResources_en.properties
WEB-INF/classes/ApplicationResources_de.properties
WEB-INF/classes/ApplicationResources.properties
WEB-INF/classes/ApplicationResources_es.properties
WEB-INF/classes/ApplicationResources_fr.properties
META-INF/context.xml
META-INF/MANIFEST.MF
```

The WEB-INF directory also contains a special file, web.xml, which is known as the Web application's deployment descriptor. It defines all the behaviors of the Web application, including URI mappings, authentication and authorization. Let's look at the deployment descriptor for this Web application.

You can see that each servlet is defined in a <servlet> element that defines the Java class that contains the code, as well as a name for the servlet (to be used later). After the servlets have been defined, they are then mapped (by name) to incoming URIs using <servlet-mapping> elements. This servlet mapping may seem tedious and verbose, but it can be very powerful for several reasons:

Listing 2. web.xml

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

<web-app xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
        http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
    version="2.5">

  <servlet>
    <servlet-name>get-tasks</servlet-name>
    <servlet-class>lj.timesheet.GetTasksServlet</servlet-class>
  </servlet>

  <servlet>
    <servlet-name>save-task</servlet-name>
    <servlet-class>lj.timesheet.SaveTaskServlet</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>get-tasks</servlet-name>
    <url-pattern>/tasks</url-pattern>
  </servlet-mapping>

  <servlet-mapping>
    <servlet-name>save-task</servlet-name>
    <url-pattern>/save-task</url-pattern>
  </servlet-mapping>

  <security-constraint>
    <web-resource-collection>
      <web-resource-name>Protected Pages</web-resource-name>
      <url-pattern>/tasks</url-pattern>
      <url-pattern>/save-task</url-pattern>
    </web-resource-collection>
    <auth-constraint>
      <role-name>*</role-name>
    </auth-constraint>
  </security-constraint>

  <login-config>
    <auth-method>BASIC</auth-method>
    <realm-name>Timesheets</realm-name>
  </login-config>

  <security-role>
    <description>Users of the timesheet application</description>
    <role-name>user</role-name>
  </security-role>
</web-app>
```

1. You can map one servlet to multiple URIs.

Container Configuration

You may be wondering how the Servlet Container knows anything about the database. The answer is found in another configuration file, specific to each Servlet Container, that includes this information. You can look at the conf/context.xml file that comes with the sample Web application files for this article (see Resources), but you'll have to refer to the Apache Tomcat Web site for details on this Tomcat-specific configuration file format. If you want to deploy the sample application on a different application server, you need to write your own container-specific configuration file, which includes your database configuration.

2. You can use wild-card mappings (/foo/bar/*).

3. You may not want to reveal any of the code's structure to remote visitors.

4. You may have servlets you don't want to map at all.

After the servlet mappings come container-managed authentication and authorization. The Servlet Specification requires that Servlet Containers provide mechanisms for authentication and authorization, and the configuration in the Web application is declarative: web.xml simply specifies what resources are protected and who is allowed to access them, using role-based authorization constraints. The setup is quite straightforward, and the Web application becomes simpler by not having to implement that capability inside the application. In this application, I've chosen to use HTTP BASIC authentication to simplify the application. DIGEST, FORM and (SSL) CLIENT-CERT are other options allowed by the Servlet Specification.

Java Servlets

Now that you have a sense of how the Web application is packaged and deployed, let's turn our attention to the real action in the Web application: the code.

Java is both a programming language and a runtime environment, much like Perl and PHP. In those cases, the compiler generally is invoked when the script is executed, while Java is always compiled beforehand. The Java programming language itself is object-oriented, procedural, block-structured and entirely familiar to anyone who has written in a C-like language. It has a number of explicitly defined primitive data types as well as reference types. All the Java code you write lives within the definition of a class, including servlet code.

Handling a Request

Let's take a look at the source code for the GetTasksServlet (Listing 3), which implements the "get-tasks" servlet, which is mapped to the URL /tasks.

