AJAX | POSTGRESQL | ZIMBRA | MOCHIKIT | FIREBUG | APTANA

LINUX JOURNAL™

Since 1994: The Original Magazine of the Linux Community

MAY 2007 | ISSUE 157

A close look at **PostgreSQL**

AJAX Ethernet Analyzer

mochikit
NOW with MochiKit Power!
Simple to Use!
Ajax
FOR THAT CLEAN, RICH CLIENT EXPERIENCE!
10 out of 10 Zimbra customers prefer Ajax!
Our extra ingredient Aptana cleans dirt fast!
NEW! Firebug removes bugs fast! Cleans with Perl and PostgreSQL too!
Combine with WordPress for brighter colors!

www.linuxjournal.com

USA $5.00 | CAN $6.50

05

0 71486 03102 4

PLUS:

- **Use Python to Manipulate ODF**
- **Single Packet Authorization in Practice**

APPRO
HPC Cluster Solutions

Enterprise and High-Performance Computing Under Your Control

Appro is leading 4P x86 computing by combining the latest technology that meets the demands of the enterprise HPC market.

4-Way *Xtreme*Workstation™

- AMD Opteron™ 8000 Series processors
- Up to 128GB of DDR2 533/667 memory
- Up to 6.0TB SATA or 2.4TB SAS
- Hot-swappable drives
- 2 PCI-E x16 slots for high-end graphics card
- Windows® or Linux OS

4-Way 3U *Xtreme*Server™

- AMD Opteron™ 8000 Series processors
- Up to 128GB of DDR2 533/667 memory
- Up to 4.5TB SATA or 1.8TB SAS
- 2 PCI-E x16 and 3 PCI-X slots
- Hot-swappable drives
- Redundant power supplies & fans
- ServerDome Management – IPMI 2.0
- Windows® or Linux OS

For more information, please visit www.appro.com
or call Appro Sales at 800-927-5464

AMD Opteron™ Processors:

- Quad-Core Ready - increase capacity without altering datacenter infrastructure
- Best performance per-watt with energy-efficient DDR2
- Optimized system performance with Direct Connect Architecture

AMD 64 Opteron™

CONTENTS MAY 2007 Issue 157

NOW with MochiKit Power!
mochikit
Simple to Use!
Ajax
FOR THAT CLEAN, RICH CLIENT EXPERIENCE!
10 out of 10 Zimbra customers prefer Ajax!
Our extra ingredient Aptana cleans dirt fast!
Combine with WordPress for brighter colors!
NEW! Firebug cleans out bugs fast!
Cleans with Perl and PostgreSQL too!

ILLUSTRATION BY GARRICK ANTIKAJIAN

FEATURES

ON THE COVER

- A close look at PostgreSQL, *p. 48*
- Ajax Ethernet Analyzer, *p. 74*
- Now with MochiKit Power, *p. 64*
- 10 out of 10 Zimbra customers prefer Ajax, *p. 42*
- Firebug removes bugs fast, *p. 22*
- Cleans with Perl and PostgreSQL too, *p. 58*
- Our extra ingredient Aptana cleans dirt fast, *p. 46*
- Combine with WordPress for brighter colors, *p. 70*
- Use Python to Manipulate ODF, *p. 86*
- Single Packet Authorization in Practice, *p. 82*

The competition doesn't stand a chance.

If you base deployment decisions on performance and price, Coyote Point's for you. We've cornered that market.

To prove it we asked The Tolly Group to evaluate our E350si application traffic manager against the competition. The results speak for themselves.

Throughput? Almost 40% more than others in our space. Cost of transactions per second? Up to four times less. Connection rate? In some cases, one-sixth the cost. One-sixth! And we're told Coyote Point is the #1 choice for today's open source networks.

But don't just take our word for it. Get the facts. Call 1.877.367.2696 or write info@coyotepoint.com for your free copy of the full Tolly Report.

Product Excellence Award 2006
Info Security Products Guide
CUSTOMER TRUST

2006 Network Computing
WELL-CONNECTED AWARDS
WINNER

CoyotePoint
Systems Inc

CONTENTS MAY 2007 Issue 157

COLUMNS

You entered room "foobar"

vader
I have to remember that tab key is no longer the auto complete key for the niks

vader
ok

wfti
Different chat client. That only works with IRC I'm afraid.

diskless
That's okay, I can handle typing out the nicks.

vader
oh well lol

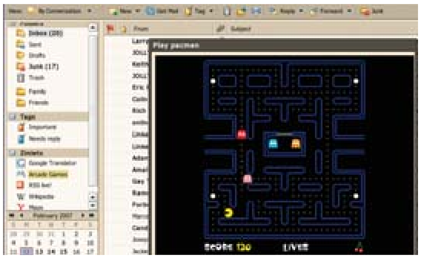
Cabernet
Can this thing do private chats?

vader
ummmmm I may be able to in time lol

wfti
Just click the person's nick on the right hand side, diskless.

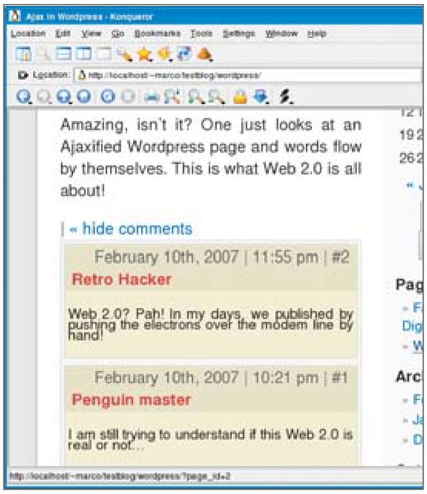
IN EVERY ISSUE

QUICK TAKES



INDEPTH

Turn all these "straight" quotes into "nice" quotes



Ajax in Wordpress - Konqueror
Location Edit View Go Bookmarks Tools Settings Window Help
Location: http://localhost/~marco/testblog/wordpress/

Amazing, isn't it? One just looks at an Ajaxified Wordpress page and words flow by themselves. This is what Web 2.0 is all about!

« hide comments

February 10th, 2007 | 11:55 pm | #2
Retro Hacker
Web 2.0? Pah! In my days, we published by pushing the electrons over the modem line by hand!

February 10th, 2007 | 10:21 pm | #1
Penguin master
I am still trying to understand if this Web 2.0 is real or not...

http://localhost/~marco/testblog/wordpress/?page_id=2

Next Month

LANGUAGES

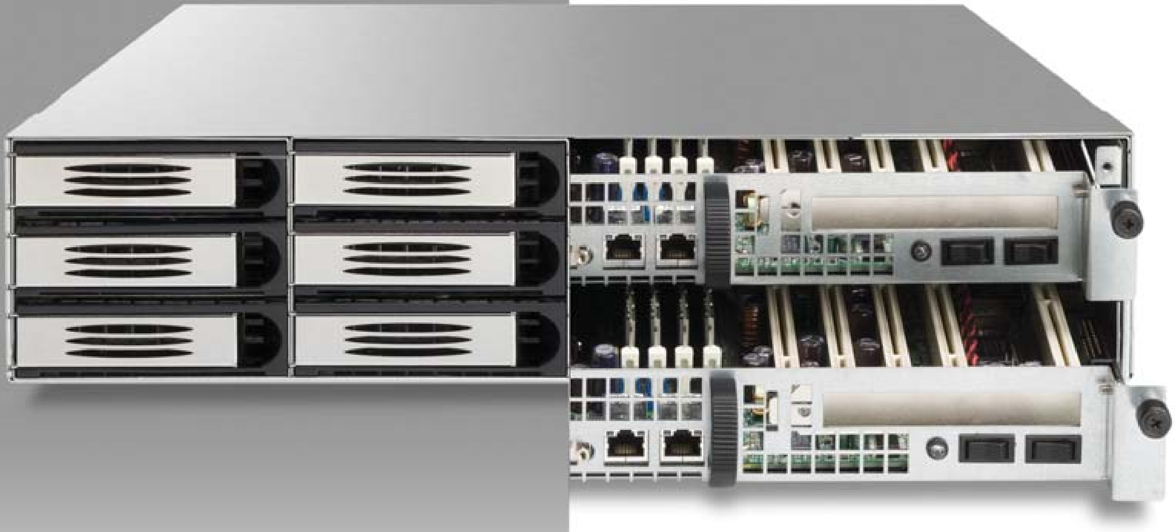
Next month's issue contains a smörgåsbord of articles on languages and programming tools. Frustrated by the fact that Qt 4.x uses synchronous access to databases by default? We tell you how to do asynchronous database access with Qt instead. Then, jump into the deep end of Python programming with the first of two Python tutorials. And, we talk with Sun's Simon Phipps about a variety of topics including the GPL-ization of Java.

There's more. Now that Reuven Lerner has given you information about MySQL and PostgreSQL, he helps you choose the database that's right for you. Plus, learn how to validate an e-mail address in PHP the right way.

USPS *LINUX JOURNAL* (ISSN 1075-3583) 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 US. Subscription rate is $25/year in the United States, $32 in Canada and Mexico, $62 elsewhere. POSTMASTER: Please send address changes to *Linux Journal*, PO Box 980985, Houston, TX 77098. Subscriptions start with the next issue.

Gemini 2U

– Ultra Dense Series



- Two fully independent systems in a 2U
- Ability to run two discrete operating systems in one box
- Up to 16 CPU cores
- Up to 12 hot-swappable SATA, SCSI, or SAS hard drives
- RAID 0, 1, 5, 6, 10, 50 available on both systems
- Opteron™ or Xeon™ multi-core processors
- Up to 64GB memory per motherboard
- One available PCI-X or PCI-E slot per motherboard
- High efficiency AC and DC power options

2 Systems in One

Built on open standards, the Gemini 2U elegantly accommodates two discrete motherboards in a 25" chassis uniquely designed for easy access from the rear.

Gemini 2U represents the realization of intoxicating power and superior environmental specifications, with considerably less power consumption, less heat and less noise. Remarkably, it all fits nicely into any standard rack. Front to back, Gemini 2U is both powerful and efficient.

At Open Source Systems we understand you need practical, customizable, and affordable solutions that are easy to manage and maintain.

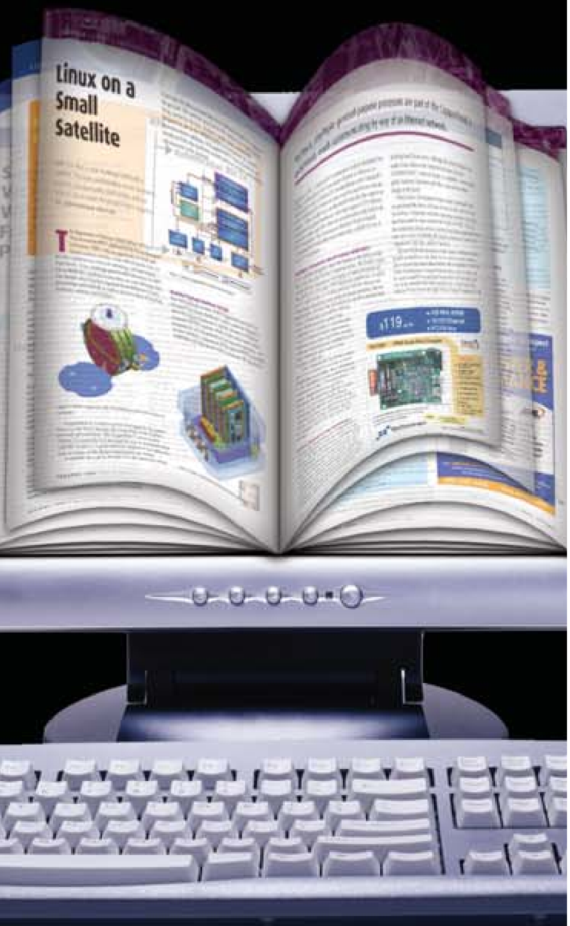
For more information and to request your evaluation unit today, visit us at **www.OpenSourceSystems.com**, or call direct at 866.664.7867.

Patent Pending

AMD Solution Provider PLATINUM

AMD 64 Opteron™

Open Source Systems™



LINUX JOURNAL

Editor in Chief
Nick Petreley, ljeditor@linuxjournal.com

Executive Editor	Jill Franklin jill@linuxjournal.com
Senior Editor	Doc Searls doc@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
Chef Français	Marcel Gagné mggagne@salmar.com
Security Editor	Mick Bauer mick@visi.com

Contributing Editors
David A. Bandel • Greg Kroah-Hartman • Ibrahim Haddad • Robert Love • Zack Brown • Dave Phillips • Marco Fioretti • Ludovic Marcotte • Paul Barry • Paul McKenney • Dave Taylor

Proofreader	Geri Gale

Publisher	Carlie Fairchild publisher@linuxjournal.com
General Manager	Rebecca Cassity rebecca@linuxjournal.com
Director of Sales	Laura Whiteman laura@linuxjournal.com
Regional Sales Manager	Joseph Krack joseph@linuxjournal.com
Regional Sales Manager	Kathleen Boyle kathleen@linuxjournal.com
Circulation Director	Mark Irgang mark@linuxjournal.com
Marketing Coordinator	Lana Newlander mktg@linuxjournal.com
System Administrator	Mitch Frazier sysadm@linuxjournal.com
Webmaster	Keith Daniels webmaster@linuxjournal.com
Accountant	Candy Beauchamp acct@linuxjournal.com

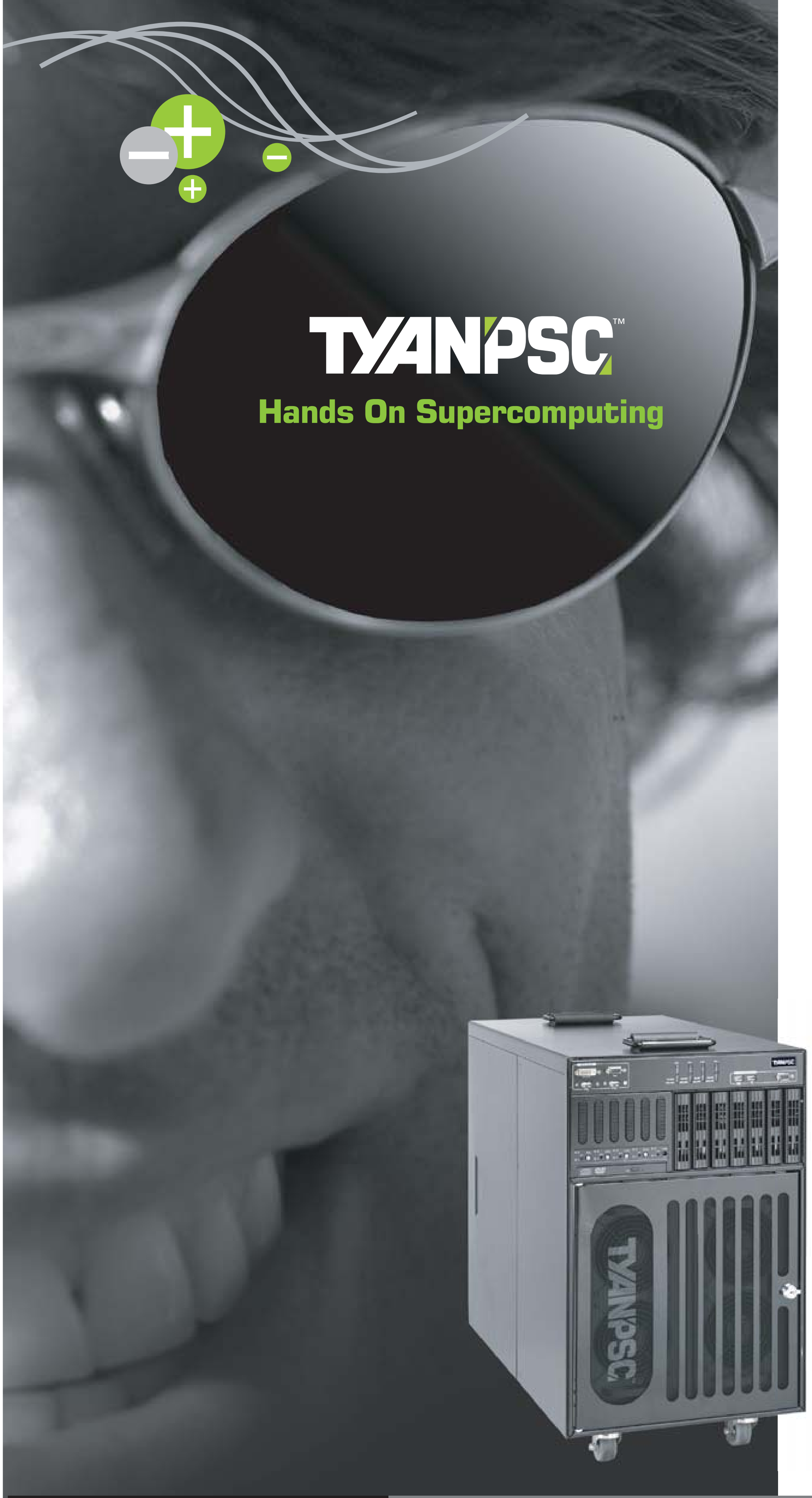
***Linux Journal* is published by, and is a registered trade name of, Belltown Media, Inc.**
PO Box 980985, Houston, TX 77098 USA

Editorial Advisory Board
Daniel Frye, Director, IBM Linux Technology Center
Jon "maddog" Hall, President, Linux International
Lawrence Lessig, Professor of Law, Stanford University
Ransom Love, Director of Strategic Relationships, Family and Church History Department,
Church of Jesus Christ of Latter-day Saints
Sam Ockman, CEO, Penguin Computing
Bruce Perens
Bdale Garbee, Linux CTO, HP
Danese Cooper, Open Source Diva, Intel Corporation

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: +1 713-589-3503
FAX: +1 713-589-2677
TOLL-FREE: 1-888-66-LINUX
MAIL: PO Box 980985, Houston, TX 77098 USA
Please allow 4–6 weeks for processing address changes and orders
PRINTED IN USA

LINUX is a registered trademark of Linus Torvalds.



Typhoon™ 600 Series Personal Supercomputer

TyanPSC's Typhoon™, the next generation turnkey Personal Supercomputer has the power to blow away all your computational needs! Purpose-built for office and laboratory environments, easy to deploy and use, the Typhoon™ provides intense computational power in remote or constrained places, works like a PC and is whisper quiet.

High Performance Computing Just Got Cooler

Clusters of Typhoons / Low Power
Small size Form Factor / Under Mobility

Typhoon

T-630 DX / T-650 QX Series

- Up to 186 / 256 GFlops at your desk!
- Turnkey, Easy-to-Deploy, and Easy-to-Use
- Integrated 5 node cluster - up to 20 / 40 processor cores in a box!
- Plugs into standard wall outlet - only uses 15 Amps
- Microsoft® Windows® Compute Cluster Server 2003 pre-installed
- High Performance in Constrained spaces: Office, Remote, Plane, Boat, etc
- RAID capable

intel Xeon inside™

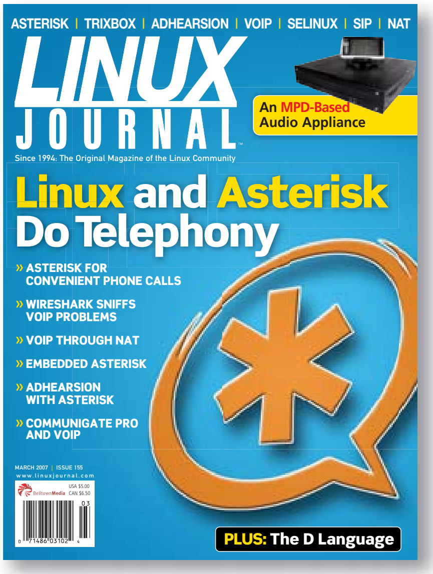
Microsoft® Windows® Windows Compute Cluster Server 2003

TYANPSC™
Personal Supercomputer

Tyan Computer USA
3288 Laurelview Court
Fremont, CA 94538 USA
Tel: +1-510-651-8868 Fax: +1-510-651-7688
Pre-Sales Tel: +1-510-651-8868 x5120
Email: marketing@tyan.com

For More Information, please visit
www.tyanpsc.com

letters


ASTERISK | TRIXBOX | ADHEARSION | VOIP | SELINUX | SIP | NAT
LINUX JOURNAL
Since 1994: The Original Magazine of the Linux Community
An MPD-Based Audio Appliance
Linux and Asterisk Do Telephony
» ASTERISK FOR CONVENIENT PHONE CALLS
» WIRESHARK SNIFFS VOIP PROBLEMS
» VOIP THROUGH NAT
» EMBEDDED ASTERISK
» ADHEARSION WITH ASTERISK
» COMMUNIGATE PRO AND VOIP
MARCH 2007 | ISSUE 155
www.linuxjournal.com
PLUS: The D Language

Feb was Fab

I think that your February 2007 issue is the BEST one ever. I'm not particularly excited about the state of "Linux Media"; many of the news sources are too geeky or too banal and boring. *LJ* is definitely one source that is not with the rest.

The February issue has something for everyone: geeky stuff, professional/sys admin stuff and general stuff. This is what I'm expecting from a magazine. Some of my favorite articles are the Perl and PHP integration one and the udev one. Both articles are completely unexpected and extremely useful; I have never seen anything covered close to it in any news source, and I thoroughly enjoyed reading them! Well done! And, please keep it up!

--
Dan

Document Rapid Change

I have a complaint about the way things get changed in Linux without giving ignorant people like me sufficient clues. I got a pre-installed Linux laptop and put my programs on it. Everything was fine until I tried to build drivers. Things had changed!

More recently, I downloaded Fedora Core 6 and installed it, and then installed svgalib. The demos failed to run, and a baffling message was issued: "cannot restore segment prot after reloc; permission denied". I asked the svgalib maintainer for help and sent a bug report to Red Hat. They pointed me to the source of the problem, which is SELinux. This is a new thing (for me). Why couldn't the error message have mentioned the name SELinux? And (a side issue) what kind of security policy lets me load a library from a removable drive but not from the hard disk?

As a programmer who tries to make things for other people to use, I know how easy it is to get lost in the details and fail to anticipate what is going to trip somebody up. We all just have to try harder.

--
William McConnaughey

Long Live Freedom

Nice mag, I read it over here in the UK as an import to my local Borders store.

I just read Nicholas Petreley's /var/opinion article in the February 2007 issue of *Linux Journal*, and I agree with his point of not using Novell tools, and hence aiding the MS strategy of infecting Linux with its intellectual property. I would add the following points for spreading Linux (or more generally, open-source software) freedoms:

1) Get Windows users started on OpenOffice.org and Firefox.

2) Public bodies need to use OOo and OSS where possible to ensure the lowest possible cost and maximum freedoms from forced file format changes.

3) Schools should be using OOo and OSS for the reasons above, but also to save the young from the indoctrination of the Windows/MS office toolset.

4) Do not promote the use of the Mono framework and C#, which has many MS patents that could be used in a future legal backlash against Linux. Better to use Java.

5) Support the concepts and goals of the Linux standard base (LSB) initiative, so that the support burden can be reduced and many more apps run without library searching and installation—not just Java.