Listing 3. GetTasksServlet.java

```
package lj.timesheet;

import java.io.IOException;

import java.util.ArrayList;
import java.util.Date;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class GetTasksServlet
    extends BaseServlet
{
    public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
        throws ServletException, IOException
    {
        String username = request.getUserPrincipal().getName();

        try
        {
            List<Client> clients = getClients();

            // Convert client list to lookup table
            Map<Integer,Client> clientMap
                = new HashMap<Integer,Client>(clients.size());

            for(Client client : clients)
                clientMap.put(client.getId(), client);

            request.setAttribute("clients", clients);
            request.setAttribute("clientMap", clientMap);
            request.setAttribute("tasks", getTasks(username));

            getServletContext().getRequestDispatcher("/tasks.jsp")
                .forward(request, response);
        }
        catch (SQLException sqle)
        {
            throw new ServletException("Database error", sqle);
        }
    }

    ...
}
```

The first line of the file declares the "package" in which the

Listing 4. tasks.jsp

```
<%@ page
  pageEncoding="UTF-8"
%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
               "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<fmt:setBundle basename="ApplicationResources" />
<html>
  <head>
    <title><fmt:message key="tasks.title" /></title>
  </head>

  <body>
    <h1><fmt:message key="tasks.title" /></h1>

    <table>
      <tr>
        <th><fmt:message key="tasks.header.date" /></th>
        <th><fmt:message key="tasks.header.client" /></th>
        <th><fmt:message key="tasks.header.description" /></th>
        <th><fmt:message key="tasks.header.duration" /></th>
      </tr>

    <c:forEach var="item" items="${tasks}">
      <tr>
        <td>
          <fmt:formatDate dateStyle="short" value="${item.date}" />
        </td>
        <td><c:out value="${clientMap[item.clientId].name}" /></td>
        <td><c:out value="${item.description}" /></td>
        <td><c:out value="${item.duration}" /></td>
      </tr>
    </c:forEach>
    </table>

    <form method="POST" action="<c:url value="/save-task" />">
      <fieldset>
        <legend>Add New Task</legend>
        <div class="form-field">
        <label for="clientId">
          <fmt:message key="tasks.header.client" />
        </label>
        <select name="clientId" id="clientId">
          <option value="">Please choose&hellip;</option>
          <c:forEach var="client" items="${clients}">
            <option value="${client.id}">
              <c:out value="${client.name}" />
            </option>
          </c:forEach>
        </select>
        </div>

        <div class="form-field">
        <label for="description">
          <fmt:message key="tasks.header.description" />
        </label>
        <input type="text" name="description"
              id="description" size="50" />
        </div>

        <div class="form-field">
        <label for="duration">
          <fmt:message key="tasks.header.duration" />
        </label>
        <input type="text" name="duration" id="duration" size="4" />
        </div>

        <div class="buttons">
          <input type="submit"
                value="<fmt:message key="task.save" />" />
        </div>
      </fieldset>
    </form>
  </body>
</html>
```

class is defined. Packages help keep code organized and have implications on variable, method, and class scope and visibility. The next set of lines are "imports" that indicate to the compiler which classes will be referenced by this class. Those classes beginning with `java.` are standard Java classes, while those beginning with `javax.servlet` are those provided by the Java Servlet Specification. Then, we define a class called GetTasksServlet that extends an existing class called HttpServlet, the basis for all HTTP-oriented servlets. The HttpServlet class defines a number of doXXX methods, where XXX is one of the HTTP methods, such as GET (doGet), POST (doPost), PUT (doPut) and so on. I have overridden the doGet method in order to respond to HTTP GET requests from clients.

The doGet method accepts two parameters: the request and the response, which provide hooks into the resources provided by the Servlet Container and to the information provided by the client for a particular HTTP request. I use two utility methods (defined later in the class) to obtain a list of clients and a list of tasks, and store them in the request object's "attributes", a location where data can be placed in order to pass them between stages of request processing. You'll see how to access this information next when I cover JSP files for generating content. Finally, I invoke the "request dispatcher's" forward method, which tells the container to forward the request to another resource: tasks.jsp.

Java Server Pages

Java Server Pages (JSPs) is a technology for dynamic content generation for things like Web pages. JSPs are analogous to PHP pages, where static text can be mixed with Java code, and the result is sent to the client. Technically speaking, JSPs are translated on the fly by a special servlet (provided by the Servlet Container) into their own servlets and compiled into bytecode, and then run

A Note about Scoping

The Java Servlet Specification defines three data scopes: application, session and request. The JSP Specification adds a fourth one: page. Each of these scopes is a place where data can be stored by the Web application for use at any time. When object identifiers are used in expressions, they are looked up in each scoping level until an object is found. First, the page scope is searched, then the request, then the session and then the application. This is how tags like <c:forEach> can define new object names, and the tags within the body can access them.

just like "normal" servlets. Listing 4 shows the code for tasks.jsp—the page referenced in GetTasksServlet's doGet() method above.