--
Mark

Less Cooking More Eating

In the March 2007 Cooking with Linux, Marcel Gagné talks about Ekiga (GnomeMeeting), and as is often the case, manages to sell me on the idea of using one of his suggested software picks. I was really interested in trying out Ekiga, so I went to its Web site to download it. Unfortunately, the site was hard to use (download was broken in a few places), and I found that building the code required all the latest installed software. I guess I'm not surprised that a GNOME-based piece of software is incompatible with my 11-month-old, out-of-the-box R Cubed Linux laptop, but I am disappointed that Gagné didn't make mention of the fact that in order to build and install this software, you had to be willing to re-engineer someone else's code or go out and purchase a new computer with a fresh install of the latest GNOME desktop. Anyway, please relay to Mr Gagné both my interest in his software picks as well as my frustration in those same picks that are incompatible with the bulk of existing Linux installations.

--
Evan Carew

Marcel replies: First off, let me apologize for getting you started on what was obviously a frustrating experience. I make it a point to avoid writing about anything that requires too much effort to build and/or install. It's 2007, and software installation under Linux shouldn't be that hard. There's a reason I stopped giving instructions for building from source in my articles. Binary packages are, for the most part, always a better idea and less frustrating to the end user (programmers and hard-core geeks excepted). Building from source should be a last resort.

I picked Ekiga partly because the bulk of modern Linux distributions either come with Ekiga on the distribution CDs (or DVDs) or provide downloads via their installation programs (such as Synaptic, YaST2, DrakConf and so on). These package managers take care of pesky issues like prerequisites and hunting down missing software by automating the process. I did my Ekiga tests on a Kubuntu system and another running Mandriva. Neither required that I download source and build. Both ran "out of the box" without fuss.

As for the Ekiga Web site, I found none of the problems you experienced when trying to download packages (since I did that as well). As for issues with building GNOME software, what can I say other than "ouch!" [insert appropriate smiley here].

My "mission" in the world of Linux and open-source software is largely to simplify Linux, to demystify its complex image and to show people that they don't have to be computer scientists to take advantage of the benefits that FOSS offers. I want everybody to use Linux, and for that to happen, it needs to be accessible. It's not that hard, and it shouldn't be.

TotalView® Introduces The Most Powerful Command For The Multi-core Age.

TotalView is the proven debugging solution built specifically to address your unique challenges when developing multi-core, multithreaded applications.

As part of a complete suite of proven multi-core debugging and performance tools that supports C, C++ and Fortran on Linux, UNIX and Mac OS X, TotalView 8.1 is the only solution you can count on for all of your multi-core debugging needs. Developed to help debug the world's most demanding applications, TotalView is extremely powerful, yet easy to use. Visually-driven, it provides enhanced graphical representations that enable users to quickly recognize problems and zoom in to identify root causes. TotalView does not require instrumentation, relinking, or rebuilding. TotalView has been proven to reduce debugging time by up to 80%.

Try it now, for free! Go to www.totalviewtech.com/command to power-up your 15-day trial version, or call 1-800-856-3766 for more information.

TOTALVIEW
TECHNOLOGIES

Built for the Multi-core age

Double Negative Double Take

Dear Nicholas: I don't like your /var/opinion column. Please don't stop writing it.

--
Simon

Please fail to stop reading it.—Ed.

The Art of War

I have to admit that I haven't agreed with you very often in your /var/opinion columns, but you hit the bull's-eye in this one [see Nicholas Petreley's "Dealing with the Devil" in the March 2007 issue].

Microsoft is taking a page out of Sun Tzu's *Art Of War*; it likes to keep its enemies close. Microsoft knows that Linux has become 100x the threat that it ever thought it would be, and its philosophy has become one of standing next to the competition in the buffet line to see what they eat next.

I too would love to be able to support Microsoft. Heck, I used to be a shareholder. Until it can play politely in the industry sandbox and stop knocking everyone's sand castles over and attempting to replace them with its shoddy imitations, I will hold Microsoft in contempt.

Great job on the magazine, I've been reading since the very first issue!

--
Gary

Doc's DIY Internet Infrastructure

Congratulations on this wonderful article [see Doc Searls' Linux for Suits column in the March 2007 issue]. It actually changed my mind (I thought of letting my subscription run out—I renewed today).

For a while already, I've thought about building a community-owned fiber network in my town. So far, I've been thrown back, because it is almost impossible to get any one of the big companies (Cisco, etc.) to call me back. And, face it, you need some router, which you cannot buy at Best Buy around the corner. And, you also need some funding. It is hard without even a quote from any of those companies to write a business plan that will be accepted by any bank.

Thanks for that article, it encourages me to pick up that idea again.

--
Henning

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 +1-713-589-2677. E-mail us at subs@linuxjournal.com or reach us via postal mail, Linux Journal, PO Box 980985, Houston, TX 77098-0985 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 to ljeditor@linuxjournal.com or mail them to Linux Journal, 1752 NW Market Street, #200, Seattle, WA 98107 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**.

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

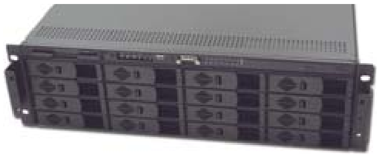
WEB SITE: Read exclusive on-line-only content on *Linux Journal*'s Web site, **www.linuxjournal.com**. Also, select articles from the print magazine are available on-line. Magazine subscribers, digital or print, receive full access to issue archives; please contact Customer Service for further information,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

The Straight Talk People℠
SINCE 1991

ABERDEEN

ABERDEEN STONEHAVEN X318

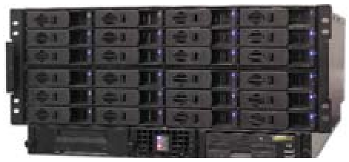


3U 12TB Dual-Core Ready Storage Server

- Up to two Dual-Core AMD Opteron™ 2000 Series processors
- Up to 16 x 750GB (12TB) Hot-Swap SATA Hard Drives
- Internal SATA 2.5" Hard Drive Bay for OS Drive
- 800+ MB/sec sustained data throughput RAID Controller
- Up to 32GB 667/533MHz ECC Registered DDR2 SDRAM
- nVIDIA nForce Pro Chipset with 64-Bit Support
- 650W Redundant Hot-Swap Power Supply
- 5-Year Warranty

Starting at $4,259

ABERDEEN STONEHAVEN X526



5U 18TB Dual-Core Ready Storage Server

- Up to two Dual-Core AMD Opteron™ 2000 Series processors
- Up to 24 x 750GB (18TB) Hot-Swap SATA Hard Drives
- Two Internal SATA Hard Drive Bays for Mirrored OS Drives
- 800+ MB/sec Sustained Data Throughput RAID Controller
- Up to 32GB 667/533MHz ECC Registered DDR2 SDRAM
- nVIDIA nForce Pro Chipset with 64-Bit Support
- 950W Triple Redundant Hot-Swap Power Supply
- 5-Year Warranty

Starting at $5,679

ABERDEEN STONEHAVEN X633

6U 24TB Dual-Core Ready Storage Server

- Up to two Dual-Core AMD Opteron™ 2000 Series processors
- Up to 32 x 750GB (24TB) Hot-Swap SATA Hard Drives
- Two Rear Hot-Swap SATA Hard Drive Bays for Mirrored OS Drives
- 800+ MB/sec sustained data throughput RAID Controller
- Up to 32GB 667/533MHz ECC Registered DDR2 SDRAM
- nVIDIA nForce Pro Chipset with 64-Bit Support
- 1350W Redundant Hot-Swap Power Supply
- 5-Year Warranty

Starting at $7,139

"The Ultimate Linux Server... too fast for our benchmarks... we recommend the Aberdeen line of servers without reservation."
Linux Journal—Aberdeen Stonehaven A261T

"terrific for video serving or other storage intensive tasks"
PC Magazine —Aberdeen XDAS

"Aberdeen surpasses HP ... markedly higher scores ... AberNAS 128 boasts outstanding features"
Network Computing—Aberdeen AberNAS 128

"powerhouse performance ... staggering ... eye-opening ... the highest WebBench numbers to date"
PC Magazine—Aberdeen Stonehaven A261S

AMD, the AMD Arrow logo, AMD Opteron, combinations thereof, are trademarks of Advanced Micro Devices, Inc. For terms and conditions, please see www.aberdeeninc.com/abpoly/abterms.htm. lj019

888-297-7409
www.aberdeeninc.com/linux

diff -u

WHAT'S NEW IN KERNEL DEVELOPMENT

It looks like **MinixFS version 3** will be supported in modern kernels. **Daniel Aragones** has had a patch floating around, and **Andries Brouwer** recently cleaned it up and made it fully kernel-worthy. **Andrew Morton** has said the patch seems harmless and could be accepted at any time.

A new stackable filesystem, called **RAIF** (Redundant Array of Independent Filesystems), is under development by **Nikolai Joukov** and other folks. This filesystem performs data replication across multiple disks like RAID, but it does so on top of any set of other filesystems the user wants to incorporate into the RAIF structure. The project still is not ready to be considered for inclusion in the official kernel, and people brave enough to experiment with it should back up their data beforehand. Still, Nikolai says RAIF has reached a level of stability so that playing with it may be more fun than frustrating.

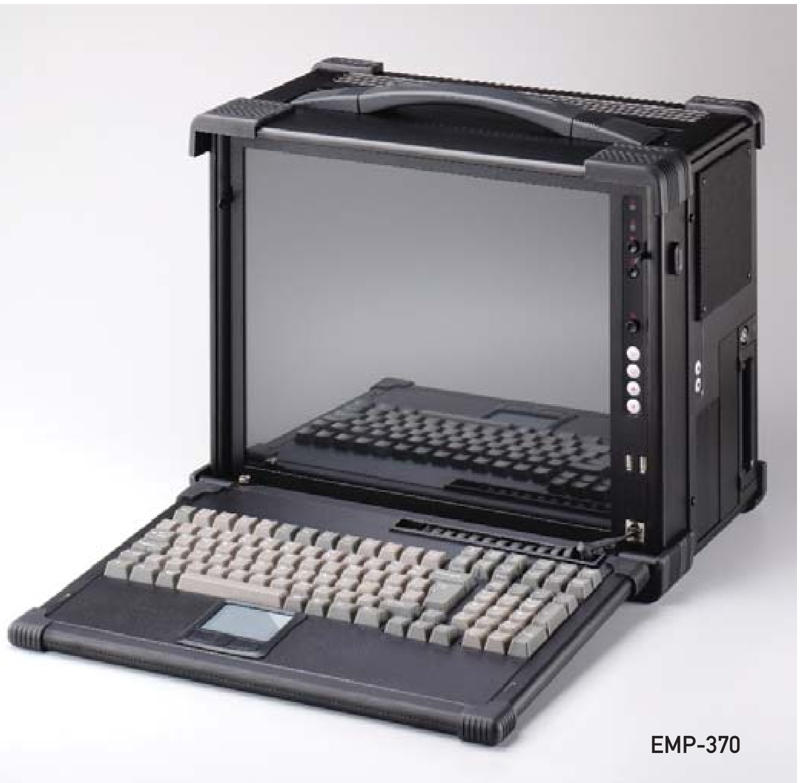
SDHC (Secure Digital High Capacity) Flash cards may soon be supported in the kernel. **Philip Langdale** has seen some very good results (in other words, no lost data in recent tests) with his newly written driver. **John Gilmore** donated an SDHC card for Philip to test on, and the **SD Card Association** has apparently published useful specs, which Philip has put to good use.

The desire of some folks to gain access to **crash reports**, even when running the X Window System, has raised the issue of whether to start migrating graphics drivers into the kernel, instead of leaving them to the X people. This would represent a fairly massive shift in kernel development, but it may be the best way to ensure that oops information is properly displayed, regardless of what graphics mode a system happens to be in when it crashes. **D. Hazelton** and probably others are working on this, but there definitely would be massive and long-term flame wars before any such transition could occur in the kernel.

Karel Zak has decided to fork **util-linux** away from the current maintainer **Adrian Bunk**, after he and others were unable to get a response from him regarding their patches and proposals to change maintainership. Unless Adrian decides to make a claim that he should stay as the official maintainer, it's likely that Karel eventually will reintegrate his code with Adrian's, having only a single code base once again with Karel as maintainer.

Greg Kroah-Hartman has started a **mailing list** for anyone who packages the kernel for a distribution. The purpose of the list is to provide a vendor-neutral place to discuss bugs and other issues associated with packaging kernels. It also may provide a more visible way for kernel packagers to submit their own changes upstream to the developers.

—ZACK BROWN


EMP-370

Return of the Luggable

If Linux is the ultimate hermit crab operating system—born without a home but able to live well in anybody's hardware—it'll be fun to see how well Linux lives in the funky form factors provided by Acme Portable Machines, Inc.

Acme's hardware is built for places where the practical outweighs the pretty by ratios that verge on the absolute: server rooms, factory floors, military aircraft and medical facilities. Acme makes KVM (keyboard/video/mouse) switches that slide on rails into racks and control multiple CPUs (among many other connected things). It makes kiosk PCs. It makes tower CPUs with the dimensions of portable sewing machines, featuring flat screens and keyboards that open out the side and slots inside that hold up to eight full-size cards.

But, perhaps the most interesting items are Acme's portable systems—luggable workstations that look like scary briefcases and boast features such as "flame resistant" cases. Acme calmly calls its EMP "a robust lunchbox computer built using heavy-duty metal to provide a tough, go-anywhere unit ideally suitable for harsh/severe environments and mission-critical applications". Jim Thompson of Netgate calls it "the ultimate LAN party box".

Acme Portable's primary market is OEMs, but you can see and buy its goods directly at **acmeportable.com**.

—DOC SEARLS

Fibre
Channel
Storage
Invoice
EtherDrive
Storage
Le$$ than a
cup of coffee!

Unlocking the Music Locker Business

Thanks to Linux and other open-source building materials, on-line storage has become a commodity. Last year, Amazon made it available in a general way through S3. Then, in February 2007, Michael Robertson's MP3tunes did the same for one vertical breed of storage, the music locker. The name of the service is Oboe, and it stores unlimited amounts of audio for you. The free service is supported by advertising. The $39.95 US annual service offers customer support, allows bigger files and works as an offsite NAS.

After the Oboe news broke, we caught up with Michael for a brief interview.

LJ: Are the servers you're using Linux ones? (I assume so, but need to ask.)

MR: Yes, of course. No Microsoft in the house. With massive storage needs, it's imperative we have an industry-leading cost structure, and you get that only with LAMP. You can't be at a cost disadvantage to your competitors.

If you look at the battle for e-mail, it's a great illustration. Hotmail had expensive EMC hardware that let it get scooped by Yahoo, which was based on Veritas. Then Google raised the ante, and thanks to LAMP technology, it was able to make a storage offer that was, and is, costly for Yahoo and Microsoft to match. Music lockers are even more storage-intensive, so cost is an even bigger issue. Relying on LAMP lets us ride the cost efficiency of declining storage costs.

LJ: What can you tell us about the other technology involved?

MR: CentOS—three hundred terabytes of storage. We deploy new servers every week. We're standardized on 750GB drives, but we'll move to 1TB drives later this year. A big issue for us is power and floor space, not processing performance.

LJ: Your music lockers seem similar in some ways to Amazon's S3, the unlimited storage of which is being used as a business back end for companies that sell offsite backup. Is there any chance you'll look to provide a back end to ad hoc or independent music services? What I'm thinking here is that a big back end can make possible a number of businesses that do not yet exist, and open the music industry to much more entrepreneurship and competition at every level.

MR: Yes, there are similarities but also differences. Both services have APIs that open them to a wide range of software and hardware applications (see **mp3tunes.com/api**). But, MP3tunes is tailored to music delivery. And, I'd contend that music is a very unique data type because of its repeat usage from many locations. So, built in to the API are a wealth of audio-related features, such as transcoding formats, down-sampling bit rates, meta-tags and cover art. Here's an example: Mp3tunes.com is a mobile interface that can stream or downloads songs from your locker directly to mobile devices, seamlessly changing format and bit rate.

LJ: Let's look at the economics of this. I easily can see partnerships with the likes of Sonos, which are also Linux-based. Seems like $40 US for a service like yours is an argument against customers maintaining their own NAS (network attached storage) device.

MR: Yes, expecting people to manage their own NASes is like expecting people to run their own backyard power generator. It's just dumb. You'll have better service and greater cost efficiency by using a centralized system that can take advantage of economies of scale. We have talked to Sonos, and there's interest. I hope they support our API.

—DOC SEARLS

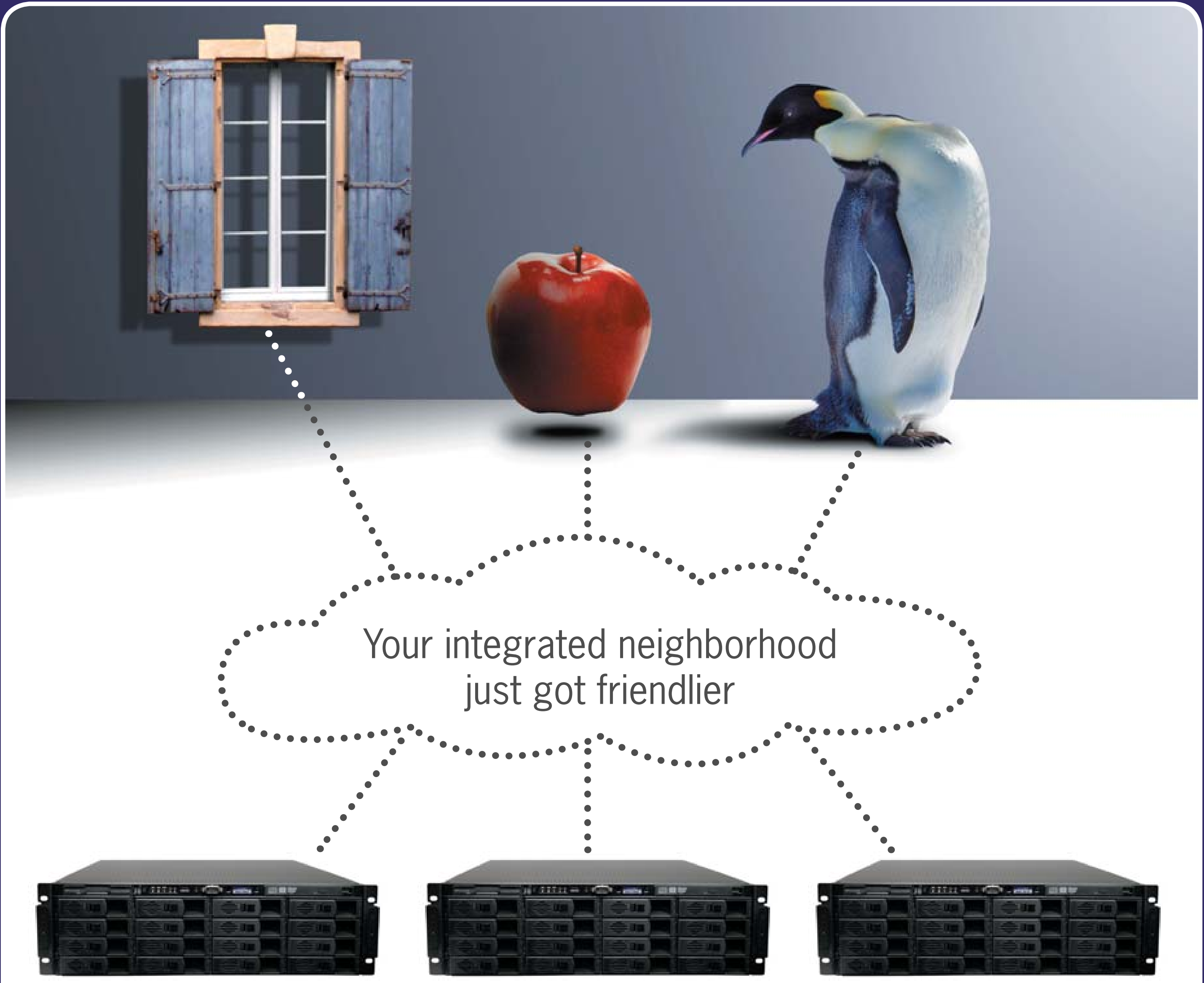
USER FRIENDLY by J.D. "Illiad" Frazer
LINUX JOURNAL EDITION
LITTLE BIT OF JAVASCRIPT HERE...LITTLE BIT THERE, AND THE NAVBAR IS ALL SMOOTH AND SHINEY...
COPYRIGHT © 2007 J.D. "Illiad" Frazer HTTP://WWW.USERFRIENDLY.ORG/
WHAT IS THAT AND WHY IS IT ON OUR WEBSITE?
IT'S JUST SOME AJAX, SID.
AHH, YOU MEAN "USER-INITIATED BLINK TAGS."
I LIKEN IT TO "MENU-BASED BROWSER GAMES."

LJ Index, May 2007

1. Billions of US dollars cable operators will spend by 2012 improving digital network capacity: 80
2. Millions of US dollars quoted in the fiber-based build-out of a San Francisco municipal high-speed Internet utility: 500
3. Percentage rate of increase in fiber-to-the-home (FTTH) subscriptions in Japan: 88
4. Millions of Japanese FTTH subscriptions in March 2005: 5.4
5. Effective radiated power in watts of KRUU "open-source radio": 100
6. Range in miles of KRUU's city-grade signal: 4
7. Number of planets served by KRUU's live Web stream: 1
8. Total dollars paid to AT&T for continuous use of a push-button phone since the 1960s by an 88-year-old: 7,500 US
9. Millions of cars at the end of 2006: 800
10. Millions of PCs at the end of 2006: 850
11. Billions of Internet connections at the end of 2006: 1.1
12. Billions of credit cards at the end of 2006: 1.4
13. Billions of TVs at the end of 2006: 1.5
14. Billions of cell phones at the end of 2006: 2.7
15. Billions of cell phones in use by September 2006: 2.5
16. Millions of cell phones added in the prior year: 484
17. Percentage of new cell phones added in Asia: 41
18. Billions of cell phones expected by the end of 2007: 3
19. Projected billions in annual cell phone shipments in 2008: 1
20. Estimated billions of human beings in July 2006: 6.525170264

Sources: 1: ABI Research | 2: "Fiber Optics for Government and Public Broadband: A Feasibility Study Prepared for the City and County of San Francisco, January 2007", by Communications Engineering & Analysis for the Public Interest | 3, 4: *Broadband Properties*, December 2006 | 5: FCCInfo.com 6: radio-locator.com | 7: KRUU | 8: *The Consumerist* 9–14: Tomi T. Ahonen and Alan Moore in *Communities Dominate Brands* | 15–18: *Wireless Intelligence*, via *Cellular News* | 19: Gartner via windowsfordevices.com 20: CIA's *World Factbook*

—Doc Searls



NAS 3004i
4TB Raw SATA Storage

NAS 3008i
8TB Raw SATA Storage

NAS 3012i
12TB Raw SATA Storage

>>>>> **From $7899** <<<<<

With Pogo's Network Attached Storage appliance, maximum interoperability with the most popular operating systems is easier. Take control of your ever-expanding storage needs with an easy to use graphical management interface, multiple snapshots, built-in back up, IP failover, and enhanced security. Powered by Intel Dual-Core Xeon processors and a fast 64-bit OS. Call today for more information.

www.pogolinux.com

pogo [linux]™

Experience, Imagination, and Support.
Pogo Linux is Hardware Solutions Built for Linux, Built for You.

To get started, contact us at 888.828.POGO or inquiries07@pogolinux.com
Pogo Linux, Inc. 701 Fifth Ave. Suite 6850, Seattle, WA 98104

intel Xeon inside™
Quad-core. Unmatched.