The page begins with a page declaration that includes some metadata about the page, including the output character encoding, and then some "taglib" tags that tell the JSP compiler I want to use some "tag libraries". Tag libraries are helper libraries that allow JSP scripts to wield powerful tools using very simple syntax. After the DOCTYPE, there is a <fmt:setBundle> element, and in the <title> of the page, there is a <fmt:message> element. These two tags, defined by the "fmt" tag library, work together to provide internationalization capabilities to this page. The <fmt:setBundle> tag defines the string resource bundle to be used by the page, and the <fmt:message> tag uses that bundle to pull localized text from the appropriate file to display in the page. The result is, when I visit this page with my Web browser set to the en_US locale, I get text in English, but if I switch the locale to fr_BE and reload the page, the page will switch into French without any further programming.

ON THE OTHER HAND, THERE ARE SOME PHILOSOPHICAL AND PRACTICAL REASONS NOT TO STUFF EVERYTHING INTO A SINGLE JSP.

The standard Java API actually provides all this capability out of the box, but the JSTL (Java Standard Template Library) "fmt" tag library gives us access to Java's internationalization APIs without having to write any Java code. By providing a Java property file (a text file with simple key=value syntax) for each locale I want to support, I get text localization practically for free. Further down in the JSP file, you can see the use of another "fmt" tag, <fmt:formatDate>. This tag formats a date object using the user's locale and a simple name for the format ("simple" in this case). This results in MM/dd/yy in the US and dd/MM/yy in Belgium.

The next JSTL tag is <c:forEach>. This tag actually encloses a body, which is evaluated multiple times: once for each item it finds in the "items" attribute. The value of `${items}` means that the value is not just a simple literal value, but an expression that should be evaluated. The object "items" is found in the request object's "attributes"—remember I put it there in the servlet code—and used here as the data for the loop. Within the body of <c:forEach>, the "item" object is defined and can be used by any JSTL tags.

The next tag, <c:out>, outputs a value in a Web-safe manner. If the value contains any < characters, they will be escaped to avoid nasty XSS attacks. The value of `${clientMap[item.clientId].name}` is again an expression that tells <c:out> to take the client ID from the item object, use that to look up a value in the "clientMap", and then get its name. The objects "item" and "clientMap" are both retrieved from the request attributes, and the <c:out> tag handles the expression evaluation and output escaping for us.

This page includes a form that allows us to enter new tasks. One of the most important attributes of the <form> is the "action", which, of course, tells the form where the data should be sent. I use the <c:url> tag here to generate a URL for us. It may seem silly to use a tag when I simply could have used `/timesheet/save-task` as the value of the action attribute, but there are some subtle issues in play here, which must be taken into account. First, a Web application can be deployed into any "context path", which means that the path to the servlet might actually be `/my-timesheet/save-task`. The <c:url> tag knows where the Web application has been deployed (courtesy of the request object, defined by the Servlet API) and can provide the appropriate path prefix to the URL. Second, <c:url> can encode the URL with a session identifier, which is essential to providing a good user experience for many Web applications. The <c:url> tag is smart enough to omit the session identifier from the URL if the client is using cookies to communicate the session identity to the server, but to include it in the URL as a fallback when cookies are unavailable. Sessions are another handy feature defined by the Servlet Specification, provided by the Servlet Container and accessible via the Servlet API.

Accepting Form Submissions

Now that I've covered the display of the timesheet and the form that can be used to submit a new task, let's take a look at the code that accepts this form submission: SaveTaskServlet.java (Listing 5), which implements the "save-task" servlet, which is mapped to the URL /save-task.

The SaveTaskServlet overrides the HttpServlet's doPost method so we can handle HTTP POST messages. It gathers the data from the request, made available through the request object's getParameter method, then creates a Task object and calls a helper method (defined later in the class) called "save". After saving the new task, the user is redirected to the "tasks" servlet to view the updated list of tasks. Did you notice that the line of code performing the redirect calls response.encodeRedirectURL and prepends the context path to the target URI? This is precisely the tedium that is avoided in JSP files by using the <c:url> tag.

Listing 5. SaveTaskServlet.java

```
package lj.timesheet;

import java.io.IOException;

import java.util.Date;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Timestamp;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class SaveTaskServlet
    extends BaseServlet
{
    public void doPost(HttpServletRequest request,
                       HttpServletResponse response)
        throws ServletException, IOException
    {
        Integer taskId;

        if(null == request.getParameter("id")
           || "".equals(request.getParameter("id").trim()))
            taskId = null;
        else
            taskId = new Integer(Integer.parseInt(
                                request.getParameter("id")));

        int clientId = Integer.parseInt(
                              request.getParameter("clientId"));
        Date date = new Date();
        String description = request.getParameter("description");
        int duration = Integer.parseInt(
                              request.getParameter("duration"));

        String username = request.getUserPrincipal().getName();

        Task task = new Task(taskId, username, date,
                              clientId, description, duration);

        try
        {
            save(task);

            response.sendRedirect(response.encodeRedirectURL(
                request.getContextPath() + "/tasks"));
        }
        catch (SQLException sqle)
        {
            throw new ServletException("Database error", sqle);
        }
    }

    // see below
}
```

SaveTaskServlet also defines a "save" method that interacts with the database. While none of this code is servlet-oriented, it's instructive to see the power of some of Java's standard APIs. In this case, it's the JDBC API that gives us access to relational databases (Listing 6).

First, this method obtains a connection from a database connection pool and then determines if the Task is being created from scratch or updated (although our UI doesn't offer an "update" method yet, this class has been designed to allow updates). In each case, a parameterized SQL statement is prepared and then filled with data passed in from the calling code. Then, the statement is executed to write to the database, and a new object is passed back to the caller. In the case of a new task, the database-generated primary key is fetched from the statement after execution in order to pass it back to the caller.

Under normal circumstances, methods such as "save" would be split out into a separate class for easier organization, testing and architectural separation, but I've left them in the servlet classes for simplicity.

The example's full source code and prebuilt WAR file are available from the *Linux Journal* FTP server (see Resources), and I encourage you to download it and play around with it. I've also included quick installation instructions for Java and the Apache Tomcat servlet container, which will be required to run the example application.

Java and Model-View-Controller Architecture

Often, Perl and PHP-based Web applications are composed of self-contained scripts that perform one task: loading and displaying tasks, for instance. This kind of thing is entirely possible using nothing but JSPs. There are tag libraries that perform SQL queries, and you even can write Java code directly into a JSP, although I haven't covered it here because it's not necessary with the rich tools provided by the JSTL. On the other hand, there are some philosophical and practical reasons not to stuff everything into a single JSP. Most (Java) programmers subscribe to the "model-view-controller" architecture, where code is separated into logical units that model your problem domain (that would be the Task and Client objects in our example), provide views of your data (that's our JSPs) and control program flow (the servlets). This architectural separation actually leads to quite a few practical benefits, including:

1. Easier code maintenance: separation promotes code re-use and simplifies automated testing.

2. Error handling: if the controller is the only likely component to fail (due to bad input, db connection failure and so on), you don't have to worry about the view component failing during rendering, ruining your output.

Listing 6. SaveTaskServlet.java

```
public Task save(Task task)
    throws SQLException
{
    Connection conn = null;
    PreparedStatement ps = null;
    ResultSet rs = null;

    try
    {
        conn = getConnection();

        if(null == task.getId())
        {
            // A new task
            ps = conn.prepareStatement(
                     "INSERT INTO
                          task (owner, date, client_id,
                               description, duration)
                          VALUES (?,?,?,?,?)",
                     PreparedStatement.RETURN_GENERATED_KEYS);

            ps.setString(1,  task.getOwner());
            ps.setTimestamp(2, new Timestamp(
                                       task.getDate().getTime()));
            ps.setInt(3, task.getClientId());
            ps.setString(4, task.getDescription());
            ps.setInt(5, task.getDuration());

            ps.executeUpdate();

            rs = ps.getGeneratedKeys();

            if(!rs.next())
                throw new SQLException(
                    "Expected auto-generated key, got none");

            int taskId = rs.getInt(1);
            if(rs.wasNull())
                throw new SQLException(
                    "Got bogus auto-generated key");

            task = new Task(taskId,
                            task.getOwner(),
                            task.getDate(),
                            task.getClientId(),
                            task.getDescription(),
                            task.getDuration());
        }
        else
        {
            ps = conn.prepareStatement(
                     "UPDATE task SET date=?, client_id=?,
                                    description=?, duration=?
                         WHERE id=? AND owner=?");

            ps.setTimestamp(1, new Timestamp(
                                       task.getDate().getTime()));
            ps.setInt(2, task.getClientId());
            ps.setString(3, task.getDescription());
            ps.setInt(4, task.getDuration());
            ps.setInt(5, task.getId());
            ps.setString(6, task.getOwner());

            ps.executeUpdate();
        }

        return task;
    }
    finally
    {
        close(conn, ps, rs);
    }
}
```