Parallel NFS (pNFS) Bridges to a Mature Standard

Thanks to the emergence of low-cost Linux clusters, high-performance computing (HPC) is no longer the domain solely of an elite group of public-sector-funded laboratories. In fact, HPC now can be found addressing challenges as diverse as simulating the behavior of the entire earth to the simulation needs of an individual product designer.

But, as clusters have become more prevalent, new challenges have emerged. The first challenge is to define a storage and I/O architecture that is not only capable of handling the vast amount of data created and consumed by these powerful compute engines, but that also is capable of keeping those engines fully utilized and fed with data. Without data, the largest and fastest supercomputers become nothing more than expensive space heaters.

The second challenge revolves around making the data generated by clusters easily available to other systems and users outside the cluster itself. Copying or moving data to other systems is clearly an option but involves inherent overhead cost and complexity. Ideally, any node on the network should be able to access and process data where it resides on the cluster.

Initially, clusters used the ubiquitous NFS standard, which has the advantages of being well understood, almost universally supported by many vendors and of providing easy access to data for systems and users outside the cluster. However, NFS moves all data and metadata through a single network endpoint (server) that quickly creates a bottleneck when trying to cater to the I/O needs of a cluster. The result is that neither bandwidth nor storage capacity scales—a new solution is required.

Parallel Filesystems and Parallel NFS

Parallel filesystems, which enable parallel access directly from server nodes to storage devices, have proven to be the leading solution to this scalability challenge. Although parallel filesystems are relatively new, the technology clearly will become an essential component of every medium- to large-scale cluster during the next few years. Several parallel filesystem solutions are available today from vendors such as Panasas (ActiveScale PanFS), IBM (GPFS), EMC (HighRoad) and Cluster File Systems (Lustre).

Government, academic and Fortune 500 customers from all over the globe have embraced parallel filesystem solutions; however, these solutions require that customers lock in to a particular vendor for the software and sometimes the hardware. Wouldn't it be nice to have a filesystem that has the same performance as these vendor-specific solutions but that is also a true open standard? Then, you could reap the performance benefits of parallel access to your data while enjoying the flexibility and freedom of choice that come from deploying a universally accepted standard filesystem.

This introductory article discusses Parallel NFS (pNFS), which is being developed to meet these needs. pNFS is a major revamp to the NFS standard and has gained nearly universal support from the NFS community.

Parallel NFS Origins

When people first hear about pNFS, sometimes their initial reaction is that it is an attempt to shoehorn a parallel capability into the existing NFS standard. In reality, it is the next step in the evolution of NFS with the understanding that organizations need more performance while keeping it a multivendor standard. The NFSv4.1 draft standard contains a draft specification for pNFS that is being developed and demonstrated now.

Panasas is the author of the original pNFS proposal. Since this original proposal was written, a number of other vendors, notably EMC, IBM, Network Appliance and Sun, have joined to help define and extend pNFS. Other vendors are contributing as well, so pNFS is gaining broad momentum among vendors.

Because pNFS is an evolution of the NFS standard, it will allow organizations that are comfortable with NFS to achieve parallel performance with a minimum of changes. Plus, because it will become part of the NFS standard, it can be used to mount the cluster filesystem on the desktop easily.

Architecture of pNFS

NFSv4.0 improved the security model from NFSv3.0, which is the most widely deployed version today, and it folds in file locking that was previously implemented under a different protocol. NFSv4.0 has an extensible architecture to allow easier evolution of the standard. For example, the proposed NFSv4.1 standard evolves NFS to include a high-speed parallel filesystem. The basic architecture of pNFS is shown in Figure 1.

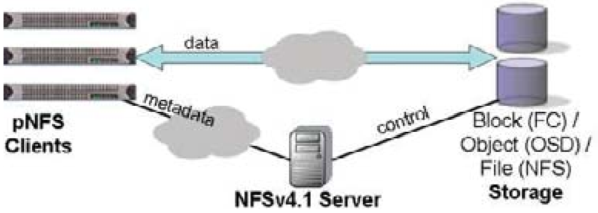


Figure 1. pNFS Architecture

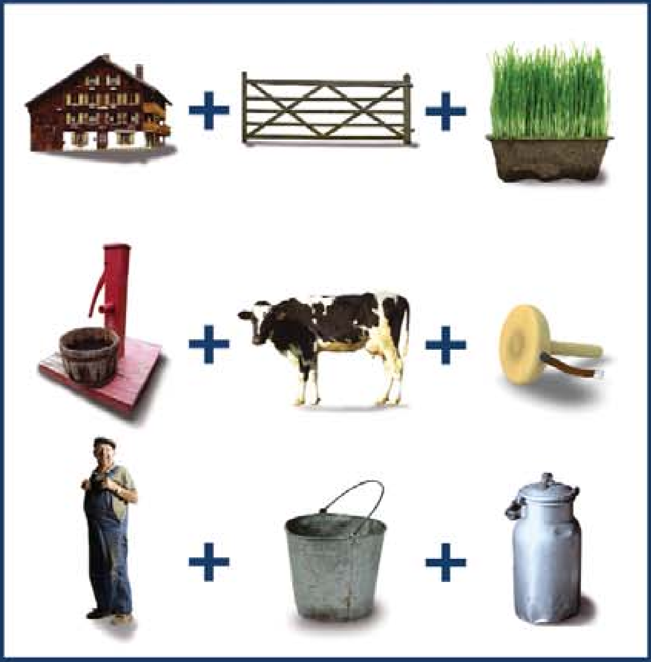
The pNFS clients mount the filesystem. When they access a file on the filesystem, they make a request to the NFSv4.1 metadata server that passes a layout back to the client. A layout is an abstraction that describes where a file is located on the storage devices. Once the client has the layout, it accesses the data directly on the storage device(s), removing the metadata server from the actual data access process. When the client is done, it sends the layout back to the metadata server in the event that any changes were made to the file.

This approach may seem familiar, because both Panasas (ActiveScale PanFS) and Cluster File System (Lustre) use the same basic asymmetric metadata access approach with their respective filesystems. It is attractive because it gets the metadata server out of the middle of the data transaction to improve performance. It also allows for either direct or parallel data access, resulting in flexibility and performance.

Currently, three types of storage devices will be supported as part of pNFS: block storage (usually associated with SANs, such as EMC and IBM), object storage devices (such as Panasas and Lustre) and file storage (usually associated with NFS file servers, such as NetApp). The layout that is passed back to the client is used to access the storage devices. The client needs a layout driver so that it can communicate with any of these three storage devices or possibly a combination of the devices at any one time. These storage devices can be products such as an EMC SAN, a Panasas ActiveScale Storage Cluster, an IBM GPFS system, NetApp filers or any other storage systems that use block storage, object storage or file storage. As part of the overall architecture, it is intended to have standard, open-source drivers (layout drivers) for block storage, object storage and file storage back ends. There will be other back ends as well. For example, PVFS2 was used in the first pNFS prototype as the back-end storage.

How the data is actually transmitted between the storage devices and the clients is defined elsewhere. It will be possible for the data to be communicated using RDMA (Remote Direct Memory Access) protocols for better performance. For example, the InfiniBand SDP protocol could be used to transmit the data. The data can be transmitted using SCSI

➤



+
+
+
+
+
+

MILK

➤

Block Command (SBC) over Fibre Channel or SCSI Object-based Storage Device (OSD) over iSCSI or using Network File System (NFS).

The "control" protocol shown in Figure 1 between the metadata server and the storage is also defined elsewhere. For example, it could be an OSD over iSCSI.

The fact that the control protocol and the data transfer protocols are defined elsewhere gives great flexibility to the vendors. It allows them to add their value to pNFS to improve performance, improve manageability, improve fault tolerance or add any feature they want to address as long as they follow the NFSv4.1 standard.

Avoiding Vendor Lock-in

A natural question people ask is "how does the proposed pNFS standard avoid vendor lock-in?" One of the primary aspects of pNFS is that it has a common filesystem client regardless of the underlying storage architecture. The only thing needed for a specific vendor's storage system is a layout driver. This is very similar to how other hardware is used in Linux—you use a driver to allow the kernel to access the hardware.

Parallel NFS also works well for the vendors because it allows their storage to work with a variety of operating systems without porting their whole proprietary filesystem stack. Because NFSv4.1 will be a standard, the basic client would be available on a variety of operating systems as long as the OS had the client. The only piece the vendor would have to provide is the driver. Writing drivers is generally an easier process than porting and supporting a complete filesystem stack to various operating systems.

If you have a current parallel filesystem from one of the storage vendors, what does pNFS do for you that the vendor does not? Initially, pNFS is likely to perform more slowly than a proprietary filesystem, but the performance will increase as experience is gained and the standard pNFS client matures. More important, pNFS allows you to mount the filesystem on your desktop with the same performance that the cluster enjoys. Plus, if you want to expand your storage system, you can buy from any vendor that provides a driver for NFSv4.1. This allows your existing clients to access new storage systems just as your computers today access NFS servers from different vendors, using the filesystem client software that comes with your UNIX or Linux operating system.

Parting Comments

Parallel NFS is well on its way to becoming a standard. It's currently in the prototyping stage, and interoperability testing is being performed by various participants. It is hoped that sometime in 2007 it will be adopted as the new NFS standard and will be available in a number of operating systems.

If you want to experiment with pNFS now, the Center for Information Technology Integration (CITI) has some Linux 2.6 kernel patches that use PVFS2 for storage (**www.citi.umich.edu/projects/asci/pnfs/linux**).

—LARRY JONES

KRUU Models Open-Source Radio

KRUU is a community FM station of the new "low power" breed that is intended to serve local areas with noncommercial programming (**www.fcc.gov/mb/audio/lpfm**). As the FCC puts it, "The approximate service range of a 100-watt LPFM station is 5.6 kilometers (3.5 miles radius)". In KRUU's case, that range nicely covers the town of Fairfield, Iowa. The station's Web stream, however, is unconstrained by those physical limitations. I listen to it in Santa Barbara, and it's already one of my faves.

The station's About page tells why it's especially relevant and cool:

> KRUU's commitment to community also extends to the software and systems that are in place at the station. All the computing infrastructure uses only Free Software (also sometimes termed open-source software). The Free in this case is in reference to freedom, and not cost—all the software comes with the underlying source code, and we contribute all our changes, edits and suggestions back to the Free Software community. The reasons for using Free Software go far beyond the scope of cost. KRUU wishes to build local knowledge using systems that do not impose restrictions or limitations on use. To this end, we support software that is licensed under a "copyleft" or "open" clause, and content that is licensed under Creative Commons.

The word "open" runs through everything KRUU plays and values. From 5–6 am, seven days a week, it runs the "Open Source Radio Hour". And, that's just the tip of the freeberg.

For listening with Linux, it profiles XMMS, Banshee, Amarok, VLC and Rhythmbox. KRUU streams in MP3, but it also podcasts in Ogg Vorbis. I listen to a lot of radio on-line, and I don't know of a station that's more committed to free software and open-source values than this little one. Find them at **kruufm.org**.

—DOC SEARLS

They Said It

I think that novels that leave out technology misrepresent life as badly as Victorians misrepresented life by leaving out sex.
—Kurt Vonnegut, *A Man Without a Country* (Random House, 2005)

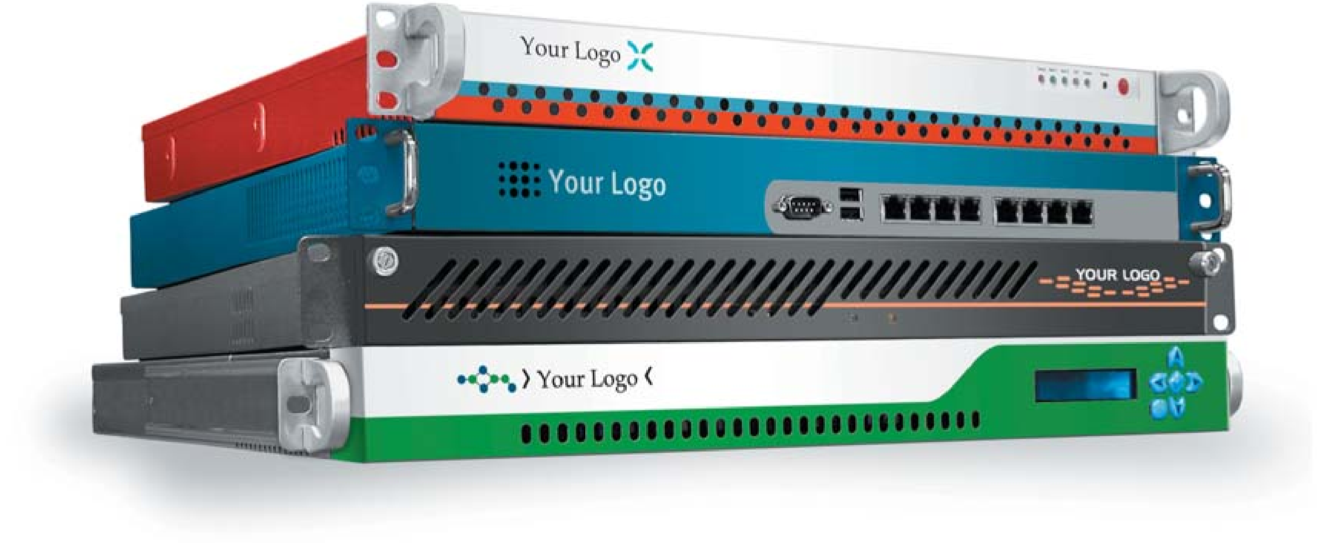
Truth arrives in conversation. Not solitude....Those of us who resist...the privatizing of the cultural commons, might ask ourselves what it is we are trying to protect. For the content industries, the tools of enclosure (copyright term extension, digital rights management, etc.) have been called into being to protect and enhance revenues; it's about the money. For we who resist, the money is the least of it. Protecting the cultural commons means protecting certain ways of being human.
—Lewis Hyde, in a talk to Berkman Center, February 13, 2007

This song is Copyrighted in the U.S., under Seal of Copyright # 154085, for a period of 28 years, and anybody caught singin' it without our permission will be mighty good friends of ourn, cause we don't give a dern. Publish it. Write it. Sing it. Swing to it. Yodel it. We wrote it, that's all we wanted to do.
—Woody Guthrie, www.publicknowledge.org/resources/quotes

I have often reaped what others have sowed. My work is the work of a collective being that bears the name of Goethe.
—Johann Wolfgang Goethe, www.publicknowledge.org/resources/quotes

The preservation of the means of knowledge among the lowest ranks is of more importance to the public than all the property of the rich men in the country.
—John Adams, www.publicknowledge.org/resources/quote

The Industry Leader for Server Appliances



Custom server appliances or off the shelf reference platforms,
built with your image and software starting under $1,000.
From design to deployment, we handle it all.

Delivering an appliance requires the right partner. MBX Systems is the right partner. We understand that by putting your name on our hardware, you're putting your reputation in our hands. We take that seriously. We provide the services you need to support your customers. Better than the competition. You never even need to touch the hardware. Engineering. Design. Deployment. We handle it all, so you can focus on what's important to you. Your software. Your sales. Your success.

Visit us at www.mbx.com or call 1-800-681-0016 today.

MBX
systems

www.mbx.com | 1-800-681-0016

TECH TIPS

Mutt macros make e-mail sorting and navigation a breeze, and a handy script to help you find other scripts.

›› An Easy Way for Mutt Users to Move Mail into Designated Folders

You'd think with all the wonderful GUI IMAP e-mail clients available, the character-based Mutt wouldn't have much appeal. Yet, a number of us at *Linux Journal* are still hooked on Mutt for at least part of our e-mail usage. One reason is that you can configure Mutt to behave just about any way you like.

Ideally, you can write server-side filters to sort your mail into the appropriate folders, and I have a number of filters to do just that. However, the filters aren't perfect, and many news alerts, press releases and the like end up in my inbox. I can spot them from the subject lines. So, I have created a long list of macros for Mutt to file away mail into specific folders from the inbox index. I can tuck away mail with a single Ctrl-keystroke, which beats the heck out of dragging and dropping mail to a sidebar with a folder tree (or worse, a cascading "move" menu). Here's a short sample of my list of macro keystrokes. You can put your own set in the .muttrc configuration file in your home directory.

The following list of macros lets me press Ctrl-R to send the currently highlighted message to my Read folder (that's Read as in past tense), Ctrl-P to send it to my Press-Releases folder (which is a subfolder of Folders), Ctrl-N to send it to News-Alerts, and so on. As you can see, I've tried to associate the letter with the action to make the keystrokes easy to remember (Ctrl-K kills the message to the SPAM bin):

```
macro index \Cr "s=Read\r"
macro index \Cp "s=Folders/Press-Releases\r"
macro index \Cn "s=Folders/News-Alerts\r"
macro index \Ch "s=Folders/Humor\r"
macro index \Ck "s=SPAM\r"
```

I also spend a lot of time stepping through messages with the preview pane open though. The above list of macros won't work in preview mode. If you want to be able to do the same operations while previewing mail with the pager, add a duplicate list like this:

```
macro pager \Cr "s=Read\r"
macro pager \Cp "s=Folders/Press-Releases\r"
macro pager \Cn "s=Folders/News-Alerts\r"
macro pager \Ch "s=Folders/Humor\r"
macro pager \Ck "s=SPAM\r"
```

Mutt gives you a way to navigate through your folders, but this is one case when it's not as easy as GUI clients that provide a sidebar folder tree where you can click on the folder you want to open. Here are some macros to navigate to your most commonly used folders to read the messages you've filed away:

```
macro index ",r"   "c=Read\nOd="
macro index ",p"   "c=Folders/Press-Releases\nOd="
macro index ",n"   "c=Folders/News-Alerts\nOd="
macro index ",h"   "c=Folders/Humor\nOd="
macro index ",k"   "c=SPAM\nOd="
```

I've gotten used to thinking of the comma key as my "go-to" key, so I can go to the Press-Releases subfolder by typing the keys , and then p. The combination ,n takes me to News-Alerts, and so on. If the comma isn't intuitive for you, pick another keystroke and modify the macros accordingly.

Naturally, you'll want to be able to get back to the inbox easily. So, I do that with the ,i combination. Once again, you may want to create a duplicate that works if you are using the preview pager. Simply substitute the address of your IMAP server for <yourmailserver>, and it should work for you:

```
macro index ",i"   "cimap://<yourmailserver>/INBOX\nOd="
macro pager ",i"   "cimap://<yourmailserver>/INBOX\nOd="
```

One final note: you may be wondering why each of the navigation macros ends with \n and then Od=. The \n executes the move to the new folder, and the 0d= tells Mutt to sort the messages by date. That way, if you have changed the sort order of messages while reading your mail, Mutt always will use the date sort when you change folders with the macro command.

You can find all the info you need to customize your copy of Mutt in the Mutt manual on-line at **www.mutt.org/doc/manual**. It may take a bit of work getting Mutt to perform just the way you like, but once you've got it customized to your tastes, you may find it very difficult to go back to using a GUI e-mail client.
—Nicholas Petreley

›› Finding a Needle in a Haystack of Scripts

This script is the combined effort of *Linux Journal* Webmaster Keith Daniels, whose work was modified by a reader named Karl, who gave us this submission. It lets you find a script based on a keyword you might recall that occurs in the script.

Objectives:

1. Don't force the four-line restriction on the length of each script's header.

2. Allow multiple search paths (not only ~/bin/).

3. Support display of the script's path/name without hard-coding it into the header.

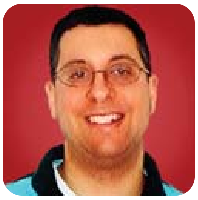
```
#!/bin/bash
```

```
#-
#- NAME:    ~/bin/scriptsearch
#- PURPOSE: grep for patterns in all scripts
#  under preset search
#  paths; output header lines in each matching script
#- NOTES:   all lines beginning with '#-' are assumed
#  to be header lines
#- USAGE:   scriptsearch <grep pattern>
#-

# paths in which all scripts will be searched
SEARCH_PATHS="${HOME}/bin /usr/local/bin"

if [ ${1} ]; then
    for PATH in ${SEARCH_PATHS}; do
        echo -e "\n...searching ${PATH}...\n"
        # find list of matching files for current
        # search path
        MATCHES="`/bin/grep -li $1 ${PATH}/*`"

        for MATCH in ${MATCHES}; do
            # print summary for each matching file
            echo "#====<script> ${MATCH}"
            /bin/grep -i '^#-' ${MATCH}
            echo -e "#====</script>\n"
        done
    done
fi
```

Note that a side effect of the way I handled the first objective is that my Perl scripts, which often have a print_usage() function with a `print qq{...}` spanning multiple lines can be searched as well without duplicating the print_usage() function. Simply prefix each line in the qq{...} with #-.

I added the -i option to the grep command for ${MATCHES}. It's a very simple change but quite important, as I don't want to lose relevant results simply because my search keywords are lowercase but the script contains matches with uppercase characters (var names and comments might be our memory cues for finding the script, and they commonly contain uppercase).
—Karl Erisman and Keith Daniels■

Linux Journal pays $100 for reader-contributed tech tips we publish. Send your tips and contact information to techtips@linuxjournal.com.



Firebug

Firebug is a brilliant means of debugging Ajax applications.

REUVEN M. LERNER

During the past year or two, Web developers have witnessed what we might call the JavaScript renaissance. That's because all the fancy Ajax, Web 2.0, mashup, interactive, collaborative, desktop-like applications that are being developed are written in JavaScript. This is possible today not only because the JavaScript language has improved, but also because browsers are increasingly compliant with Web standards. And, of course, the availability of cross-platform JavaScript libraries, such as Prototype, has added many features to the language, while simultaneously ensuring cross-platform compatibility.

So, with all the JavaScript development happening today, what is the most popular way to debug programs? That's right, it's the built-in alert function:

```
alert("value of x = '" + x + "'");
```

Alerts might be unpleasant, ugly and downright annoying, but they have been the best and easiest way to debug JavaScript for several years. Sure, there have been a few JavaScript debuggers, but none of them has been all that exciting to use, let alone easy or productive.

Well, I'm happy to say that the situation has now changed. Firebug is an open-source plugin for the Firefox browser that aims to be a one-stop debugging tool not only for JavaScript, but also for everything that a Web developer needs. Written by Joe Hewitt, one of the founders of the small startup Parakey, Firebug 1.0 was released in early 2007. It already has become wildly popular among Web developers, and for good reason.

This month, we look at Firebug, so that we can debug, inspect and optimize modern Web pages. Firebug already has improved my ability to debug modern Web pages dramatically, and I wouldn't be surprised if this turns out to be the case for many other Web developers.

Installing Firebug

Firebug is distributed as an extension for the Firefox Web browser. It is most easily downloaded and installed from the Firebug site (**www.getfirebug.com**). To install it, click on the download Firebug button. If you already have told Firefox this is an allowed download site, you will be able to download and install this Mozilla extension. If not, you need to add getfirebug.com to your list of trusted download sites, and then repeat the download procedures. Once the extension is installed, restart Firefox.

Once you do this, your Web browser will look much the same as before, but with some small changes. First, there now will be an icon at the bottom of the screen in the status line. This icon will look like either a green V in a circle (to indicate that it is running) or a gray circle with a slash through it (to indicate that it is disabled). Firebug can be enabled all of the time, but you're probably interested in debugging only a small number of sites that you visit. Thus, it's useful that by clicking on the Firebug icon—or by going to Tools→Firebug in the Firefox menu—you can indicate the sites for which Firebug should be active.

You can add a new site to this list by selecting open Firebug from that same Tools→Firebug menu, or by adding it manually with the allowed sites dialog box from that same Firebug menu. In either case, the site you currently are visiting will be viewable in Firebug.

Now that we have started Firebug, what can we do with it? Let's have some fun by going to the *Linux Journal* site (**www.linuxjournal.com**). Activate Firebug for this site, and your browser window will be cut in half, with the top half still showing the Web page and the bottom half containing Firebug. I generally prefer to work with Firebug in this way, but if you prefer to keep your browser window separate from your debugging window, you might want to choose open Firebug in new window, rather than simply open Firebug.

The main menu for Firebug contains the Firebug icon, which offers most of the same menu options as the icon in the status line and Tools→Firebug menu, along with links to the Firebug documentation and home page. An Inspect button always sits next to that icon, and it lets you zoom in on a particular item on the page. The rest of that menu bar changes according to the context in which you are operating, which is determined by the second row of buttons, marked Console, HTML, CSS, Script, Dom and Net.

Debugging HTML

One of the first, and easiest, tasks to take on with Firebug is debugging HTML. Click on the HTML button and choose Inspect. You immediately will see the HTML source code for the current page highlighted in the Firebug window.

Now, here's where the magic begins: with HTML/Inspect selected in Firebug, move your cursor over the Web page (the upper frame). As you move the cursor, the HTML element over which it is passing is highlighted in blue. In the Firebug frame, the HTML source corresponding to that rendered content also is highlighted.

This functionality is particularly useful when I know something is going wrong with the rendering of my Web page, but I'm not quite sure which part of the HTML source is to blame. A few clicks of the mouse later, and you easily can know which part of the file you need to edit.

Firebug highlights the HTML source code that it displays, albeit using different colors than the View Source page that Web developers know and love. (I think that Firebug's color choices are better.) Moreover, Firebird displays the HTML source as a tree, including indentation. This, along with a display of all of the current element's parent tags (next to the edit button on the top row) provides a great sense of the current element's context in the document.

Inspecting the HTML certainly is useful and interesting, but it gets better. If you double-click on the text in the Firebug frame, you now can edit it. Obviously, your changes do not get saved back to the server, meaning you can return to the original content by refreshing the page. Nevertheless, it is quite useful (and fun!) to replace text on existing pages, right from your browser.

Replacing text in an HTML element is a good start, but what if we want to modify the markup itself, rather than only its text or attributes? Right-clicking on the tag (or any of the text within the tag) displays a pop-up menu, letting you copy the HTML, innerHTML or XPATH of the selected tag. You can ask for the selected tag to be displayed in the top frame, scrolling if necessary in order to reach it.

Finally, you can add new attributes to this element or ask to inspect the element in the DOM tab. And indeed, the DOM view provides another way of looking at the same document. While still inspecting one of the HTML elements, click on the DOM button in the second row of the Firebug frame. The frame changes its appearance, listing a large number of DOM attributes associated with the element. Thus, inspecting an image in the DOM tab shows that its type is IMG, while inspecting a link shows that it is of type A. As always, Firebug lets you edit any attribute you like by clicking on the value and replacing it.

Debugging CSS

So far, we have seen how Firebug can help inspect and modify HTML. But, of course, HTML provides only the basic content and structure for a page; if you know what you're doing, you can inspect and modify the CSS definitions as well.

One of the many amazing and clever things about CSS is the way it handles inheritance. If you have set some attributes for <p> in css1.css and others in css2.css, both will apply to <p> tags in your file. Things can become more complicated than that, such as when you have a <p> tag with an id attribute, with conflicting style information. In these cases, the most specific style (that is, the id tag) applies.

On a large site with many different styles, it sometimes can become difficult to keep track of which styles are being applied. Fortunately, Firebug provides some wonderful capabilities for inspecting the CSS associated with a site, and even for editing it.

To use this functionality, click on the CSS button (next to the HTML button we were using earlier), and then on the Inspect button above it. Firebug continues to show its tree representation of the current document, but the right-hand frame displays all of the CSS styles that apply to whatever you're pointing to. Moreover, it indicates which CSS file, and which line of that file was applied. And, as if that isn't enough, it crosses out any styles that were overridden by more specific ones.

As with the HTML inspector, you easily can edit the CSS associated with any element by double-clicking on a style declaration. For example, if you want to change text from roman to italic type, you merely click on CSS/Inspect, then point to the text you want to change. If there already is a font-style property in the CSS, you can change it to read italic. If not, you can right-click on the CSS pane, choose new property, and add the property and its value.

Firebug knows what the legal property names are for CSS styles, so it is able to complete the style name automatically when you begin typing.

The right-hand frame offers more than merely a textual indication of styles. It also has a graphical representation of the CSS box model, showing the number of pixels used by the element, padding, border, margin and offset.

Debugging JavaScript

Finally, we get to the JavaScript debugger. As I mentioned earlier, JavaScript is becoming a mainstream application programming language, which means people are writing increasingly complex programs with it. And, although I've long used print statements for much of my debugging during the years, there's no doubt that a good, interactive debugging environment can make it easier to solve certain issues.

Firebug provides several powerful tools

for JavaScript debugging. It introduces a new logging system that makes it possible to produce debugging output without using the alert function. I still can't quite believe it took this long for someone to realize it would be helpful to have a method for logging that didn't produce a modal dialog box. Regardless of how long it took, we can now use console.log to write messages to the Firebug console, such as:

```
console.log("Now executing the 'foo' function.")
```

Or:

```
console.log("Username parameter was '" + username + "'")
```

Actually, Firebug makes it possible to do much better than this, with embedded, printf-like strings:

```
console.log("Username parameter was '%s'", username)
```

Firebug provides a set of logging functions, each of which produces output with a different warning level and color. These are:

- console.debug
- console.info
- console.warn
- console.error

The output from these methods appears in the Firebug console, which you can view by clicking on the Console button, right under the Firebug icon. Note that using these methods means your methods might be incompatible with browsers that lack Firebug.

The console is not only an area that receives messages though. As with Ruby's IRB and Python's interactive mode, Firebug's console allows you to execute JavaScript within the context of the page. Interested in getting a list of forms on the page? Simply type:

```
document.forms
```

and you will get a list of the forms. You also can use Prototype-style methods to retrieve information about much of the page, as in:

```
$('cvb_form')
```

which, on the *Linux Journal* home page, produces the following form header:

```
<form id="cvb_form" action="/" method="get">
```

Firebug also has extensive capabilities for monitoring functions, setting breakpoints and profiling functions. Go to the Script button, and you are greeted by the source code for an included JavaScript file. (You can change the file you're looking at by choosing from a menu.) Pointing to a line of code with the mouse cursor brings up information about the current state of those objects.

Clicking on the code allows you to set a breakpoint (including a conditional breakpoint) to copy the function (useful when trying to communicate with other programmers or even when writing a column about programming) or to handle a breakpoint by continuing or stepping through the code.

To find out which JavaScript functions have been taking the most time, and which are invoked most frequently, use Firebug's profiler. It's amazingly simple to use. Click on the Console/Profile button, and then go about your business, using the Web site as much or as little as you want. When you click the Profile button again, Firebug reports the number of times each JavaScript function was called, the time each call took and where each function is defined.

Finally, the Net button makes it easy to keep track of any Ajax calls embedded in the page. Clicking on Net and XHR produces a graphical indication of the Ajax calls that have been made on this page, as well as how long each took to execute. If you are developing an Ajax application, this is the best way to find out where your bottlenecks are.

Conclusion

It's rare for me to call a product revolutionary, but I am sorely tempted to do so in the case of Firebug. It puts a lot of information in one nicely designed, easy-to-use package. The ability to interact with my packages has a vaguely Lisp-like feel, in that I'm suddenly interacting with a live environment. Firebug has given me the ability to "see" what my pages are doing and to better understand them.

Firebug has become an indispensable tool in my arsenal, alongside the Web Developer extension. In this way, Firefox is beginning to demonstrate that it is indeed a platform for Internet applications, rather than merely a Web browser.■

Reuven M. Lerner, a longtime Web/database consultant, is a PhD candidate in Learning Sciences at Northwestern University in Evanston, Illinois. He currently lives with his wife and three children in Skokie, Illinois. You can read his Weblog at altneuland.lerner.co.il.

Resources

The Firebug home page is **www.getfirebug.com**. There is limited documentation, including a FAQ, on that site.

An introduction to Firebug by its author, Joe Hewitt, in *Dr. Dobb's Journal*: **www.ddj.com/196802787**.

RouterBOARD™ 100 Series

made by routerboard.com

RouterBOARD 133

Low cost AP/CPE $89

RouterOS L4 Firewall/Routing

MIPS32 4Kc 175MHz embedded

32MB SDRAM

64MB onboard NAND storage

Three 10/100 ethernet ports MDI/X

Three miniPCI Type IIIA/IIIB slots

Power: 9-28V DC; Overvoltage protection

PoE: 12-28V DC (no power over datalines)

117mm x 105mm (4.61 in x 4.13 in), 115g

RouterBOARD 133C

Low cost CPE $59

RouterOS L3 Firewall/Routing

MIPS32 4Kc 175MHz embedded

16MB SDRAM

64MB onboard NAND storage

One 10/100 ethernet ports MDI/X

One miniPCI Type IIIA/IIIB slots

Power: 9-28V DC; Overvoltage protection

PoE: 12-28V DC (no power over datalines)

117mm x 105mm (4.61 in x 4.13 in), 79g

MARCEL GAGNÉ

When Ajax Held the World on His Shoulders

If the ancient Greeks had created open-source Web applications, would they have used Ajax...or maybe Atlas?

Sorry I'm late, François. I had to pick up tonight's wine from Henri. He suggested it would be perfect for tonight's menu, and I didn't want to pass it up. It's a 2003 Domaine Mathieu, Châteauneuf-du-Pape, and after my little quality assurance taste at Henri's Fine Wines, I am convinced our guests will throughly enjoy this wine.

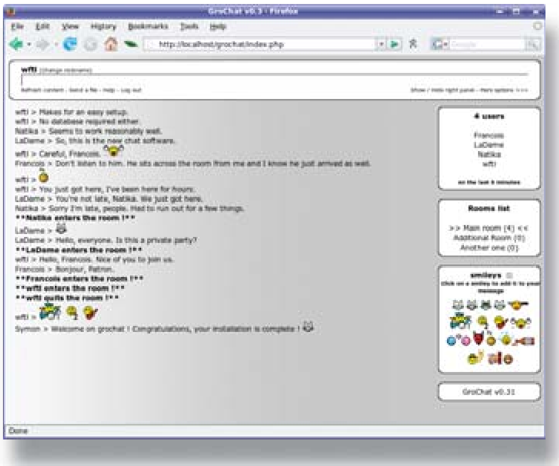
Mon Dieu, François! You have redecorated while I was away? Well, yes, the restaurant certainly looks impressive, but what is the idea behind the ancient Greek theme? And, the tablecloths—Atlas holding the world on his shoulders? I don't understand. *Quoi?* This issue's theme? *Non, non, non, mon ami*, it's not *Atlas*, it's *Ajax*. No, Ajax is not a Greek god; he was a human warrior in Greek mythology and a hero of the Trojan War. In our case, however, Ajax refers to Asynchronous JavaScript and XML.

Welcome, *mes amis*, to *Chez Marcel*, which (due to my faithful waiter's impromptu redecorating) is looking very Greek tonight. Nevertheless, we continue to serve exceptional wine with equally superb Linux and open-source software. Make yourselves comfortable, and I will have François serve the wine *immédiatement*. Because I've already picked up the wine tonight, and he is no doubt tired from his remodeling, we shall save him a trip to the cellar. Please, *mon ami*, pour for our guests. Of course, now that I think about it, tonight's wine doesn't quite fit the decor—regardless, this is a superb wine, *mes amis*. Enjoy!

Before you arrived, I was explaining to François that Ajax, besides being a character in Greek mythology, is a Web development technique designed primarily to create rich, interactive and fast Web applications that don't require a Web page to reload each and every time. A perfect example of when this technique pays off is with browser-based chat applications. In the past, I've covered Web-based chat programs, but their tendency to reload and regenerate conversations feels old compared with modern Ajax implementations.

Tonight's menu features two Ajax-based chat programs. The first is Simon Oualid's GroChat (Figure 1), an easy-to-use chat application that you can integrate into your own Web site. GroChat's features include public as well as private chats, a private file exchange system, multiple chat rooms, client-side chat history (so you can catch part of an earlier conversation) and more. It's frightfully easy to set up and requires no database. You do, however, need an Apache Web server with PHP to use GroChat, but that's pretty much it.

GroChat comes in a zip file. To install it, extract it somewhere in your Web server's path. The files will arrive in a directory with the release number (for example, grochat-0.33), so you may want to rename it to something less verbose, such as grochat, before you continue. Inside the release directory, there's a subdirectory named chatfiles. The text files inside need to be writable by your Web server—for instance, if your Apache user is apache


GroChat v0.3 - Firefox
File Edit View History Bookmarks Tools Help
http://localhost/grochat/index.php
wfti (change nickname)
Refresh content - Send a file - Help - Log out
Show / hide right panel - more options >>>
4 users
Francois
LaDame
Natika
wfti
on the last 4 minutes
Rooms list
>> Main room (4) <<
Additional Room (0)
Another one (0)
smileys
click on a smiley to add it to your message
GroChat v0.31
Done

```
wfti > Makes for an easy setup.
wfti > No database required either.
Natika > Seems to work reasonably well.
LaDame > So, this is the new chat software.
wfti > Careful, Francois.
Francois > Don't listen to him. He sits across the room from me and I know he just arrived as well.
wfti >
wfti > You just got here, I've been here for hours.
LaDame > You're not late, Natika. We just got here.
Natika > Sorry I'm late, people. Had to run out for a few things.
**Natika enters the room !**
LaDame >
LaDame > Hello, everyone. Is this a private party?
**LaDame enters the room !**
wfti > Hello, Francois. Nice of you to join us.
Francois > Bonjour, Patron.
**Francois enters the room !**
**wfti enters the room !**
**wfti quits the room !**
wfti >
Symon > Welcome on grochat ! Congratulations, your installation is complete !
```

Figure 1. GroChat is about as simple as it gets, and it still offers a rich chat experience.

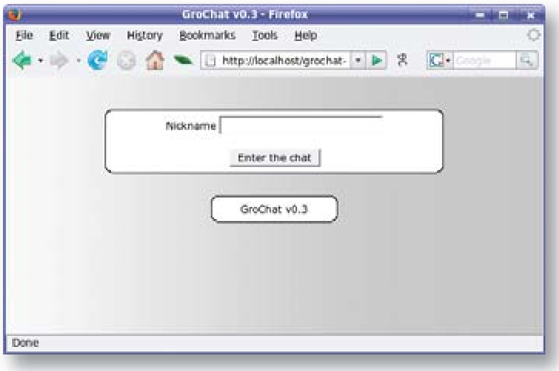

GroChat v0.3 - Firefox
File Edit View History Bookmarks Tools Help
http://localhost/grochat-
Google
Nickname
Enter the chat
GroChat v0.3
Done

Figure 2. After a simple installation, you are ready to use GroChat.

servers ▶ DIRECT™
MORE PRODUCTS, BETTER SERVICE, GUARANTEED.

GO STRAIGHT TO THE SOURCE!

SERVERS THAT WORK SMART

SAVE TIME AND REDUCE OPERATIONS COSTS WITH THE DUAL-CORE INTEL® XEON® PROCESSOR IN YOUR SERVERSDIRECT SYSTEM

1U Twin™ Innovation

* High Density Computing Technology
* Reducing Cost, Engery and Space Requirements
* Support up to **16 processor cores** Quad Xeon 5300 Series

SDR-6015T-TB 1U Data Center Clustering Server

Two systems (nodes) in a 1U form factor. Each node supports the following:

* Dual-processor Quad & Dual Core Intel® 64-bit Xeon® Support
* Up to 32GB DDR2 667 & 533 SDRAM Fully Buffered DIMM (FB-DIMM)
* 2x Intel® (ESB2/Gilgal) 82563EB Dual port Gigabit Ethernet Controller
* 2x Hot-swap SATA Drive Bays
* 900/980W High-efficiency Power Supply

SDR-2501T

2U Application Server

Ideal solution for a stable business-critical application server that minimizes deployment and support costs

* 2U Chenbro Rackmount Chassis with 600W power supply
* Supermicro X7DVL-E Server Board with Intel® 5000V (Blackford VS) Chipset
* Intel Dual-Core Xeon 5030 Processor 2.6GHz 667MHz
* Total 1024MB, 2pcs x 512MB Kingston DDR2 533MHz FB-DIMM ECC
* Seagate SATA II 80GB 7200 RPM 8MB Cache SATA 3.0Gb/s Hard Drive
* 6 x 1" Hot-swap SATA Drive Bays
* Intel® (ESB2/Gilgal) 82563EB Dual-port Gigabit Ethernet Controller
* Intel ESB2 SATA 3.0 Gbps Controller RAID 0, 1, 5, 10 support

STARTING PRICE **$1,299**

SDR-3111T

3u Mission Critical Application server

Designed to deliver exceptional availability, simplified manageability, outstanding performance and revolutionary scalability to help you build a cost-effective, flexible IT infrastructure

* 3U Chenbro Rackmount Chassis with 600W power supply
* Intel S5000PSLSATA Xeon 5000P Server Board
* Intel Dual-Core Xeon 5030 Processor 2.6GHz 657MHz
* Total 1024MB, 2pcs x 512MB Kingston DDR2 533MHz FB-DIMMECC
* Seagate SATA II 80GB 7200 RPM 8MB Cache SATA 3.0GB/s Hard Drive
* 12 x 1" Hot-swap SATA Drive Bays
* Dual-port Gigabit Ethernet Controller
* Intel SATA 3.0 Gbps 6-PORT Controller RAID 0, 1, 10 support

STARTING PRICE **$1,699**

SDP-7045B-TR+B

Ideal choice for small businesses looking for their first server or to upgrade an existing server

* Supermicro 4U Rackmountable / Tower Chassis with 800W High Efficiency Redundant Power Supply
* Supermicro Super X7DBE+ Server Board
* Intel Dual-Core Xeon 5030 Processor 2.6GHz 667MHz
* Total 1024MB, 2pcs x 512MB Kingston DDR2 533MHz FB-DIMM ECC
* Seagate SATA II 400GB 7200 RPM 16MB Cache SATA 3.0Gb/s Hard Drive
* 8 x 1" Hot-swap Drive Bays
* Intel® (ESB2/Gilgal) 82563EB Dual-port Gigabit Ethernet Controller
* Intel ESB2 SATA 3.0Gbps Controller RAID 0, 1, 5, 10 support

STARTING PRICE **$1,999**

SDR-5500T

5U advanced Storage Server

True SATA II 3Gb/s storage server configured with dual Xeon CPUs and room for a tape backup

* AIC 5U Rackmountable with 950W Triple Redundant Power Supply
* Supermicro Super X7DBE Server Board with Intel® 5000P (Blackford) Chipset
* Intel Dual-Core Xeon 5050 Processor 3.0GHz 667MHz
* Total 1024MB, 2pcs x 512MB Kingston DDR2 533MHz FB-DIMM ECC
* Seagate SATA II 400GB 7200 RPM 16MB Cache SATA 3.0Gb/s Hard Drive
* 24 x 1" Hot-swap Drive Bays
* Intel® (ESB2/Gilgal) 82563EB Dual-port Gigabit Ethernet Controller
* Intel ESB2 SATA 3.0Gbps Controller RAID 0, 1, 5, 10 support

STARTING PRICE **$5,399**

intel® Xeon® inside™ Dual-core. Do more.

SERVERS DIRECT CAN HELP YOU CONFIGURE YOUR NEXT HIGH PERFORMANCE SERVER SYSTEM - CALL US TODAY!

Our flexible on-line products configurator allows you to source a custom solution, or call and our product experts are standing by to help you assemble systems that require a little extra. Servers Direct - your direct source for scalable, cost effective server solutions.

1.877.727.7887 | www.ServersDirect.com

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, Pentium, and Pentium III Xeon are trademarks of Intel Corporation or it's subsidiaries in the United States and other countries.

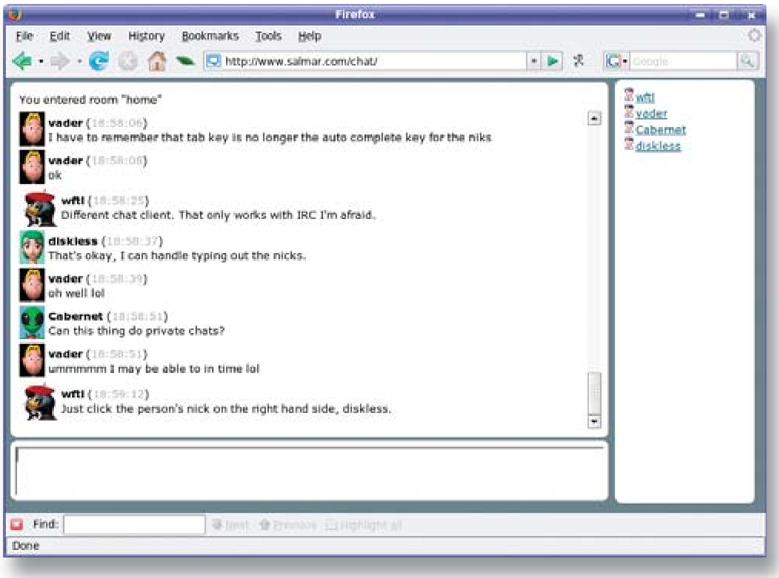
with group apache, type:

```
chown apache:apache chatfiles/*.txt
```

You also might verify that the Apache user and group both have write permission to these files. Other than that, your installation is complete. To use GroChat, simply connect using http://*yourserver*/grochat/. You'll find yourself at a simple login screen (Figure 2).

Choose a nickname (or user name), click Enter the chat, and you are good to go. The text input field is near the top of the screen. Directly below that are links to refresh content, send a file to another user, get help or log out. On the top right, you can choose to hide the information panel if you want. Also interesting is the link labeled More Options. This lets you change the color of your GroChat session or the session language (French, English and Spanish). It's all pretty simple, but you may want to do a little customization for your site. The config.php file in the the grochat directory allows you some control over your site. The most interesting items are near the top of the file:

```
$this->CHAT_FILE = "chatfiles/chat.txt";
$this->CHAT_HISTORY = "chatfiles/chatHistory.txt";
$this->CHAT_SIZE = "50";
$this->SESSION_DURATION = "5"; // in minutes
$this->DATETIME_FORMAT = "G:i:s"; // php format
$this->DEFAULT_LANGUAGE = "english"; // english, french or spanish
$this->DEFAULT_SKIN = "gray"; // gray, pink or blue
$this->MAX_UPLOAD_SIZE = 2000000; // in bytes
$this->MAX_UPLOAD_POOL = 10000000; // in bytes
$this->REFRESH_TIME = 500; // in milliseconds
```


Firefox
File Edit View History Bookmarks Tools Help
http://www.salmar.com/chat/
Google
Find:
Next Previous Highlight all
Done

You entered room "home"

vader (18:58:06)
I have to remember that tab key is no longer the auto complete key for the niks

vader (18:58:08)
ok

wftl (18:58:25)
Different chat client. That only works with IRC I'm afraid.

diskless (18:58:37)
That's okay, I can handle typing out the nicks.

vader (18:58:39)
oh well lol

Cabernet (18:58:51)
Can this thing do private chats?

vader (18:58:51)
ummmmm I may be able to in time lol

wftl (18:59:12)
Just click the person's nick on the right hand side, diskless.

wftl
vader
Cabernet
diskless

Figure 3. Chatting in an AjaxChat session is smooth and doesn't feel much like a Web application at all.