Most Java projects are going to be split up in this way, so I wrote my example to illustrate this architecture, and I hope you consider using this architecture in your Java projects too.

Conclusion

Adding Java to your repertoire for building Web applications gives you access to the built-in services guaranteed by the Servlet Specification as well as a plethora of high-quality third-party libraries. Servlet containers provide many services useful to your Web applications through simple configuration and/or APIs. Java Server Pages can be used to build complex Web pages quickly while avoiding business logic. The Servlets you write to implement your business logic have full access to many APIs for just about anything you can think of. The power of Java Web applications and the stability and scalability of Linux can be combined into a platform on which many high-quality on-line services are built, including mine. I hope I've given you a taste of how easy it is to create a robust and useful Java Web application using the tools provided by the Java Servlet Specification, and that you consider using Java for your next Web application.■

Christopher Schultz is the CTO of Total Child Health, Inc., a healthcare software company based in Baltimore, Maryland. He has been developing Web applications in Java since those words could reasonably be placed in the same sentence. He is an active member of the Apache Tomcat users' mailing list, and he is a committer on the Apache Velocity Project. He lives in Arlington, Virginia, with his wife Katrina, son Maxwell and dog Paddy.

Resources

Example Web Application for This Article: **ftp.linuxjournal.com/pub/lj/listings/issue197/10810.tgz**

Java Servlet Specification (version 2.5): **jcp.org/aboutJava/communityprocess/mrel/jsr154/index2.html**

JavaServer Pages Standard Tag Library: **https://jstl.dev.java.net**

Apache Tomcat Web Site: **tomcat.apache.org**

LINUXCON

NORTH AMERICA 2010

August 10th - 12th, 2010

Renaissance Boston Waterfront Hotel, Boston, MA

Keynote Speakers Include:

Ravi Simhambhatla

Vice President and Chief Information Officer at Virgin Atlantic

Wim Coekaerts

Senior Vice President, Linux and Virtualization Engineering at Oracle

Eben Moglen

Chairman at the Software Freedom Law Center (SFLC)

Rob Chandhok

President of Qualcomm Innovation Center, Inc. (QuIC)

Linux Kernel Roundtable Panelists:

James Bottomley

SCSI Subsystem Maintainer

Dave Jones

Fedora Kernel Maintainer

Jon Corbet

Editor at Linux Weekly News

Chris Mason

btrfs File System Creator

Ted Ts'o

Core File System Maintainer

THE LINUX FOUNDATION

REGISTER NOW FOR ONLY $400

http://events.linuxfoundation.org/

▶Create a Debian Repository for Your System

If you have a Debian-based system, once you've got everything installed, you can create a Debian repository from it and use that repository for installing additional similarly configured systems, or you can use it as a source for a re-install in the event that your system somehow becomes corrupted.

To do this, install the package dpkg-dev. You can install it with apt-get from the command line, or you can install it using a GUI package manager, such as Synaptic.

Now, create a directory—for example, my_repo. This will be the root of your repository. Under this, create a directory named binary. Next, copy all the .deb files from /var/cache/apt/archives/ into the binary directory. Then, go to the my_repo directory, and run the command:

```
$ dpkg-scanpackages binary /dev/null | gzip -9c > binary/Packages.gz
```

This creates your packages list. After that, save the whole my_repo directory onto another system. Then, change the /etc/apt/sources.list file, and add the path of the my_repo:

```
deb file:///home/boss/my_repo binary/
```

Now, reload the repository list and check your new repository.

—**KOUSIK MAITI**

▶Auto-Typing and Mouse Movements

Sometimes you may need to type the same thing repeatedly, whether it's filling out a form or typing a common word or phrase over and over again. There is a simple program for Linux called xte that allows you to control virtual key presses and mouse gestures that are sent to a program. xte is part of the xautomation package. It should be available through your package manager. For Debian-based systems, you can run:

```
$ sudo aptitude install xautomation
```

Once the package is downloaded and installed, you can use xte from the command line, like so:

```
$ xte 'sleep 5' 'str hello world'
```

This command waits five seconds and then types the string "hello world" into whatever application has focus. You not only can send strings, but you also can send key presses. So, let's say you want to send the key press for Enter, after you send the string "hello world". Simply do the following:

```
$ xte 'sleep 5' 'str hello world' 'key Return'
```

There are a number of keys that can be sent using xte. Some modifier keys include Shift_L, Shift_R, Ctrl_L and Ctrl_R. As you can see, xte not only can send a Ctrl key press, but it also can distinguish between left and right Ctrl key presses. This is important, because some programs have different functions for the left and right Ctrl keys.

When typing the command for these key presses, keep in mind that the commands are case-sensitive. For instance, `key Return` will work, but `key return` will not. Use the `xte --help` command to get a full list of useful keys that you can send.

You can use xte for many useful things. Let's say you type your name, or maybe the name of your company, a lot throughout the day. You easily can create a script with xte that will send the string of information and then link that script to a set of shortcut keys for your desktop environment. So, instead of typing out "Johnson, Joseph and Jack's Law Office", you simply can press Ctrl-Alt-N, and the script will type it for you.

I also use xte for was controlling Compiz on the touchscreen in my car. Without a mouse or keyboard, I was unable to use some of Compiz's useful features, such as scaling. So, after setting scaling to be controlled by moving the cursor to the top-right corner of the screen, I added an icon to the GNOME toolbar that ran a script that did the following:

```
$ xte 'sleep 1' 'mousemove 9999 0'
```

The first number (9999) is the X-axis value, and the second (0) is the Y-axis value. This command waits one second, which allows me to lift my finger from the touchscreen before the cursor moves, and then relocates the mouse cursor to the far right of the screen and up to the very top. Now, in combination with my Compiz settings, I can press the icon on my toolbar and get a nice view of all my open windows. I click the one I want, and I'm off and running. This makes touchscreen usage much more convenient and raises the cool factor a bit.

xte has many options I haven't touched on here (such as mouse clicks and holding a key or a mouse press for a given amount of time). I hope it has sparked an interest in you to give it a try and play with some. It just may be the tool you need to get a job done.

—**KRISTOFER OCCHIPINTI**

▶Three Steps to Find Your Total Download Bandwidth Usage

I have been working on bandwidth-monitoring of late, and I find the following three steps handy to find my download byte count. These steps use iptables, which is available with almost all distributions. It most likely already will be installed on your system (it is the basic firewall in Linux).

Steps one and two set up the monitoring, and step three allows you to view your download byte count. The first two steps need to be done only once (at boot time, if you want this available all the time). You need to run all the steps as root.

Step 1: create a chain:

```
$ iptables -N input_accounting
```

This creates an iptables chain named input_accounting.

Step 2: add a rule:

```
$ iptables -I INPUT -j input_accounting
```

This causes all incoming packets to "pass through" your newly created chain.

Step 3: start checking your bandwidth:

```
$ iptables -L -v | \
                grep input_accounting | \
                grep anywhere | \
        awk '{ printf("%s\n", $2) }'
```

This should output your download byte count—for example, "500K".

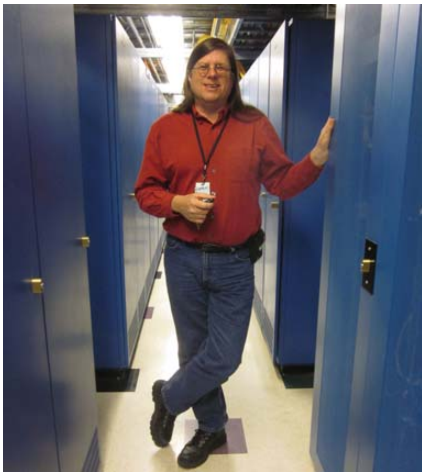
—**TANMAY MANDE**

Adventures in Re-provisioning

Hanging with Tim Pozar on the wireless frontier.