Most of these are self-explanatory, but the defaults may not be suitable to everyone. For instance, you may want to limit the file transfer size to something other than 2MB. When you log in to a default GroChat session, you'll notice you are in the "Main Room". On the right, in a box labeled Room list, you'll see two additional rooms creatively labeled "Another Room" and "Another one". You can change those names, add more rooms or delete them. This also is done via the config.php file. A little further down from the session defaults, is the following text:

```
// Comment this variable initialisation
// to disable multiroom support
$this->EXTRA_ROOMS = array(
      array("Additional Room","chatfiles/additional.txt",
"chatfiles/additionalHistory.txt"),
      array("Another one","chatfiles/additional2.txt",
"chatfiles/additionalHistory2.txt"),
```

Simply edit these to suit your own site and needs. If you feel particularly creative, you might want to add to the default collection of emoticons included with the package. Yes, these are also defined in the configuration file.

François, *mon ami*, this would be a good time to refill our guests' glasses. I do wish you hadn't chosen to wear that himation, though. It's a bit unsettling for some of our guests.

A somewhat more complex chat program, with more advanced features, is Lukasz Tlalka's AjaxChat. The only catch with this program is that you need a MySQL server to handle user registration, personal information, statistics and room creation, among other things. This is a nice, responsive application that looks good and feels good to use. I've used it with both Firefox and Konqueror, and both handle the features well. Take a look at Figure 3 to see AjaxChat in action.

Before I go into the details, I should point out that there are two chat applications on the **www.ajaxchat.org** site. Chat is the program I show you here, and Shoutbox is a simpler application, missing some of the richer features of its Chat sibling. Explore Shoutbox on your own. For now, let's concentrate on AjaxChat Chat.

To install AjaxChat, extract the tarred bundle into your Web site's directory structure. The resulting directory (with associated files and subdirectories) is called chat. If you connect to your Web site using http://www.*mywebsite*.dom/chat, AjaxChat takes you to the first screen of a configuration dialog. There are several screens here, which ask for the database type (such as MySQL) and name, the database administrator and password, and so on. As you fill in each screen, click Proceed to continue. I won't spend a lot of time explaining MySQL, but here are the basic steps I took. First, I created a MySQL database called chat:

```
mysqladmin -u root -p create chat
```

Polywell's Ultimate Linux Systems

More Choices, Better Service, Great Value

Tower Systems

4-way Opteron Workstation PolyStation 2055A-2210

-2xAMD Opteron™ Dual-Core 2210 Processors
-4GB Dual Channel 667MHz DDR2 ECC
-NVIDIA Quadro™FX 1300 OpenGL Graphics
-250GB SATA Drives, On-board RAID-5
-Sony 16X DVD-RW, Logitech K/B, Mouse
-2x Gigabit LAN, 2 x PCIe 16x Slots
-Linux / BSD / Windows OS Supported
$1999 (custom config. available)

SAS Disk Workstation PolyStation 2055SS-2216

-2xAMD Opteron™ Dual-Core 2216 Processors
-8GB Dual Channel 667MHz DDR2 ECC
-NVIDIA Quadro™FX 3500 OpenGL Graphics
-2x15K RPM 73GB SAS Drives, SAS Controller
-Sony 16X DVD-RW, Logitech K/B, Mouse
-2x Gigabit LAN, 2 x PCIe 16x Slots
-Linux / BSD / Windows OS Supported
$4500 (custom config. available)

32GB RAM Server with 3TB Tower Server 2500M-2220

-2xAMD Opteron™ Dual-Core 2220 Processors
-32GB Dual Channel 667MHz DDR2 ECC
-11-Bay Tower with 2x700W Power Supply
-3TB 6x500GB SATA Drives, RAID-5
-Sony 16X DVD-RW, Logitech K/B, Mouse
-2x Gigabit LAN, 2 x PCIe 16x Slots
-Linux / BSD / Windows OS Supported
$8999 (custom config. available)

Desktop Systems

Low-Cost Linux Appliance PC Poly 485Ax-2800SP

-AMD Sempron™ 2800+ 64-bit Processor
-256MB 667MHz DDR2 Memory
-ATI X300 PCIe Graphics
-80GB SATA Drive, SATA-RAID Controller
-16X DVD-ROM Drive, Logitech K/B, Mouse
-100Mbit LAN, 2 PCI + 1 PCIe 16x Slots
-Linux / BSD / Windows OS Supported
$299 (for volume purchase only)

2TB Storage Appliance NetDisk-485Ax - NAS

-AMD 64-bit Processor
-512MB 667MHz DDR2 Memory
-4 x SATA RAID Controller
-2TB 4x500GB SATA Drives
-2 PCI, PCIe Slots, Optional IDE Flash
-100Mbit LAN, Optional Gigabit LAN
-FREEBSD or Linux based NAS OS
$999 (for Volume OEM only)

High-End PC, 4G DDR2 Poly 430AM2-6000

-AMD Athlon™64 Dual-Core 6000 Processor
-4GB Dual Channel 667MHz DDR2
-NVIDIA GeForce™ 8600GT Graphics
-10K RPM 74GB + 320GB SATA Drives
-16X DVD-RW, 20-in-1 Reader, 1394a
-Gigabit LAN, 2 PCI + 1 PCIe 16x Slots
-Linux / BSD / Windows OS Supported
$1699 (custom config. available)

ISP Rack Servers

Low-Cost ISP 1U-Server 485Ax-1U 3500

-AMD Athlon™64 3500+ 64-bit Processor
-1GB Dual-Channel 667MHz DDR2 Memory
-4xSATA-RAID Controller
-2 x 80GB SATA Drives RAID-0/1
-1U 17" Short Rack, 1 Slim-CD, 2 x 3.5" Bays
-100Mbit LAN, 1 Optional PCIe or PCI RISER
-Linux / BSD / Windows OS Supported
$499 (for ISP volume purchase only)

4GB DDR2 ISP 1U-Server 430AM2-1U 4200

-AMD Athlon™64 X2 Dual-Core 4200+ Processor
-4GB Dual-Channel 667MHz DDR2 Memory
-4SATA RAID-5 Controller
-80GB SATA Drive, Optional 2nd Mirror Drive
-1U 17" Short Rack, 1 Slim-CD, 2 x 3.5" Bays
-Gigabit LAN, 1 Optional PCIe or PCI RISER
-Linux / BSD / Windows OS Supported
$799 (for ISP volume purchase only)

8GB ECC, Opteron Server 1000SL-1U Opteron 1210

-AMD Opteron™Dual-Core 1210 Processor
-8GB Dual Channel 667MHz DDR2 ECC
-2 x GigaLAN, 4SATA RAID-5 Controller
-2x80GB SATA Drives, Optional 2x500G
-1U 17" Short Rack, Slim-CD+2x3.5" Bay
-Optional PCIe or PCI RISER Slot
-Linux / BSD / Windows OS Supported
$1999 (custom config. available)

High-Density Multi-Processor Servers

16-way Opteron, 128GB RAM PolyServer 8850T5U-8220

-8 x AMD Opteron™ Dual-Core 8220 Processors
-128GB Dual Channel 667MHz DDR2 ECC
-12TB 24x500GB SATA Drives, RAID-5
-4 x Gigabit Ethernet, 4 PCI-X, 2 PCIe Slots
-5U 24-Bay RackCase, Redundant Power Supply
-Linux / BSD / Windows OS Supported

16-way 8220, 128G RAM, 24x500G $69999
16-way 8212, 32GB RAM, 8x500GB $17999

18TB 4U Storage, 64GB RAM 4024AIS 2500M-2220

-2 x AMD Opteron™ Dual-Core 2220 Processors
-64GB Dual Channel 667MHz DDR2 ECC
-18TB 24x750GB SATA Drives, RAID-5
-2 x Gigabit Ethernet, 4 PCI-X, 2 PCIe Slots
-4U 24-Bay RackCase, Redundant Power Supply
-Linux / BSD / Windows OS Supported

18TB 24x750G, 64G, 2x2220 $36999
12TB 24x500G, 4GB, 2x2210 $9999

9TB 2U Storage, 32GB RAM 2012SC 2500M-2220

-2 x AMD Opteron™ Dual-Core 2220 Processors
-32GB Dual Channel 667MHz DDR2 ECC
-9TB 12x750GB SATA Drives, RAID-5 Controller
-2 x Gigabit Ethernet, 4 PCI-X, 2 PCIe Slots
-2U 12-Bay RackCase, 3 RISER or 7 Lowprofile
-Linux / BSD / Windows OS Supported

9TB 12x750G, 32G, 2x2220 $13999
6TB 12x500G, 4GB, 2x2210 $5999

3TB 1U Server, 32GB RAM 2500A16-1U4 2212

-2xAMD Opteron™Dual-Core 2212 Processors
-32GB Dual Channel 667MHz DDR2 ECC
-3TB 4x750GB SATA Drives, RAID-5
-2 x Gigabit Ethernet, 1 PCI-X or PCIe RISER
-1U RackCase, 1 Slim CD, 4 Swap HD Bays
-Linux / BSD / Windows OS Supported

3TB 4x750G, 32G, 2x2212 $8999
2TB 4x500G, 4GB, 2x2210 $2999

AMD Dual-Core technology enables one platform to meet the needs of multi-tasking and multi-threaded environments; provides platform longevity

AMD 64 Opteron™

AMD 64 Athlon X2

- 20 Years of Customer Satisfaction
- 5-Year Warranty, Industry's Longest
- First Class Customer Service

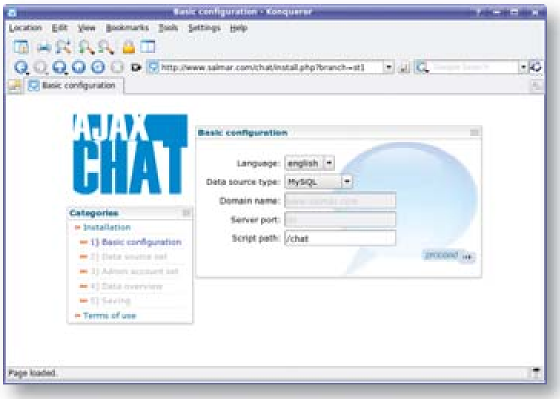
Polywell OEM Services, Your Virtual Manufacturer
Prototype Development with Linux/FreeBSD Support
Small Scale to Mass Production Manufacturing
Fulfillment, Shipping and RMA Repairs

888.765.9686
www.Polywell.com/us/LJ

POLY
POLYWELL

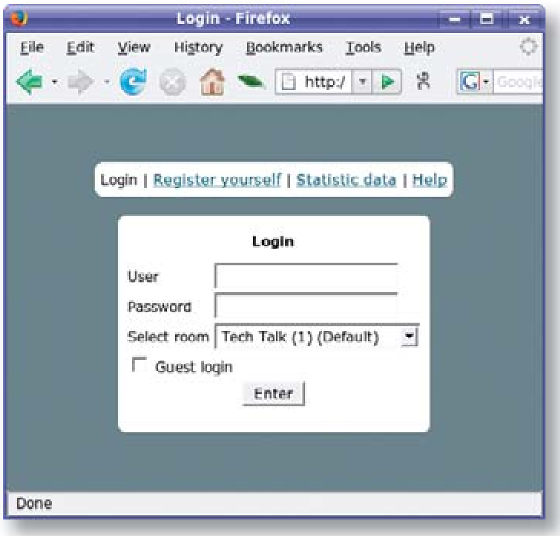
Polywell Computers, Inc 1461 San Mateo Ave. South San Francisco, CA 94080 650.583.7222 Fax: 650.583.1974

Opteron, Sempron and ATHLON are trademarks of Advanced Micro Devices, Inc.. Quadro, nForce and Nvidia are trademarks of NVIDIA Corporation. All other brands, names are trademarks of their respective companies.



Language: english
Data source type: MySQL
Domain name:
Server port:
Script path: /chat

Figure 4. Once you have created your database, the remaining configuration steps are done via the AjaxChat Web interface.


Login - Firefox
File Edit View History Bookmarks Tools Help
http:/
Google
Login | Register yourself | Statistic data | Help
Login
Enter
Done

User
Password
Select room Tech Talk (1) (Default)
□ Guest login

Figure 5. The AjaxChat Login and Registration Screen

The mysqladmin program then asked for my admin password before creating the database for me. Next, I needed an admin name to serve as my login to the chat and to administer the AjaxChat system. To do that, I went into an interactive MySQL session by typing `mysql -u root -p`. After entering the password, I found myself at a `mysql>` prompt. From that command prompt, I created an admin called chatadmin with a super-secret password:

```
mysql -u root -p
mysql> GRANT ALL PRIVILEGES ON fishy.* TO 'fishyadm'@'localhost'
 ➥IDENTIFIED BY 'supersecretpassword';
mysql> FLUSH PRIVILEGES;
mysql> \q
```

That's it. The rest was simply a matter of entering the appropriate details into the setup screens (Figure 4).

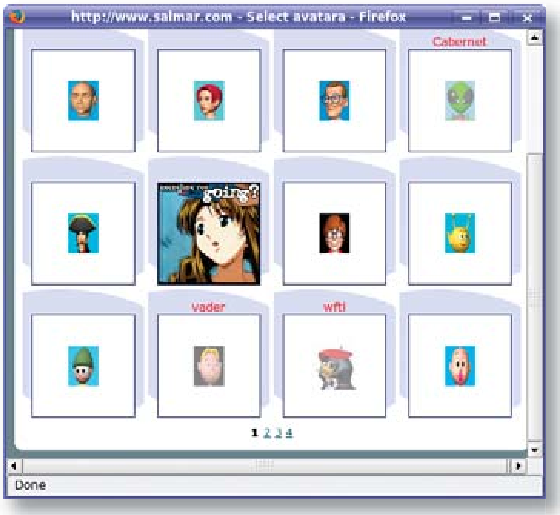
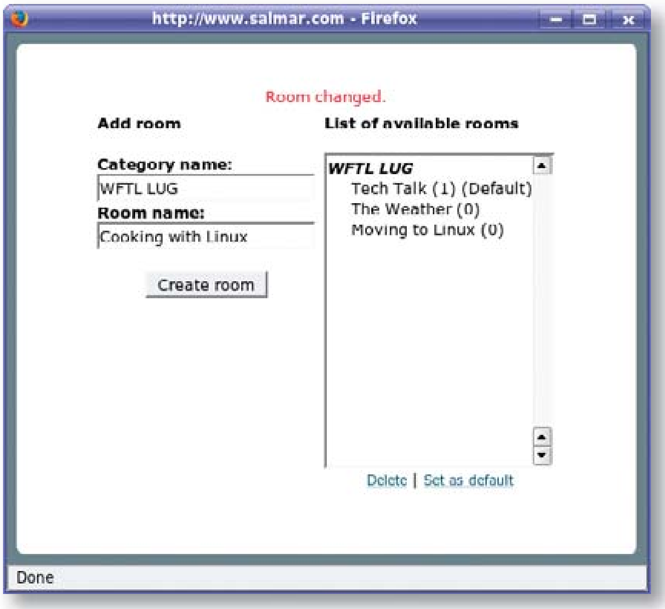


Figure 6. Several avatars are included by default, but you can add more as demonstrated by the appearance of my waiter's likeness.

When the setup is complete, visiting http://www.*yourwebsite*.dom/chat will take you to a login screen (Figure 5). From there, you can either log in or register before logging in. Two additional options are worth mentioning at this point. Statistical data will tell you what rooms are available, how many users are registered, the amount of traffic on the site and a few other items, without going into private details or conversations. There's also a help link that will show you some of the commands available in a chat session.

Users don't have to register to join the chat, but some nice features are associated with doing so. For one, your nickname is protected, and no one else can impersonate you. Furthermore, the system keeps statistics on how many messages you've sent, your date of registration, last time on and so on. Registered users also can select an avatar to represent them. To do that, type `/avatar` in the text entry field. A pop-up window appears with a list of several avatars (Figure 6). These are included with the AjaxChat distribution, and are located in the img/avatars subdirectory of your chat directory. Adding your own avatars is as easy as adding them to the avatars subdirectory. Don't overdo it though. Most of the included images are 30x40 pixels.

AjaxChat supports additional features, which you can discover by typing `/help` in the text input area. To start a private chat with another user, simply click his or her nickname in the list on the right-hand side. A pop-up window appears in which you can chat without others on the list noticing. You also can have multiple rooms and jump from chat room to chat room. To configure rooms, type `/configrooms`. To add rooms, select (or create) a category name, and then add rooms to those categories (Figure 7).



Room changed.

Add room

Category name: WFTL LUG

Room name: Cooking with Linux

Create room

List of available rooms

WFTL LUG
- Tech Talk (1) (Default)
- The Weather (0)
- Moving to Linux (0)

Delete | Set as default

Done

Figure 7. Creating a New Room

Click the Create room button, and that's it. You can add as many rooms as you like or delete existing rooms via this dialog. You also can define any of the rooms you create as the default room—that's the room users find themselves in when they log in to the AjaxChat session.

All this chatting has made the time fly, *mes amis*. Still, there is plenty of wine left. While François tops up your glasses, feel free to continue your on-line and off-line chats. Before we meet here again, I'll make sure François returns things to normal *and* that he wears something more appropriate than a himation. Until next time, please raise your glasses, *mes amis*, and let us all drink to one another's health. *A votre santé! Bon appétit!*■

Marcel Gagné is an award-winning writer living in Waterloo, Ontario. He is the author of the all-new *Moving to Free Software*, his sixth book from Addison-Wesley. He also makes regular television appearances as Call for Help's Linux guy. Marcel is also a pilot, a past Top-40 disc jockey, writes science fiction and fantasy, and folds a mean Origami T-Rex. He can be reached via e-mail at mggagne@salmar.com. You can discover lots of other things (including great Wine links) from his Web site at www.marcelgagne.com.

Resources

AjaxChat: **www.ajaxchat.org**

GroChat: **grochat.sourceforge.net**

Marcel's Web Site: **www.marcelgagne.com**

The WFTL-LUG, Marcel's On-Line Linux User Group: **www.marcelgagne.com/wftllugform.html**

DAVE TAYLOR

Displaying Image Directories in Apache, Part II

Get the Apache images in thumbnails by putting everything in a for loop.

Last month, we started writing a shell script to turn the boring Apache directory display into a more useful and visually interesting page that helps you figure out what images you have and what they look like.

By utilizing the file command, I showed how you easily can differentiate between files that actually are images and those that aren't, so you don't get into an awkward situation where you're trying something like this:

```
<img src=mypage.html alt=mypage.html />
```

If you tell the Web browser to try to display an HTML source file as an image, well, the results aren't going to be what you desire!

Everything in a for Loop

The basic script iterates through every file in the current directory with a for loop, using the common shell construct:

```
for name in *
do
   commands
done
```

Let me point out that this won't display files that start with a ., which is good in that it doesn't display . and .., but which is potentially a problem if you were being tricky and had filenames like .secret-pict.png. But, then again, if you're trying to hide files by making them dot files, it's not unreasonable that this script glosses over them too.

As I've shown earlier, if you do have an image file, the best way to display it in HTML is to use something akin to:

```
<img src="filename" alt="filename" />
```

A better solution is maybe to label the image in the alt tag, but let's just jump into the loop and add this code. Recall that I'm using the file command to figure out what's an image and what isn't, so now our core loop looks like this:

```
for name in *
do
  if [ ! -z "$(file $name | grep 'image data')" ]
  then
    echo "<img src=$name><br />$name<br /><br />"
  else
    echo "<a href=$name>$name</a><br /><br />"
  fi
done
```

This works pretty well, displaying the images (and their names) for those files that are recognized as images, and just displaying a hypertext reference to the other files in the directory without erroneously indicating they're images.

Thumbnails Please

The problem with this approach is demonstrated quickly if, like me, you have lots of variable-size images; the resultant page is huge! What I really want displayed are small thumbnails or previews of my images, not the big images themselves.

Fortunately, Web browsers are pretty darn good at scaling images if you ask them to do the work. For example, if you have a 300x300 image but specify a height and width of 50, the image is scaled and displayed as 50x50 in the browser automagically. What you might not realize is that browsers also can scale an image if you simply specify either a different height or width value. In other words, this works fine:

```
<img src=100x100.png height=50 />
```

That's good news, because although you can figure out the size of an image on the fly in your shell script, it's fairly complicated. So, if we simply can specify that one parameter always should be a given height, you quickly can get quasi-thumbnails, albeit sometimes oddly sized ones.

The problem, by the way, is if we say that we always want images to be 50-pixels high and scale appropriately, an image that's 480 wide by 50 wide becomes, well, a 480-pixel-wide thumbnail. Ideally, our thumbnails would fix into a 50x50 box instead, but let's start with a basic solution:

```
for name in *
do
  if [ ! -z "$(file $name | grep 'image data')" ]
  then
    echo "<img src=$name alt=$name height=50 />"
    echo "<br />$name<br /><br />"
  else
    echo "<a href=$name>$name</a><br /><br />"
  fi
done
```

I also added an alt attribute to the img, though it doesn't really make any difference in the display. As you can see in Figure 1, the display overall is pretty nice. But, I have one image, logo-small.png, that turns out to be 850x40, so forcing a height of 50 pixels actually increases its width, by scaling up, not down!

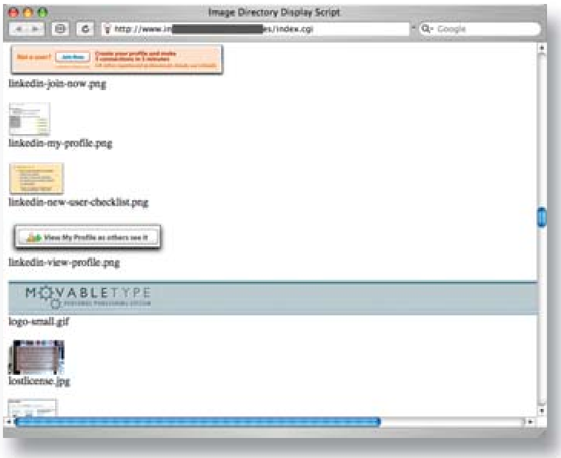


Image Directory Display Script
http://www.in[redacted]es/index.cgi
Google

linkedin-join-now.png

linkedin-my-profile.png

linkedin-new-user-checklist.png

linkedin-view-profile.png

logo-small.gif

lostlicense.jpg

Figure 1. Example Thumbnail Display

Figuring Out Image Size

It would be quite useful to be able to ascertain the size of an image and scale it appropriately. In some versions of Linux, you can get the image size information from the file command itself:

```
$ file xml.gif walt-disney-world-logo.jpg zeralights-logo.png
xml.gif:  GIF image data, version 89a, 36 x 14,
walt-disney-world-logo.jpg: JPEG image data,
JFIF standard 1.01, resolution (DPI), "AppleMark", 72 x 72
zeralights-logo.png: PNG image data, 225 x 93,
8-bit/color RGB, non-interlaced
```

The problem here is that the file command doesn't know how to ascertain the size of JPEG files, so the 72x72 reported for the image walt-disney-world-logo.jpg is actually the resolution of the image, not its size—a terrible limitation, but one we can live with, albeit reluctantly. Anyway, you should be using PNG format, not JPEG, right?