DOC SEARLS

You can model the San Francisco Bay Area with your right hand. Bring the tip of your thumb and index finger together, then part them slightly. The gap between them is the Golden Gate, with its famous bridge. Going clockwise from there, the end of your bent index finger is Marin County. Berkeley and Oakland are at the base of your index finger. San Jose is at the base of your thumb. The rest of your thumb is the Peninsula. San Francisco is the whole tip of your thumb. The knuckle below your thumb's nail is San Bruno Mountain. That's where our story starts.



Tim Pozar in one of his natural habitats.

In the 1940s, San Bruno Mountain's long ridge began to bristle with towers for TV and FM stations. Since then, most of those stations have moved to Sutro Tower in the midst of the City—a landmark locals call "the world's largest roach clip". One of the few remaining TV transmitters on San Bruno is KTSF, also known as Channel 26—that was its old analog channel. Now digital, it actually radiates on Channel 27, even if TV tuners still say "26". In fact, the displayed channels for most TV stations in the US are now other than those they transmit on. Being digital, they can transmit data that tells the receiver what channel to display, regardless of the actual channel used. But that stuff hardly matters, because the percentage of Americans still watching over-the-air TV is down to single digits. Instead, they watch cable or satellite or bypass TV altogether and watch on a computer or a handheld mobile device. Still, KTSF puts out a signal with a listed power of a half-million watts. Thus, TV's mainframe age persists.

Inside the base of KTSF's tower is a small yellow shack with a leaky roof. It was here that I stood with Tim Pozar last May as he showed me how gear in a rack inside the shack was relaying many megabits' worth of Internet bandwidth from one point to another, each transmitter emitting signals measured in thousandths of a single watt. He and a colleague were busy shaking down a link to the Maker Faire that would happen down on the Peninsula a few days later. (It worked fine.)

Tim's long résumé includes decades spent both as an Internet pioneer and a broadcast engineer. Of the two tracks' convergence, he explained, "I do work for Univision in San Francisco building out video servers for them. I was talking to Don Ready over there, who is the Assistant Chief Engineer. He told me with some regret that he is doing very little 'broadcast engineering' now. Most of the time he is doing IT. Fiber, twisted pair, Ethernet, IP, switches, routers and Linux servers are the new technology for distribution of broadcast."

But the shift isn't a matter just of swapping one tech for another. It involves clever and resourceful re-provisioning. In both principle and practice, re-provisioning is central to the means and missions of openness in general and Linux in particular. Old purpose-built structures host new stuff made for new purposes—or (as with the case of Linux) better ways of serving the old purposes. In many cases, the old and the new coexist and cooperate. KTSF and its high-wattage structures still operate within a broadcast regime that's leveraging its original infrastructure about as far as it will go, while new infrastructure gets built within the old. Credit goes to old systems welcoming the new and to resourceful pioneers, such as Tim and his colleagues with the Bay Area Wireless Users Group (and other groups with similar names).

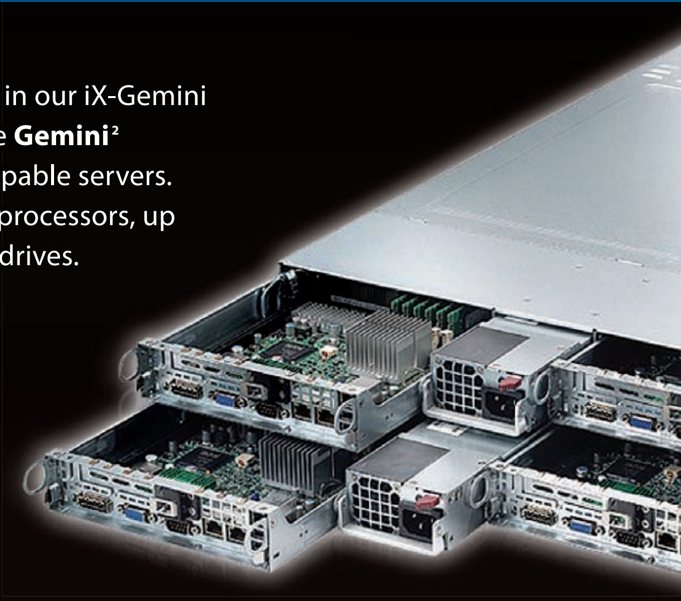
In that latter category is the work Tim and his buddy Matt Peterson are doing on the Farallons Broadband Project, which is organized by the California Academy of Sciences, the City of San Francisco, the Internet Archive and the US Fish & Wildlife Service. The Farallons, or Farallones, are a collection of small rocky islands 27 miles from San Francisco. There is no hard electrical infrastructure connecting the Farallons to land. Electricity is made there mostly by generators. The Internet is provided by a few milliwatts of wireless from San Bruno Mountain.