Based on that output, here's a shell function that returns height and width for GIF and PNG images and a null value for JPEG and any non-image files:

```
figuresize()
{
   image=$1

   fileout="$(file -b "$1")"

   if [ ! -z "$(echo $fileout|grep "GIF image")" ]
   then
     # GIF image, width x height are params 6-8
     width=$(echo $fileout | cut -f6 -d\  )
     height=$(echo $fileout | cut -f8 -d\  )
   elif [ ! -z "$(echo $fileout|grep "PNG imag")" ]
   then
     # PNG image, width x height are params 4-6
     width=$(echo $fileout | cut -f4 -d\  )
     height=$(echo $fileout | cut -f6 -d\  )
   else
     height=""; width=""
   fi
}
```

This is now integrated easily into our original loop, so we also can display the size of the image in our output:

```
for name in *
do
  if [ ! -z "$(file -b $name|grep 'image data')" ]
  then
    figuresize $name
    if [ ! -z "$height" ] ; then
      echo "<img src=$name alt=$name height=50 />"
      echo "<br />$name ($height x $width)<br />"
    else
      echo "<img src=$name alt=$name height=50 />"
      echo "<br />$name<br />"
    fi
  else
    echo "<a href=$name>$name</a><br /><br />"
  fi
done
```

I've run out of space to show how you can use that information to change how you scale your thumbnails, so that'll have to cascade into next month, but I encourage you to experiment with this code a bit and see what kind of results you get. Also, as a tip, if you want to get the size of all image types reliably, there's no better toolkit to add to your Linux box than ImageMagick, which you can find at **www.imagemagick.org**.■

Dave Taylor is a 26-year veteran of UNIX, creator of The Elm Mail System, and most recently author of both the best-selling *Wicked Cool Shell Scripts* and *Teach Yourself Unix in 24 Hours*, among his 16 technical books. His main Web site is at www.intuitive.com, and he also offers up tech support at AskDaveTaylor.com.

JON “MADDOG” HALL

What Is the Value of Software?

It looks pretty, but what can it do?

As someone who has spent a fair amount of time on or around the sea, I always enjoy looking at boats and ships—and comparing them with canned corn.

Canned corn is a commodity. Although we may appreciate what companies go through to select the corn, clean it, can it and move it to store shelves, for the most part, one can of corn is exactly like another can of corn. No consumers spend too much time standing in front of the canned corn shelves at their store, wondering whether a particular brand would be better for their family than another one. They select a can and move on. As I said, canned corn is a commodity.

This is not the case with boats. Although most people would expect any given boat to float, different designs of boats meet different needs. There are sailboats, power boats, tugboats, and different models of each of these types. Each model of boat has a slightly different design based on the needs of the captain, crew and business.

Just as most people would not spend hours in front of a shelf deciding on which can of corn to buy, most people would not run down to the yacht broker, throw a suitcase of money on the dock and say that they would take any boat the broker had in stock.

And, although the total cost of ownership of the boat must be taken into consideration, the real determining factor in buying a boat is whether it meets your needs, the pay-back period on the purchase, and whether you could modify the boat to meet future desires.

In other words, the boat is not a commodity.

These days, you hear a lot about the operating system being a “commodity”, or that “no one cares about the operating system”. I believe that this broad generalization, like a lot of broad generalizations, fails under many circumstances. Let’s see where this particular generalization runs aground.

A bank in Brazil had automatic teller machines (ATMs). The operating system that it used in these ATMs was OS/2. The company that made OS/2 was “retiring” OS/2, and the bank would no longer be able to get the support it needed for new motherboards, device drivers and so forth. The ATMs used Intel 386, 486 and low-end Pentiums as CPUs, with varying amounts of main memory, disk sizes and other differences, reflecting a long time line of development and deployment.

The bank decided to use Linux as its new operating system, because it supported all the different CPUs, and the bank would be able to maintain it or expand on it into the far future. Here, the value of the operating system was in its potential longevity.

A lottery system, also in Brazil, decided to use Linux and free and open-source software inside the lottery terminals. It was replacing software from a proprietary vendor who had taken months to deploy requested changes to the lottery system. With a local development team, these deployment schedules dropped from months to weeks. This generated millions of dollars of additional revenue, as the new features could be delivered faster. The value here was the speed in making changes.

A public transportation authority was spending millions of dollars a year on proprietary software that was used only to write letters, do presentations and run simple spreadsheets. By moving to OpenOffice.org, the authority was able to save those millions of dollars and spend that money on hiring new cleaners for the transportation system. The manager of the transportation authority said that nobody cared which software was used to type the letters, but that everyone mentioned how the transportation vehicles were being kept very clean. The value was in re-deploying the money that would have been spent on royalties to services people desired.

A company in Rio de Janeiro needed a system that would work in Portuguese, but the software it could buy was only available in English. A free software developer duplicated the functionality of the English-only software using MySQL, Python, FreeGIS software, Gnuplot and other free and open-source software. The combined software was designed to prompt and report in Portuguese, and it cost less to develop than the packaged system. The value was having software in the company’s own language and creating a local programming job.

So, you see that the value of the software, as with a boat, cannot be measured by the cost of building it, but by how well it meets the needs of the user. And, although I have seen many, many families that can eat the same can of corn, I have never, even one time, seen exactly the same business problem.■

Jon “maddog” Hall is the Executive Director of Linux International (www.li.org), a nonprofit association of end users who wish to support and promote the Linux operating system. During his career in commercial computing, which started in 1969, Mr Hall has been a programmer, systems designer, systems administrator, product manager, technical marketing manager and educator. He has worked for such companies as Western Electric Corporation, Aetna Life and Casualty, Bell Laboratories, Digital Equipment Corporation, VA Linux Systems and SGI. He is now an independent consultant in Free and Open Source Software (FOSS) Business and Technical issues.

EmperorLinux

...where Linux & laptops converge

Portable

Since 1999, EmperorLinux has provided pre-installed Linux laptops to universities, corporations, government labs, and individual Linux enthusiasts. Our laptops range from full-featured ultra-portables to desktop replacements. All systems come with one year of Linux technical support by phone and e-mail, and full manufacturers' warranties apply.

Toucan T60/T60ws

ThinkPad T60/T60ws by Lenovo

- Up to 15.4" WSXGA+ w/ X@1680x1050
- ATI Mobility FireGL V5200
- 1833–2333 MHz Core 2 Duo
- 512 MB–4 GB RAM
- 60–120 GB hard drive
- CDRW/DVD or DVD±RW
- 5.2–6.0 pounds
- 10/100/1000 Mbps ethernet
- 802.11a/b/g (54Mbps) WiFi
- **Starts at $1950**

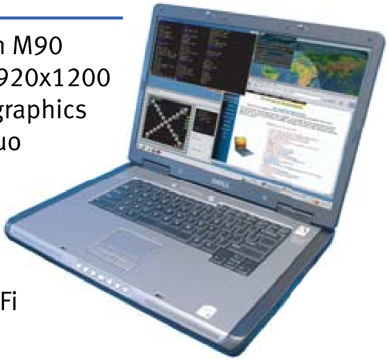
Powerful

EmperorLinux specializes in the installation of Linux on a wide range of the finest laptops made by IBM, Lenovo, Dell, Sony, and Panasonic. We customize your choice of Linux distribution to your laptop and provide support for: ethernet, wireless, X-server, ACPI power management, USB, EVDO, PCMCIA, FireWire, CD/DVD/CDRW, sound, and more.

Rhino D820/M90

Dell Latitude D820/Precision M90

- Up to 17" WUXGA w/ X@1920x1200
- NVidia Quadro FX 3500M graphics
- 1667–2333 MHz Core 2 Duo
- 512 MB–4 GB RAM
- 60–160 GB hard drive
- CDRW/DVD or DVD±RW
- 6.3–8.6 pounds
- 802.11a/b/g (54Mbps) WiFi
- ExpressCard/EVDO
- **Starts at $1480**



Unique

EmperorLinux offers Linux laptops with unique features. Ruggedized Panasonic laptops are designed for harsh environments: drops, vibrations, sand, rain, and other extremes. ThinkPad tablet PCs are like other laptops, with an LCD digitizer for pen-based input both as a mouse and with pressure sensitivity for writing and drawing on-screen.

Raven X60 Tablet

ThinkPad X60 Tablet by Lenovo

- 12.1" SXGA+ w/ X@1400x1050
- 1667–1833 MHz Core Duo
- 1–4 GB RAM
- 80–120 GB hard drive
- 3.8 pounds
- Pen/stylus input to screen
- Dynamic screen rotation
- Handwriting recognition
- X60s laptops available
- **Starts at $2300**

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

Model prices, specifications, and availability may vary. All trademarks are the property of their respective owners.

Penguins in Winnipeg

A conversation with Michael Collins about what's up with the Manitoba Media Centre.

DOC SEARLS

Sometimes I like to get back to UpFront stories that deserve more than the few inches they get the first time around. Such was the case with the news that the Manitoba Media Centre (**manitobamediacentre.org**), a new "Open Source Entertainment Engineering, Innovation, and Production Research Facility" in Winnipeg, was already funded for seven figures with plans to push that number up to eight [see Doc's story in *LJ*'s March 2007 UpFront section]. I got to thinking..."Why so much money? And, why Manitoba?"

So, I began corresponding with Michael Collins, the CEO of Linux Media Arts (LMA) and the prime mover behind the project. Veteran *Linux Journal* readers who followed Robin Rowe's coverage of Linux Hollywood Domination may recall Robin's encounter with Michael at the 2001 NAB (National Association of Broadcasters) show, where Michael said, "Our goal is to make Linux the premier multimedia editing and media production platform in the world, largely using open-source software" (**www.linuxjournal.com/article/4743**). The next year, Robin reviewed LMA's Broadcast 2000 nonlinear editor (NLE)—a product that has since been succeeded by Cinelerra (**cinelerra.org**), a Linux-based 64-bit open-source editing system, which, at last check, had around 700,000 search results on Google. Although this was a Linux Movie Mojo story on its own right, LMA's Manitoba move seemed too interesting to ignore. So, I decided to interview Michael, with interesting results. Read on.

LJ: What got the Manitoba Media Centre going?
MC: The concept for an Open Source Research and Development facility in Manitoba was first considered three years ago at the IBC show in Amsterdam. I made plans to meet up with a colleague from Winnipeg with whom I had been collaborating remotely in the design of a high-definition video board. His name is José A. Rueda. José is an experienced multimedia systems engineer with a PhD in Electrical and Computer Engineering and an MBA. José had a good understanding of the business side of media engineering and open source, and he was very willing to collaborate. José spoke glowingly about Manitoba's technological and financial resources and its highly stable professional population—something lacking in other regions. So, it looked to me like Manitoba might be a perfect R&D climate, and I kept that in the back of my mind while we moved forward with various other projects.

Then, business prospects rose dramatically, and with them the need for good R&D. So, last spring, I called up José to discuss establishing an R&D facility for LMA in Winnipeg. Then, I flew to Winnipeg in August for a four-day trip to meet with José and a few of his colleagues from business and government. José, who is very well respected in Manitoba business and political circles, took me around town to meet with numerous people from the public and private sectors. We first met with Edward Suzuki of Destination Winnipeg, the economic development agency for the city of Winnipeg, and he was very keen in helping us and assisting us plan my itinerary for the meetings in my first visit and my future visits. We also met with John Clarkson, who is the Deputy Minister for Science, Technology, Energy, and Mines in the Provincial Government of Manitoba.

I laid out a vision for a worldwide center for open-source media technologies. John Clarkson was very curious, supportive and perceptive, and he surprised me at the end of the meeting by mentioning that the Premier of Manitoba, Gary Doer, was very interested in alliances with technology companies based in California. He said the Premier was scheduled for a trip to Los Angeles in a few weeks' time and might be interested in discussing the concept further.

A few weeks later, I got an e-mail from John telling me the Premier wanted to visit our Burbank offices and wanted me to present our vision directly to him. So, Mr Doer came accompanied by John Clarkson, and I laid out my vision for a research center in Winnipeg that took advantage of the professional engineering population and the supportive business community in Manitoba.

A week later, we heard back that the Premier wished to go forward with the concept. So, I took two more trips to Winnipeg to determine the project's viability and to make plans for its eventuality. José and I continued to meet with those interested in the financing, support and professional requirements for building a facility from scratch. Eventually, we came up with a viable plan and decided to announce the Manitoba Media Centre (MMC), along with the initial $20 million investment, on December 15, 2006, at the Kodak Theatre in Hollywood.

LJ: Can you say who the investors are?
MC: I can say that Linux Media Arts is devoting $5 million in current revenues to the Centre to jump-start development endeavors. The rest is coming from a combination of investments from the province and from regional interests in Manitoba. The multimillion-dollar investment will enable us to engage in business opportunities and relationships that were not available to us before.

LJ: What projects are you starting out with?
MC: LMA has a number of complete systems architecture

projects in-house right now. We will be hiring employees and contributors of the MMC to support key technology development. Additionally, the MMC will support technical development where markets currently exist and are not being properly supported by the industry, in our opinion.

LJ: What are some of those markets?

MC: Our focus is on larger markets such as broadcasting and film production systems engineering projects, particularly in Asia. Wherever people want their open-source multimedia project to be developed to a professional standard, we can help. Our engineering staff is very experienced. We have PhDs with qualifications from the largest media engineering and media production companies.

LJ: What do you expect to see coming out of the Centre?

MC: We expect to see the MMC become a worldwide leader in open-source development technologies, where a full-time staff will support research and development, and technologists will come to the Centre from all around the world. In fact, we intend to participate at NAB in April 2007. We also intend to host a three-day conference in Winnipeg in fall 2007.

In addition, we have plans to build a multimedia production centre for the Misipawistik Cree Nation in Manitoba. Users will be able to record and produce programming for the Internet, cable and satellite distribution.

We are also assisting in the development of a curriculum based on open-source media applications for the Red River College in Winnipeg. We are collaborating with the University of Manitoba Experimental Media Centre as well.

LJ: What are some of the technologies you're talking about here?

MC: Archiving, indexing, restoration, telecine and media distribution.

LJ: Let's look at the big Linux development picture and put these in context. Right now, the LAMP stack has grown to more than 140,000 components, if we look at the list of projects on SourceForge alone. What's still missing there for multimedia?

MC: In our opinion, what's missing is a professional perspective with the finishing touch and polish necessary to get professional attention. Open-source media applications and systems have to be better to achieve a larger reach in a competitive and dynamic market.

LJ: In the December 2006 announcement, you said the MMC would work on "an Open Source Media Distribution Operating System". Is Linux not already that?
MC: Of course it is. However, what we intend to do is to release it strictly from the perspective of what is important to multimedia users. In other words, tool and applications and kernel changes that will improve the media experience.

LJ: Can you be more specific about kernel changes?
MC: Well, we stick pretty closely to what Red Hat and SUSE have been distributing for 64-bit architectures on the 2.6 kernel. The changes we make in the release and device drivers are more applicable to unique video cards, video networking devices, moving large files and adherence to SMPTE protocols. We also find it advantageous to remove some of the options for a typical production worker who is using a system as an appliance. Also, in streamlining the size and scope of the release, it tends to pick up speed, and this is also important for moving video frames.

LJ: What are some examples of archival and indexing tools or other open-source multimedia applications?
MC: MXF is a metadata-based archiving tool that is open source. We're also talking about many compression technologies, which have the ability to place indexable wrappers on them.

LJ: Will there be efforts to carry forward existing open-source multimedia apps that are already out there?
MC: Absolutely. There are numerous examples that will be included with the distribution. In particular, we expect to finally bring the promise of Cinelerra to fruition for professional markets on a wide scale.

LJ: How is Cinelerra doing?
MC: Cinelerra continues to be developed by its originator and main developer Adam Williams, and supported by an active open-source team from all over the world. Nothing proprietary will be implemented in Cinelerra on our watch. But, it will soon become clear that Cinelerra will compete favorably and beat the biggest names in the business. That is one of the main goals of the Manitoba Media Centre. We want to bring Cinelerra into its rightful place as a superior media software application that lives up to its original vision: to change how we edit and tell stories with video.

LJ: I see Cinelerra is GPL'd. Will the Manitoba Media Centre work to make other developments GPL'd as well?
MC: Absolutely. We have been very successful in developing and implementing GPL software on consulting contracts. We intend to devote a great deal of effort developing and promoting GPL software.

LJ: How will the development communities work? Who will keep the code repositories?
MC: Like other communities, these will attract contributors who have a stake in what the code will do. And, that's in addition to the professionals we're hiring to work on the code as well. Both the code and the repositories will be located on our servers in Winnipeg and mirrored elsewhere.

LJ: How is Linux progressing in Hollywood in general? Can we expect to see proprietary solutions increasingly replaced?
MC: Oh, yes. Linux continues to be high on the development agendas of numerous major studios. Current integrations are usually with in-house solutions. We expect efforts such as the MMC to change that by making more professional tools available on the open market for everybody.

LJ: How will it change your business in particular?
MC: Soon we'll be announcing a motion picture with a major Hollywood star to be shot entirely in Manitoba by a Canadian director and produced in collaboration with our team using our media production systems. All of our media solutions will be involved from pre-production to production to post-production to distribution. We can even provide the systems to project the film at an electronic cinema.

LJ: Let's look at developments on the demand side of the market, where more consumers are becoming producers, and the quality of digital goods is moving toward what you pros call "cine" quality. What's the biggest step for getting us there?
MC: Cheaper 1080p cameras, cheaper lenses and affordable easy-to-use tools.

LJ: How soon do you think 1080p shooting and production will be within reach of amateurs and low-budget independent moviemakers?
MC: It's already within the reach of people willing to take a professional approach to their production. In other words, it's very much in reach of the first adopters from a financial perspective. "The typical amateur" includes so many different types of people that we're bound to see lots of good work. To do that work, they'll find that software and systems are in place right now. The most expensive part of the production is the capture end, meaning the camera. After that, it's a storage issue. If you want to take it to film, then those are additional costs. It is expensive, but it's not impossible, as it was just a few years ago. If you want to have something professional for a film festival or even for distribution, you can do it.■

Doc Searls is Senior Editor of *Linux Journal*. He is also a Visiting Scholar at the University of California at Santa Barbara and a Fellow with the Berkman Center for Internet and Society at Harvard University.

Coraid's SR1521T EtherDrive Network Storage Tower

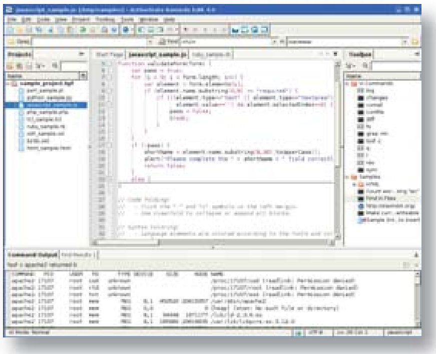
The Coraid folks have released a free-standing tower version of their EtherDrive Storage Appliance, christened the SR1521T. The SR1521T, with internal RAID, provides true networked storage over standard Ethernet and is based on the open ATA over Ethernet (AoE) protocol. Coraid says that "AoE is a block level storage protocol that is simpler to implement than other SAN technologies and at considerably less cost than iSCSI and Fibre Channel solutions." Further, because AoE is non-routed, "the need for TCP/IP processing overhead or expensive network adapters" is eliminated. The target customers are those that do not seek a rackmounted solution and "that need powerful and scalable network storage readily available at the departmental level". The AoE protocol is native in the Linux 2.6 kernel, and software drivers are available for Mac OS X, Windows, Solaris and FreeBSD.

www.coraid.com

ActiveState's Komodo Edit

ActiveState recently released its new version 4.0 of Komodo Edit, the firm's free, multilanguage editor for dynamic languages. Based on ActiveState's Komodo IDE 4.0, Komodo Edit not only supports Perl, PHP, Python, Ruby and Tcl, but also client-side languages and Ajax technologies, including JavaScript, CSS, HTML and XML. OS support includes Linux, Mac OS X and Windows. Among the product's advantages, says ActiveState, are capability additions via .xpi extensions, storage of configurable elements (such as run commands, macros, code snippets and so on), syntax checking and coloring, an active community site and more. A gratis download of Komodo Edit is available from ActiveState's Web site.

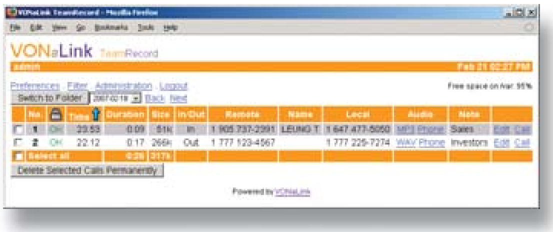
www.activestate.com



Arcosoft's VONaLink TeamRecord

Arcosoft joins the growing list of companies that have come to their senses: it just released a Linux version of its VONaLink TeamRecord VoIP-based call-recording application. TeamRecord works with any VoIP phone system based on the SIP standard (think Asterisk), centrally recording all phone calls for a company's workgroup. VONaLink says that TeamRecord, for instance, allows "business transactions over the phone to be verified and disputes resolved" or comply with Sarbanes-Oxley or other disclosure provisions. Furthermore, TeamRecord replaces both expensive analog recording equipment and proprietary products from the dreaded phone company. The recording process involves unobtrusive monitoring of network packets via the port mirroring capability of a network switch and results in an inaudibly watermarked MP3 or WAV file. Users can listen to recordings of their own calls from a Web browser. TeamRecord is available for x86 Linux and Windows platforms.

www.arcosoft.com


VONaLink TeamRecord - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
VONaLink TeamRecord
admin
Feb 21 02:27 PM
Preferences Filter Administration Logout
Free space on /var: 95%
Switch to Folder
2007-02-19
Back Next

	No.	🔒	Time	Duration	Size	In/Out	Remote	Name	Local	Audio	Note	
□	1	OK	23:53	0:09	51k	In	1 905 737-2391	LEUNG T	1 647 477-5050	MP3 Phone	Sales	Edit Call
□	2	OK	22:12	0:17	266k	Out	1 777 123-4567		1 777 225-7274	WAV Phone	Investors	Edit Call
□	Select all			0:26	317k							

Delete Selected Calls Permanently
Powered by VONaLink

Dyne.org's dyne:bolic Linux Distribution

When not spending time in Amsterdam's offbeat cafés, the members of Dyne.org are busy developing dyne:bolic, upgraded to version 2.4, a live-or-installed Linux distribution focused on the needs of media-production fanatics. The main advantages of dyne:bolic include recognition of a wide variety of devices and peripherals, numerous tools for recording, editing, encoding and streaming audio and video, and the ability to run on lower-powered hardware. New in version 2.4 are improved user-friendliness via the Xfce-4.4 desktop, encrypted nests for preventing access to personal data stored in home directories, new (QParted) and updated (Cinelerra) software and a modularization of the inclusion of different kernels. You also can run (the previous edition of) dyne:bolic on "modded Xbox game consoles" and even cluster them! Raw ISO CD images can be downloaded for free or purchased for a minimal cost.

www.dynebolic.org

Avocent's MergePoint 5224 and 5240 Appliances

Two new arrows in Avocent's quiver are the MergePoint 5224 and 5240, appliances for controlling the service processors found in nearly any server. Service processors help manage servers independently of the main processor, controlling such functions as power, hardware monitoring and alerts. With its MergePoint appliances, Avocent claims to be the first company to "enable IT administrators to manage multiple service processors in Windows, Linux and UNIX servers from a single console", leveraging the "embedded management capabilities of servers already in their network". Product advantages include the ability to manage and control nearly all types of service processors (DRAC, iLO and RSA II) with a single gateway; increased efficiency through the unified utilization of service processors; reduced costs via consolidation of service-processor Ethernet ports; and added security through authentication, authorization and accounting features.

www.avocent.com

Sun Microsystem's OpenDocument Format (ODF) Plugin for Microsoft Office

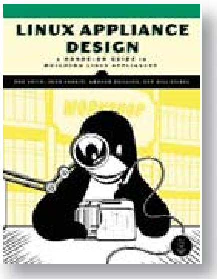
Monopolists need tools like this to keep them honest. You can now download Sun's plugin application for Microsoft Office 2003 that will allow for "seamless two-way conversion of Microsoft Word's documents to and from ODF". At the time of this writing, support for spreadsheets and presentations is due in April 2007. The conversion is claimed to be fully transparent to the user. Might this be the end of the beginning of the end of the Office stranglehold?

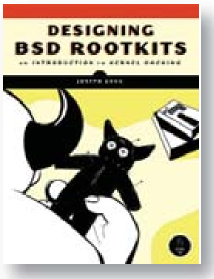
sun.com/openoffice

BakBone Software's NetVault: Backup APM for MySQL

Database admins should note that BakBone Software has released the new NetVault: Backup APM for MySQL, version 3.0. The solution provides low-complexity deployment and protection of the MySQL Enterprise and Community editions, consolidated backup and recovery and a common administrative UI that allows users to set up, configure and define a wide range of backup policies and scenarios. Additional highlights include full, incremental and differential backups while data is on-line and accessible; a common UI across multiple storage engines; consolidation of multiple storage engines into a single job; and protection down to the table level. The product is a MySQL Enterprise Gold Certified solution.

www.bakbone.com





No Starch Press Titles

Here's the deal. Right now, No Starch Press is giving life to so many great titles, I'm not completely sure which single title will bring you the most geek enlightenment. So, let's give the coolest ones some abridged love, shall we? First, there's *Linux Appliance Design* by Bob Smith, John Hardin, Graham Phillips and Bill Pierce. Although many books tell readers how to run Linux on embedded hardware or how to build a Linux application, No Starch says this is the first title to demonstrate how to merge the two to create a Linux appliance. The CD includes a prototype appliance—a home alarm system—that readers can use and modify. Next up, because we know many of you do BSD, there's *Designing BSD Rootkits: An Introduction to Kernel Hacking* by Joseph Kong. Written in a cheeky style with lots of geek humor, the book covers the fundamentals of programming and developing rootkits under the FreeBSD operating system. Finally, *The Book of Qt 4* by Daniel Molkentin, a core KDE developer, shows readers how to build applications both with and without Qt's graphical GUI builder, Qt Designer.

www.nostarch.com

Please send information about releases of Linux-related products to James Gray at newproducts@linuxjournal.com or New Products c/o *Linux Journal*, 1752 NW Market Street, #200, Seattle, WA 98107. Submissions are edited for length and content.

Zimbra Collaboration Suite, Version 4.5

Zimbra exploits Ajax for a compelling Web groupware experience. MARCEL GAGNÉ

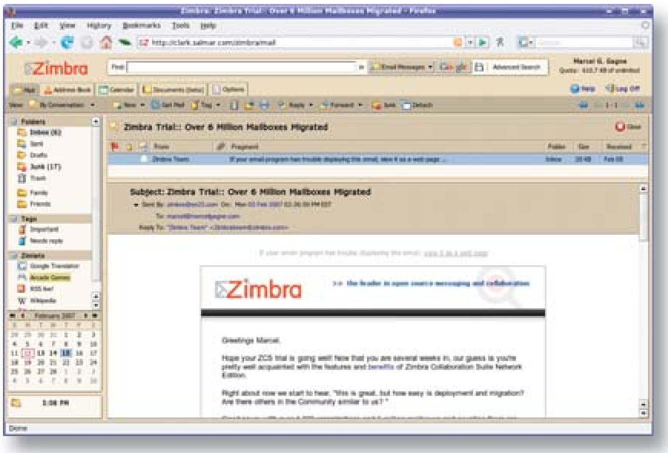
E-mail is the backbone of modern corporate communication, and frankly, it has been for a long time. As with any tool that gets this much use and features so prominently in any networked business, e-mail has demonstrated a host of limitations. To improve the experience, software developers have integrated certain obvious functions, such as calendaring, simple mailing lists and address books, thereby creating what is popularly referred to as a contact management system.

The great leap for enterprise contact management was the creation of groupware, or collaboration tools, allowing those calendars and contact lists to be shared among members of an organization. Administering complex packages on thousands of personal computer desktops creates its own set of problems though. As these increasingly complex systems grew, users noticed that they were now drowning in a sea of words, always trying to reach for that elusive piece of information that would help them stay afloat.

Zimbra's raison d'être is to improve the e-mail experience by providing a suite that simplifies as well as enhances the ease with which users can sort through and retrieve information.

The Zimbra Collaboration Suite (Figure 1) is a powerful, scalable application, suited for small to medium to very large enterprises. Zimbra can be deployed in a high-availability, clustered environment where it can serve up huge numbers of clients, and serve them it does. For instance, Zimbra customer H&R Block serves up 100,000 users, while the University of Toronto's numbers are in the tens of thousands. Zimbra's customers are varied (for example, Mozilla and Digg.com), from ISPs and content providers who offer Zimbra as a service to businesses, universities, government offices and nonprofits.

I've had a few weeks to play with the Zimbra Collaboration Suite, and I am, overall, very impressed. That said, nothing is perfect—more details as I go along.



Zimbra Trial:: Over 6 Million Mailboxes Migrated

Subject: Zimbra Trial:: Over 6 Million Mailboxes Migrated

If your email program has trouble displaying this email, view it as a web page

Zimbra >> the leader in open source messaging and collaboration

Greetings Marcel,

Hope your ZCS trial is going well! Now that you are several weeks in, our guess is you're pretty well acquainted with the features and benefits of Zimbra Collaboration Suite Network Edition.

Right about now we start to hear, "this is great, but how easy is deployment and migration? Are there others in the Community similar to us?"

Figure 1. The Zimbra Collaboration Suite, Ajax Client View

Note:

When speaking with Zimbra, I asked about the service model where customers sign up for Zimbra e-mail accounts, and where those accounts are ad-supported or value-add is provided through Zimbra extensions. Zimbra informed me that about 80% of Zimbra installations are on premise deployments. The company doesn't believe that e-mail is going to a hosted model as with CRM.

What Does Zimbra Do?

The Zimbra Collaboration Suite belongs in a class of network applications we've come to call groupware—an intelligent collection of CRM, of which e-mail is the core, usually provided through a Web interface. In addition to e-mail, the Zimbra messaging server supports shared group calendars and contact management tools. Zimbra is both a client-side solution and a powerful integrated messaging server application that includes Postfix, LDAP, Apache and more. Zimbra includes some clever extensions, such as zimlets (more on these later), and enterprise mashups that allow users to interact naturally with information embedded in their e-mail messages, so they easily can use that information in other applications. These applications even can be third-party applications, such as a purchasing system.

It's impossible to talk about a product like Zimbra without making comparisons to Microsoft Exchange, so I answer the obvious questions here. Zimbra is an impressive replacement for Microsoft Exchange. Zimbra offers migration tools that simplify the move from Exchange to Zimbra (Lotus Notes migration tools also are available). Outlook clients are fully supported, and that means it works with all Outlook e-mail, contacts and calendar functions (MAPI sync, however, is available only with the Zimbra Network Edition). If you happen to be fond of another mail client, such as Thunderbird or Eudora, you can continue using it. That said, once you start working with Zimbra's impressive browser-based Ajax client, you may say goodbye forever to your old client software. Zimbra also supports a number of mobile devices, such as the iconic BlackBerry.

As the above paragraph hints, Zimbra is available in different flavors, including a community-supported Open Source Edition and a commercially supported Network Edition.

Working with Zimbra

Zimbra is a Web application that doesn't feel like a Web application.

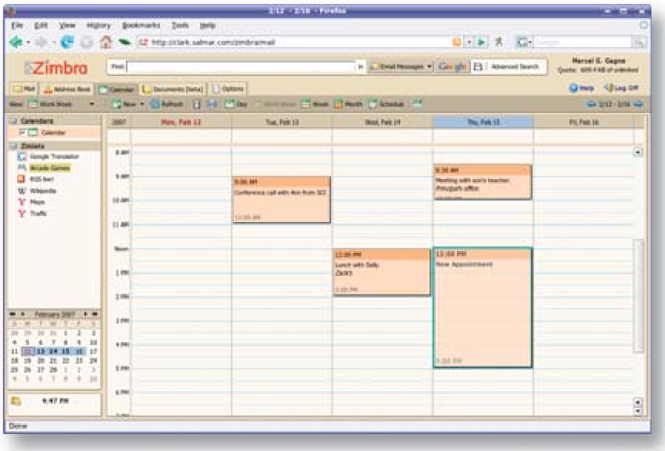


Figure 2. Zimbra incorporates an easy-to-use and powerful shared calendar application that is tightly integrated with the rest of the suite.

Several very cool features take Ajax design to a very advanced destination. For instance, Zimbra provides natural keyboard mapping.

Let me give you an example. With most Web mail applications, you would delete a message (or a batch of messages) by selecting each message with a click in a check box beside the message, finding the Delete button and clicking there. Zimbra works the way you expect your desktop application to work. Simply select a group of messages (click, Shift-click, or Ctrl-click), and then press the Delete key.

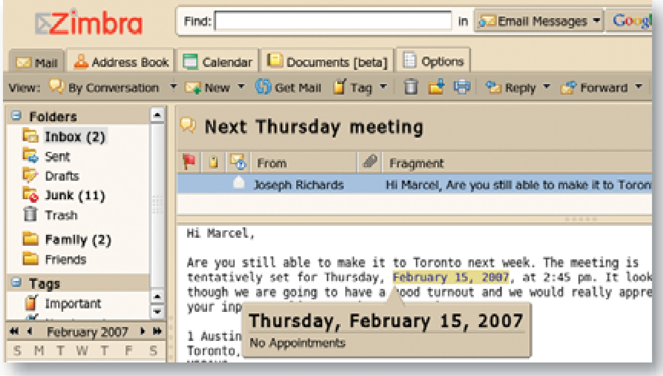
Let me give you another example. In your run-of-the-mill Web mail application, you move messages into a folder by selecting them via a check box, clicking a "move to" button, clicking on the destination folder from a drop-down list, then clicking Move. Zimbra users select their messages and drag them into the appropriate folder—exactly like you would on your desktop. To switch folders, double-click on the folder. To read a message, double-click on the message. The key mapping is so natural that after a while, you forget you are working with a Web application.

This natural, or at least familiar, approach is consistent across all Zimbra applications. In the calendar view, creating an appointment is simply a matter of dragging your mouse across a time slot, and Zimbra pops up an appointment dialog for you to fill in (Figure 2).

When working with this sort of application, it's not unusual to check your e-mail, then switch to calendar view to confirm your free time, and then go back to your e-mail again. Zimbra can recognize strings inside your e-mail messages, such as a phone number or a date. Figure 3 shows me looking at an e-mail message in which the sender is asking whether I'm free for a particular date. Zimbra recognizes that this is a date and informs me of my availability for that time slot.

Note:

Zimbra's Ajax interface is supported on several different browsers, including Firefox 1.0+, Netscape 7.1+, Internet Explorer 6.0+, Mozilla 1.4+ and Safari 1.3+ (although that is currently beta support). When I tried to use Zimbra with Opera and Konqueror, I was stopped with an error message listing the supported browsers. With browsers such as Konqueror, it's possible to change the browser identification and masquerade as something else. After doing that, Zimbra allowed me to continue, but I still ran into problems. Apparently, it means it when it says unsupported. For those unsupported browsers, Zimbra offers its basic interface, with a link provided on the error page.



Next Thursday meeting

From	Fragment
Joseph Richards	Hi Marcel, Are you still able to make it to Toron

```
Hi Marcel,

Are you still able to make it to Toronto next week. The meeting is
tentatively set for Thursday, February 15, 2007, at 2:45 pm. It look
though we are going to have a good turnout and we would really appre
your inp

1 Austin
Toronto,
```

Thursday, February 15, 2007
No Appointments

Figure 3. No need to switch from your e-mail to your calendar. Zimbra gives you the information you need, inline.

If the time slot is free, there's no need to switch to calendar view here either—simply right-click on the date, select New Appointment and then enter the details. It's wonderfully simple and intuitive.

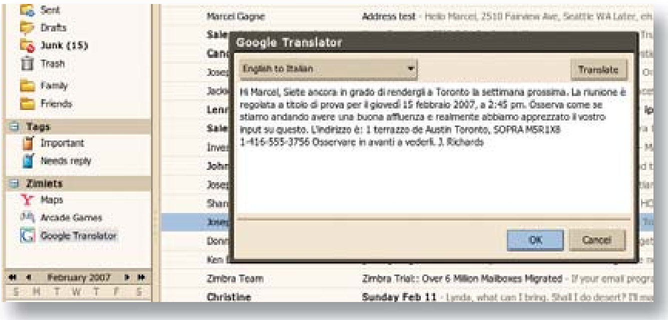
It gets better. Zimbra also is able to recognize other contextual information (highlighted in blue), such as the word tomorrow, or even multiple words, such as next Wednesday. The program then checks your calendar for you when you hover over the word (or words). Zimbra also understands addresses and phone numbers, highlighting them as well. If it sees a phone number, Zimbra offers to launch a telephony program such as Skype or Ekiga. Hover over an e-mail address, and Zimbra looks up that contact in your address book and floats a pop-up with that person's contact information. No need to leave your e-mail and switch to your address book to call your client.

The user experience is enriched further with built-in antispam and antivirus tools, smart folders, RSS subscriptions and, of course, zimlets.

Zimlets

When you hover over a time in an e-mail message and a pop-up alerts you of your appointments for that time, you're seeing the work of a zimlet. A zimlet can launch Skype when you hover over a phone number. A few important zimlets are included in a default Zimbra installation, but there are several of which you might not immediately be aware, ranging from extremely useful to amusing distractions. When you install (or deploy) these extra zimlets, they appear in your sidebar, right above your mini-calendar.

Some of the extra zimlets are worth exploring. In addition to the phone dialer (which uses Skype by default), there's also a zimlet that takes advantage of an Asterisk VoIP system. Deploy the Yahoo! Maps zimlet, and Zimbra recognizes addresses embedded in text and pops up a street map when you hover over the address. This function didn't catch all address strings I tested, but you can enter the address manually by clicking on the zimlet. There's a Wikipedia zimlet and a Google translator zimlet as well (Figure 4). Simply drag a message onto the Google translator zimlet, select a language and click OK.



Hi Marcel, Siete ancora in grado di rendergli a Toronto la settimana prossima. La riunione è regolata a titolo di prova per il giovedì 15 febbraio 2007, a 2:45 pm. Osserva come se stiamo andando avere una buona affluenza e realmente abbiamo apprezzato il vostro input su questo. L'indirizzo è: 1 terrazzo de Austin Toronto, SOPRA M5R1X8 1-416-555-3756 Osservare in avanti a vederli. J. Richards

Figure 4. No cutting and pasting required for translations. Simply drag a message onto the Google Translator zimlet.

As much as I hate to admit it, the zimlet I wasted far too much time on is the arcade zimlet (call it a nostalgia trip). Several classic arcade games are included to provide some much-needed diversion. After all, all work and no play does not necessarily translate into increased productivity. Spend a few minutes blasting *Asteroids*, helping *Frogger* cross the street, saving the Earth from the evil *Space Invaders* or de-stress with a little game of *Pac-Man* (Figure 5).

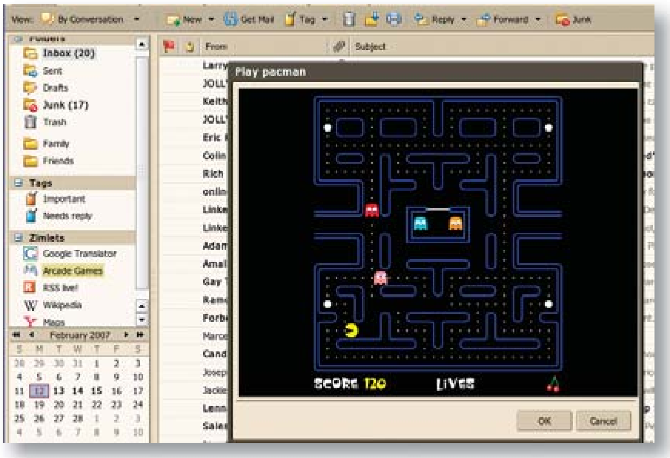
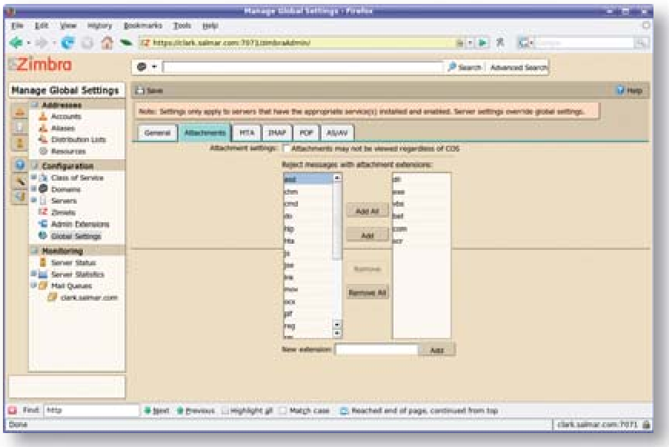


Figure 5. How about a little downtime with a classic game of *Pac Man*?

There are also zimlets for traffic reports (letting your employees know when it's a good time to go home or what areas to avoid), and zimlets for Amazon.com sales, tracking flights, booking travel arrangements, looking up the status of an order or package, sending short messages (SMS) to a cell phone and more.

Zimlets can be installed manually (via the command line) or through the graphical Zimbra administrative interface, which I would recommend. The administrative interface is as well designed as the user interface but provides access to different functions. These include reporting and system statistics, user maintenance, global address lists, aliases, as well as server, site and domain configuration. Settings related to security are configured here as well. For instance, you may want to block out attachments with certain extensions, such as .BAT, .EXE, .VBS and so on (Figure 6).

The administrative interface also provides download links to migration tools for Exchange and Lotus Domino servers. There's also a download there to help you import information from PST files.



Note: Settings only apply to servers that have the appropriate service(s) installed and enabled. Server settings override global settings.

Attachment settings: □ Attachments may not be viewed regardless of COS

Reject messages with attachment extensions:

adi, chm, cmd, do, hlp, hta, js, jse, lnk, mov, ocx, pif, reg

pl, exe, vbs, bat, com, scr

New extension:

Reached end of page, continued from top

Figure 6. The administrative interface lets you reject messages whose attachments have questionable extensions.

Any Minuses?

Well, yes. Browser support could be a little better, but it covers the majority of browsers and certainly the most popular. I personally would like to see Konqueror supported and Opera as well. Some of the zimlets didn't always do what they were supposed to do—for example, the Yahoo! Maps zimlet sometimes highlighted addresses and sometimes didn't. Even there, I could double-click on the zimlet, enter the address manually, and it would come up.

The biggest negative, in my opinion, had to do with getting Zimbra up and running in the first place. The command-line text-install process feels a little last-century to me. Not that there's anything wrong with the command line—you're looking at one of the great proponents of the command line's power and flexibility. Nevertheless, having the installation shell script terminate to tell me I need another package, then having to restart it to discover I needed something else (then having it terminate again), is far from ideal.

Then comes the configuration part of the text install where parts of the dialog scroll off the screen. Sure, I can scroll back, but why not make it fit on a screen? Heck, a Web interface that assisted you in dealing with any prerequisite or configuration issues should be easy for a company that can come up with such a slick client interface.

I blame most of my installation problems on that inflexible text-only script. Before finding myself with a finished product, I ran three different installations. I tried the first install on my own production system, assuming (falsely) that I could run Zimbra concurrently—a bad move that took my own e-mail and Web services off-line for a few hours. To be fair, I can't really blame Zimbra here, because I should not have been doing that on a production system, but I often install and test software on my production systems without any problems. A simple warning that Zimbra's Postfix and Apache servers would occupy the same network space as mine (and that I might want to reconsider) would have stopped me in my tracks.

Eventually, I chose another clean system for my installation, and it just plain refused to finish. The generic error message told me nothing. Because it was a supported release, I tried again, re-installing (as opposed to upgrading), and everything worked perfectly. Why? This was the same system where it failed a few minutes before, so I don't have a good explanation.

When I finally got Zimbra installed, it was such a great experience I almost forgot about my installation headaches—almost.

Licensing and Costs

Zimbra is a free and open-source package, but it also offers commercial packages and support. The Network Edition offers full commercial support plus additional, value-added features. For instance, clustering, including advanced backup and recovery features, and a powerful attachment search function, aren't part of the Open Source Edition. Neither are the Outlook MAPI and Apple iSync connectors; for these, you need the Network Edition. Over-the-air mobile synchronization (Simbian, Treo and Windows mobile phones with no additional software required on the handset) is another such value-add. Blackberry support is available through a third party. (Note to RIM—it would be really, really great if there were a general-purpose API for BEZ synchronization.)

The Zimbra Collaboration Suite is available for a variety of distributions and platforms. If packages for your particular distribution aren't available, Zimbra provides source so you can compile your own. Both Network and Open Source Editions are released at the same time. Zimbra doesn't want the installation to be different, regardless of whether it is the Network or Open Source Edition. Even so, only a few major distributions (such as Red Hat Enterprise Linux and SUSE Linux Enterprise Server) are supported for the Network Edition. If you are interested in the Network Edition, you may, for the time being, want to stick with the Open Source Edition.

If yours is one of the supported Network Edition platforms, I recommend that you download the Network Edition first. A free, 60-day trial is included, which provides you with all the Network Edition features. If you choose not to continue with the Network Edition after the 60-day period expires, your Zimbra Collaboration Suite automatically returns to the Open Source Edition, and you'll have lost nothing.

Prices for the Network Edition are reasonable, starting at $25 US per user, per year. Special discounted rates are available for educational institutions, governments and nonprofits.

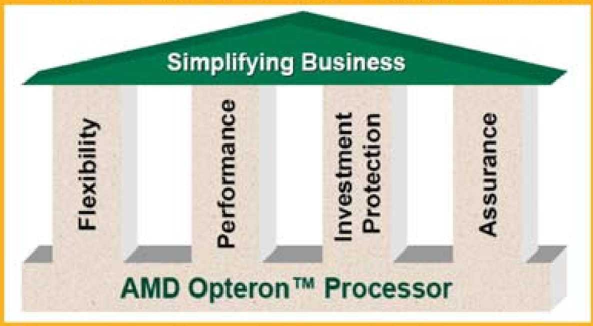
Conclusions

Color me impressed! The Zimbra Collaboration Suite is a fantastic product and well worth your consideration. Working with Zimbra's polished Ajax client is a pleasure, and even the Open Source Edition is feature-rich. Some of the zimlets, though imperfect (such as the Yahoo! Maps zimlet), still provide a great improvement to the standard groupware experience.