The wireless project is an exercise in minimized cost and complication. The "kit" includes Ubiquiti Bullet M2 (2.4GHz) and Rocket5 (5.2/5.5GHz) radios, a Pacific Wireless radome antenna, a Soekris net5501 comms computer and a Cisco WCX-C2950 switch. He hopes we'll forgive him using OpenBSD 4.5. (He's done plenty with Linux in other settings, but for this project he says, "pf is great. Easy syntax, handles NAT tricks well.") You can see one result through the Farallones Cam, which shows you plenty of the island's two main features: scary surf and zillions of birds.

I also followed Tim on a visit to the giant Digital Realty Trust data center, on the south edge of San Francisco. The center occupies a former retail chain furniture warehouse, which is another example of re-provisioning at work. The place is packed with servers for some of the most familiar domains on the Web, plus hundreds more of all sizes. What struck me standing in there, between racks and racks of humming equipment, is how much the whole place resembled broadcast transmission rooms. I could see there how the Net is almost done subsuming broadcast—and yet how giant data centers are hardly an end state for the Net itself. Much re-provisioning has yet to be done. In fact, it will never stop, as long as old structures and systems learn from new technologies and uses.

A photo set of my visit with Tim is at the *Linux Journal* Flickr site: **www.flickr.com/photos/linuxjournal**■

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.



Cool, Fast, Reliable

GPGPU computing for your office and data center

Designed from the ground up for ultimate customer satisfaction, Microway's WhisperStation integrates the latest CPUs with NVIDIA Tesla GPUs. Tesla's massively multi-threaded Fermi architecture, the CUDA™ C and FORTRAN language environments, and OpenCL™ provide the best performance for your application.

- Up to Four Tesla Fermi GPUs per WhisperStation, with 448 cores and 6 GB GDDR5, each delivering 1 TFLOP single and 515 GFLOP double precision performance
- Up to 24 cores with the newest Intel and AMD Processors, 128 GB memory, 80 PLUS® certified power supply, and eight hard drive
- Nvidia GeForce GTX 480 for state of the art graphics
- Ultra-quiet fans, strategically placed baffles, and internal sound-proofing

The Microway Advantage: Custom Integrations and HPC Expertise Since 1982

Put our years of expertise with Linux, Windows, CUDA and OpenCL to work for YOU!

Every Microway system is backed by pre and post sale techs who speak HPC. Whether it's graphics or GPGPU, FORTRAN or MPI, hardware problems or Linux kernel issues; you can talk to Microway's experts to design and support solutions for power hungry applications.

WhisperStation with 4 Tesla Fermi GPUs

1U Node with 2 Tesla Fermi GPUs

2U Twin² Node with 4 Hot-Swap Motherboards
Each with 2 CPUs and 256 GB

Microway's Latest Servers for Dense Clustering

- 1U nodes with 48 CPU cores, 512 GB and QDR InfiniBand
- 1U nodes with 24 CPU cores, 2 Tesla GPUs and QDR InfiniBand
- 2U Twin² with 4 Hot-Swap MBs, each with 2 Processors + 256 GB
- 1U S2070 servers with 4 Tesla Fermi GPUs

The Fastest CPUs and GPUs Ever

- 12 Core AMD® Opterons with quad channel DDR3 memory
- 8 Core Intel® Xeons with quad channel DDR3 memory
- 448 Core NVIDIA® Tesla™ Fermi GPUs with 6 GB GDDR5 memory

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

NVIDIA. TESLA CUDA

intel Channel Partner Premier 2010

AMD Fusion Partner Premier

GSA

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

Microway®
Technology you can count on℠