Installation, on the other hand, could be a lot smoother, and I'd like to see more distributions supported in the Network Edition, but these installation-time issues don't affect the user experience.

If you don't need the features provided by the Network Edition, the price for Zimbra is certainly right, offering a great deal of functionality without the cost. If you want to take advantage of the Zimbra features but don't want to do the hosting on your own, Zimbra provides a list of hosting partners at **www.zimbra.com/partners/zimbra_hosting.html**. The main Zimbra site is **www.zimbra.com**.■

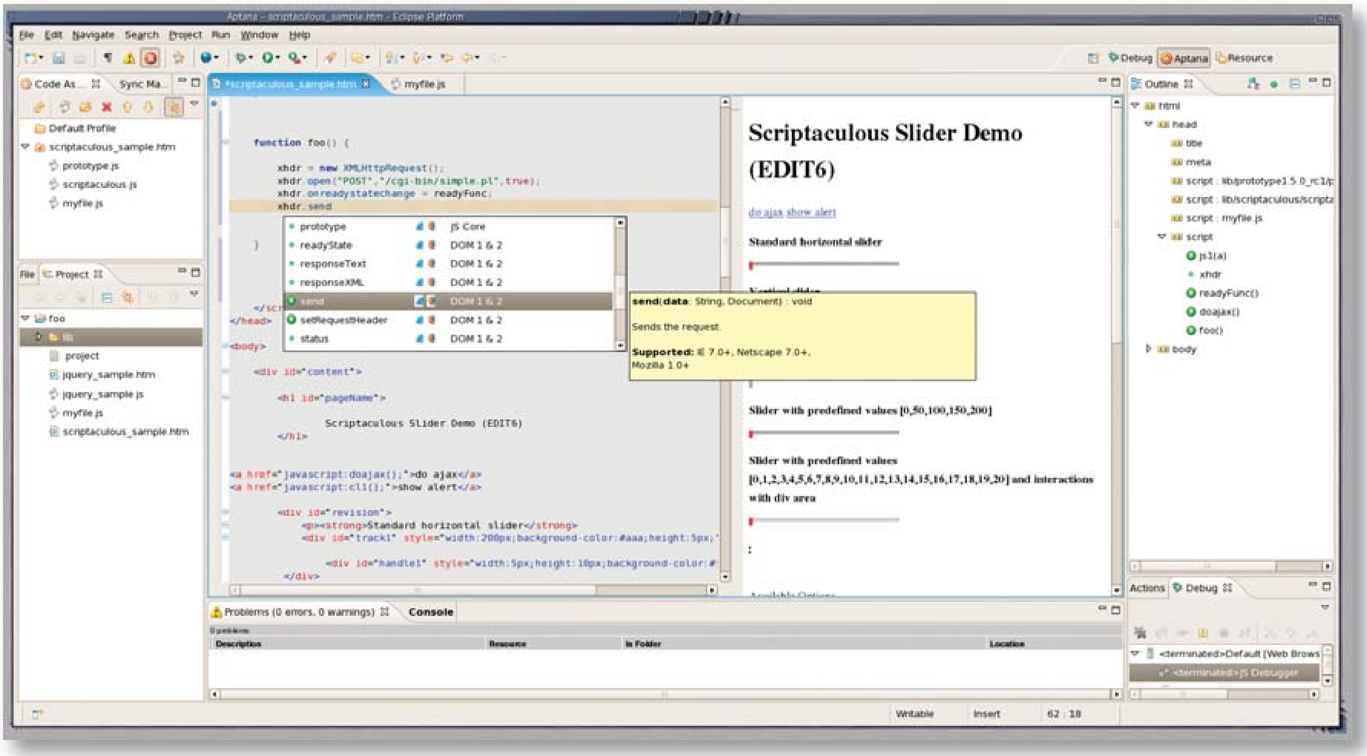
Marcel Gagné is an award-winning writer living in Waterloo, Ontario. He is the author of the all-new *Moving to Free Software*, his sixth book from Addison-Wesley. He also makes regular television appearances as Call for Help's Linux guy. Marcel is also a pilot, a former Top-40 disc jockey, writes science fiction and fantasy, and folds a mean Origami T-Rex. He can be reached via e-mail at mggagne@salmar.com. You can discover lots of other things (including great Wine links) from his Web site at www.marcelgagne.com.



The Aptana IDE for Ajax Development

Aptana is a mixed-content IDE for Web development.

BEN MARTIN



```
function foo() {

    xhdr = new XMLHttpRequest();
    xhdr.open("POST","/cgi-bin/simple.pl",true);
    xhdr.onreadystatechange = readyFunc;
    xhdr.send

}

</sc
</head>

<body>

  <div id="content">

    <h1 id="pageName">

          Scriptaculous Slider Demo (EDIT6)
    </h1>

<a href="javascript:doajax();">do ajax</a>
<a href="javascript:cl1();">show alert</a>

    <div id="revision">
      <p><strong>Standard horizontal slider</strong>
      <div id="track1" style="width:200px;background-color:#aaa;height:5px;"

        <div id="handle1" style="width:5px;height:10px;background-color:#
     </div>
```

```
prototype          JS Core
readyState         DOM 1 & 2
responseText       DOM 1 & 2
responseXML        DOM 1 & 2
send               DOM 1 & 2
setRequestHeader   DOM 1 & 2
status             DOM 1 & 2
```

send(data: String, Document) : void

Sends the request.

Supported: IE 7.0+, Netscape 7.0+, Mozilla 1.0+

Scriptaculous Slider Demo (EDIT6)

do ajax show alert

Standard horizontal slider

Slider with predefined values [0,50,100,150,200]

Slider with predefined values [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20] and interactions with div area

Figure 1. Aptana performing code assist for the send() method of an XMLHttpRequest. Methods for the object are shown, and for the current candidate method, the arguments and minimal browser requirements are shown as extra assistance. The outline view on the far right allows quick navigation through HTML, CSS and JavaScript mixed content. In the middle right is a preview of the current HTML page being edited in the middle left.

Aptana is an IDE for HTML, CSS and JavaScript coding based on Eclipse. I reviewed Aptana using Fedora Core 6 on a modern dual-core machine with 2GB of RAM. Installing Aptana can be a little difficult with Fedora Core 6 and may be so for other modern Linux distributions as well.

Initially, I attempted the install with the "zip (no installer)" download, which is about 43MB without the Java runtime. I extracted the zip file and ran the Aptana script with the Sun JDK 1.5.0 and 1.6.0, but in both cases, the IDE refused to start.

So, I changed to the "BIN Plus Java Runtime 1.4.2" download, which is 74MB, and the Aptana_IDE_Setup.bin file refused to execute with a libc.so.6 error. After searching the forums, I found that you can resolve this issue with a little use of sed (see Resources). Once the graphical installer was complete, Aptana still refused to start with an "XPCOM error -2147467262".

The fix for the XPCOM issue is to install Mozilla and export MOZILLA_FIVE_HOME to Mozilla's base path. Unfortunately, Mozilla is no longer in Fedora Linux. I found that with the BIN plus runtime installation, after a short period of time, the JVM would SEGV with "Too many open files".

A much better method is to install Eclipse from your distribution's repository and then install the Aptana plugin.

The IDE offers syntax highlighting, code completion, file navigation through document structure (going to parts of the DOM or JavaScript functions by name), breakpoint debugging of code running in Firefox and some synchronization options.

Syntax highlighting is context-sensitive and works quite nicely across mixed HTML, CSS, PHP and JavaScript content. Code completion didn't work on the little PHP I played with, but it worked quite well for JavaScript—the one exception was when I created a variable for a new instance of XMLHttpRequest, the method name code completion didn't

Aptana

The Debatable:

- The use of Eclipse can be a polarizing factor.

The Good:

- The syntax highlighting and completion works well across HTML, CSS and JavaScript mixed content.
- Setting breakpoints and stepping around JavaScript code running in a Firefox browser launched from Aptana.

The Bad:

- Using Aptana breakpoint debugging with Ajax XMLHttpRequest calls is not simple.
- Sometimes things become a little slow, even on grunty hardware.
- Debug mode sometimes requires that you restart Aptana to recover.

initially work for that variable. This seemed to sort itself out though. Code completion on the browser DOM also shows the browser and version of it in which that feature is supported. According to an August 2006 Web interview with Paul Colton, founder of Aptana, the ability to set requirements for a set of browsers and versions and have the IDE flag methods outside these requirements as errors is a planned feature.

Aptana allows you to create a new JavaScript project and select from a bunch of third-party libraries, including Aflax, Dojo, MochiKit, Mootools, Prototype, Rico, Scriptaculous, Yahoo UI, jQuery and yui-ext.

For HTML files, you can have a browser either in another tab or horizontally/vertically next to the file you are editing. This browser is reloaded automatically when you save the HTML. This makes a hack-and-see session for some JavaScript fairly quick.

Selecting debug from the IDE caused Firefox to run and installed the extension so that Aptana could talk to Firefox. With this combination, you can set breakpoints in Aptana, run debug from the IDE (which loads the page into Firefox), and when a breakpoint is hit in Firefox, Aptana springs to the front at that breakpoint. The integration of the IDE with Firefox works well; however, a few times when Aptana became slow, I had to restart Eclipse to recover. This may be a result of my configuration and not Aptana, although I was running Fedora Eclipse with only the Aptana plugin and a bare-bones Firefox setup.

As far as Ajax support, an Ajax Monitor is available that shows the requests and responses issued. Unfortunately, the preview for an HTML file you are editing most surely will bomb out with a permission-denied error if you attempt Ajax with XMLHttpRequest from it. If you use the debug mode in Aptana to start an external instance of Firefox, XMLHttpRequest likely will not work as well. The Firefox instance that Aptana starts will load a URL, such as http://127.0.0.1:8001/foo.htm, which is served by a Web server that Aptana includes. Assume there is an XMLHttpRequest for a simple path, such as /cgi-bin/foo.pl, to make sure you request from the originating server and avoid the permission-denied error from the browser. Then, the browser will ask the Web server running in Aptana on port 8001 for this resource and fail. It would be nice if you could configure the Aptana Web server to proxy some requests to a server and return the result back to the Firefox instance you are using for debugging. If, for example, you change the URL in the Firefox instance that Aptana started for debugging to a version of the site served by Apache, such as http://localhost/foo/myapp.html, your Ajax requests might function properly, but Aptana breakpoints no longer will work.

It would be nice to be able to set up breakpoints in the Ajax Monitor, so the JavaScript onreadystatechange callback would be halted—for example, if an XPath2 matched against the response of a request. You can work around this by setting a suitably verbose expression breakpoint in the JavaScript function, but it would be better if it were integrated more directly into the Ajax Monitor itself.

The IDE includes support for synchronizing files with FTP or SFTP servers, but rsync support is currently missing.

The Aptana screen casts page is a good resource for quickly seeing Aptana's features and how to use them; however, some of the presentations may be less useful for people who are familiar with IDEs in general.

This review has highlighted some short-comings (particularly to do with XMLHttpRequest and debugging stability) and a few bugs here and there. But, overall, Aptana is a very nice IDE for hacking some HTML, CSS, JavaScript and possibly PHP code.■

Ben Martin has been working on filesystems for more than ten years. He is currently working toward a PhD on combining Semantic Filesystems with Formal Concept Analysis to improve human-filesystem interaction.

Resources

Aptana: **www.aptana.com**

Aptana Screen Casts: **www.aptana.tv**

Fedora Eclipse: **sources.redhat.com/eclipse**

Linux Installation Issues: **www.aptana.com/forums/viewtopic.php?t=134**

Installing Aptana on Linux: **www.aptana.com/docs/index.php/Installing_Aptana_on_Linux**

Mozilla Issue and Aptana: **www.aptana.com/forums/viewtopic.php?t=169&view=next**

Tutorials: **www.aptana.com/docs/index.php/Aptana_Tutorials**

Using the Ajax Monitor View (although there's no information about getting around the permission problem for Aptana Web server sessions): **www.aptana.com/docs/index.php/Using_the_AJAX_Monitor_View**

Interview with Paul Colton, Founder of Aptana: **playbacktime.com/2006/08/30/interview-paul-colton-founder-of-aptana-ajax-web20**

Open-Source Databases, Part II: PostgreSQL

Feature-rich PostgreSQL delivers on database integrity. REUVEN M. LERNER

If you're developing an application that depends on storing and retrieving large amounts of data, you undoubtedly have discovered how useful a relational database can be. Databases have been around for a while, but for many years, the open-source offerings were inferior to their commercial counterparts. Today, we have several open-source options from which to choose. Last month, I gave an overview of MySQL, which is probably the most popular open-source, client-server relational database.

This month, we look at MySQL's best-known competitor, PostgreSQL. (Next month, I plan to compare the two programs.) PostgreSQL has a smaller community than MySQL, which sometimes leads people to write it off, think of it as unimportant or even believe that it's not as powerful as MySQL. But, PostgreSQL includes nearly all the functionality of MySQL, and it offers a large number of features that MySQL does not currently offer. As this article shows, it's worth considering PostgreSQL any time you need a database.

History

PostgreSQL began as a research project at the University of California, Berkeley, in 1985. Michael Stonebreaker, a professor of computer science, already had developed and released one database product known as Ingres. After commercializing Ingres, he returned to Berkeley and designed a new database (Postgres), meant in part to solve the problems he had seen with Ingres. Despite a growing community of users, Stonebreaker shut down the Postgres project in 1993.

However, Postgres was distributed under the BSD open-source license, which meant users were free to modify and redistribute it. Two Berkeley students, along with a handful of people from elsewhere in North America, decided to see how easily they could modify the code and distribute a working database.

One of the changes these developers made was in the way Postgres communicated to the outside world. Originally, Postgres had used the QUEL query language that Stonebreaker had designed for Ingres. The developers removed QUEL and added an SQL interpreter to be more compatible with other products. To indicate that this version of Postgres used SQL, the database was renamed PostgreSQL.

Today, many of those original developers continue to work with the project, fixing bugs and contributing patches. The effort is coordinated by a volunteer steering committee. Individual PostgreSQL developers may be hired (as employees or contractors) by various companies, but no one company or organization controls the development or direction of PostgreSQL as a whole.

The most recent release of PostgreSQL is 8.2.3, released in early February 2007. This was a bug-fix release for version 8.2, which was released in late 2006. Major releases typically come out once per year, with additional minor releases for security and other serious bugs.

Installation

The easiest way to install PostgreSQL is to use a packaging system, such as debs or RPMs. On my Ubuntu system, for example, I was able to install PostgreSQL easily and quickly with apt-get. Note that most packaging systems distinguish between the PostgreSQL client, server and developer libraries, so be sure to retrieve the packages that are most appropriate for your needs. Installing PostgreSQL via apt-get or RPMs should achieve everything you need to get started—from creating a postgres user to initializing a data directory.

If you must install PostgreSQL from source, I suggest reading the documentation that comes with it. Compiling PostgreSQL is not hard, but it does require more description than I have room for in this article. One piece of advice though, for anyone compiling it from scratch, is to unpack the archive, as well as compile and test it, as the postgres user. Trying to compile and test PostgreSQL as the root user is bound to fail, and other users also might not have sufficient privileges for it to work correctly.

Now, we're ready to start up the server. Prebuilt packages generally will include a shell script (to be placed in /etc/init.d/ or the equivalent) that starts the server for you. Even if you have downloaded and installed the source code for PostgreSQL, you will find an appropriate startup script in the contrib directory. I suggest using (or at least modifying) this script rather than writing one from scratch. On my system, I can start PostgreSQL with:

```
/etc/init.d/postgresql start
```

Simple Connections

If all goes well, our server should now be running. (We can check this by typing `ps aux | grep postgres` at the command line.) The easiest way to access the server is to use the psql interactive client, which comes with PostgreSQL. To get a list of databases in the current cluster, use the following syntax:

```
psql -U <username> -l
```

where <username> is a PostgreSQL user with sufficient access privileges on the server. The -U option lets us indicate the user name, and the -l option asks the server to list those databases that are available, such as:

```
List of databases
    Name    |    Owner   | Encoding
------------+------------+-----------
testserver  | reuven     | SQL_ASCII
postgres    | postgres   | SQL_ASCII
template0   | postgres   | SQL_ASCII
template1   | postgres   | SQL_ASCII
(4 rows)
```

Like many database systems, PostgreSQL maintains its own list of users and groups separately from the underlying operating system. To create a new user, we use the createuser program, which comes with PostgreSQL:

```
createuser -U postgres reuven
```

This invokes createuser as postgres (which has permission to create other users) and then creates a new user named reuven. If we make this new user a superuser, reuven also will be allowed to create new databases and roles.

Now, we can create a new database in this cluster:

```
createdb -U reuven linux
```

Double-check that the database exists with a reuse of psql -l:

```
psql -U reuven -l
```

You might have noticed that we have not given a password any time we have invoked a command that required a user. The default setting for PostgreSQL makes the server available via a local socket (and thus unavailable over the network). Because only local users will be allowed access, we allow connections from any defined user name, even without a password.

We can change this behavior, as well as other security- and connection-related behaviors, in the pg_hba.conf file located at the top of the cluster directory. The file contains extensive documentation and explains how to set up the connection parameters.

Creating a Table

Let's connect to our database and see what happens:

```
$ psql -U reuven linux
```

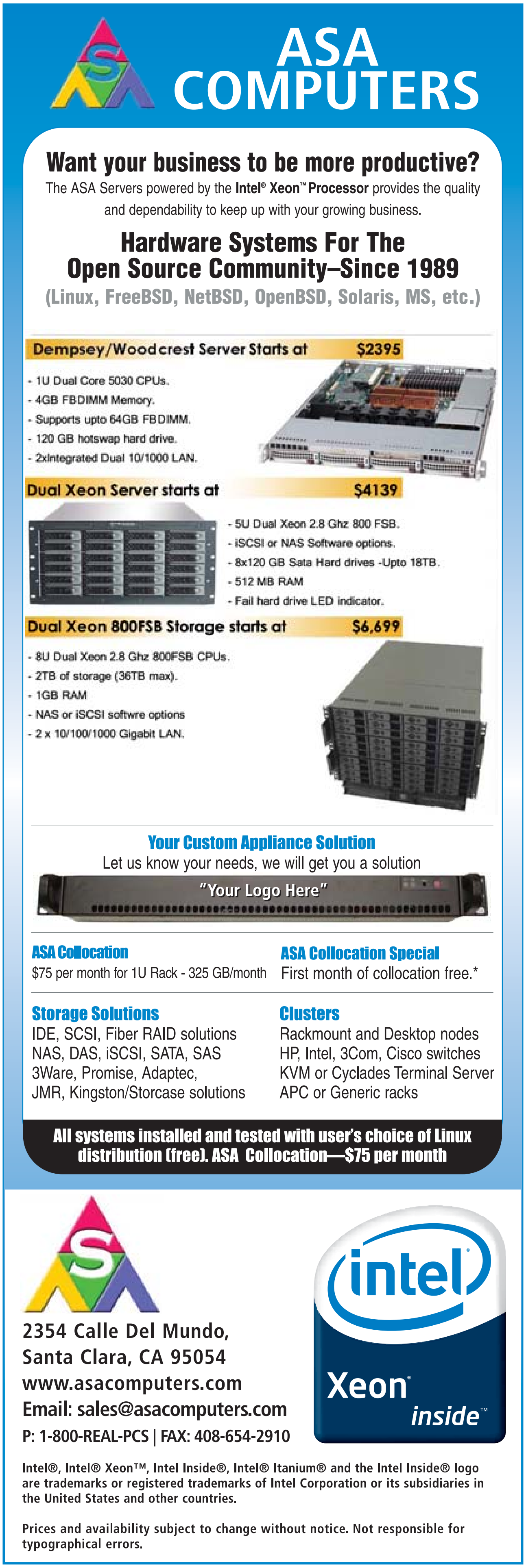
Sure enough, we get the psql prompt, inviting us to enter a query. A list of psql comments is available by typing \? at the prompt. We also can get help with SQL syntax and commands with \h, as in:

```
\h CREATE TABLE
```

Sure enough, let's start by creating a table:

```
CREATE TABLE People (
    id     SERIAL     NOT NULL,
    first_name     TEXT     NOT NULL,
    last_name     TEXT     NOT NULL,
    email_address     TEXT NOT NULL,
    added_at     TIMESTAMP   NOT NULL   DEFAULT NOW(),

    PRIMARY KEY(id),
    UNIQUE(email_address)
);
```

This table has five columns and three different data types. The most common type is TEXT, which is the typical way to store textual data. PostgreSQL has full support for Unicode, and TEXT columns may contain very long strings. (Until version 8.0,



PostgreSQL could store a limited amount of data in each row, so this might come as news to some people.) There is no built-in support for indexing TEXT columns, but an add-on module, called tsearch2, is relatively easy to install and provides such functionality for anyone who needs it.

The added_at column is of type TIMESTAMP. PostgreSQL provides a very sophisticated set of time- and date-handling routines, thanks to the division between two basic data types, known as TIMESTAMP and INTERVAL. A timestamp indicates when something happened, whereas an interval tells you how long something took. We thus can subtract two timestamps (to get an interval) or add an interval to a timestamp (to get a new timestamp). Moreover, PostgreSQL lets us define intervals using English-like syntax, such as:

```
SELECT id FROM People WHERE added_at > NOW ()
 ➥- interval '30 days';
```

The above query shows us all of the people who were added to our contact database in the last 30 days.

The added_at column defines a DEFAULT value as well. This means that if we fail to INSERT an explicit value for added_at, PostgreSQL will use the current time (at the time of insertion).

The SERIAL Data Type

The id column, which we define to be our primary key, uses a SERIAL data type. The thing is, SERIAL isn't a data type at all. Rather, it's syntactic sugar that does several things:

- It creates a new sequence object, whose values are integers that start at 1 and increase each time we ask for a value.
- It defines the column type to be INTEGER.
- It sets the DEFAULT clause for our column to be the result of requesting a new value from the sequence.

This might sound like a complicated way of saying, "SERIAL gives us an auto-incrementing column". And that's true, in a sense, but you can have as many SERIAL columns as you like in a table, and each sequence can have all sorts of properties associated with it, including its starting point and increment.

Finally, by defining id to be a primary key and email_address to be unique, we implicitly ask PostgreSQL to create indexes on these two columns. When we execute the above query, PostgreSQL notifies us what it's doing behind the scenes:

```
NOTICE:  CREATE TABLE will create implicit sequence
"people_id_seq" for serial column "people.id"
NOTICE:  CREATE TABLE / PRIMARY KEY will create
implicit index "people_pkey" for table "people"
NOTICE:  CREATE TABLE / UNIQUE will create implicit
index "people_email_address_key" for table "people"
```

If and when you drop the People table, these implicitly defined objects are dropped automatically.

To list all of the tables, sequences and views in our database, we can use the \d command:

```
linux=# \d
      List of relations
 Schema |      Name      |   Type   | Owner
--------+----------------+----------+--------
 public | people         | table    | reuven
 public | people_id_seq  | sequence | reuven
(2 rows)
```

We can add a letter to \d to get a list of only tables (t), indexes (i), sequences (s), functions (f) or views (v). For example, here is a list of the indexes that we have created:

```
linux=# \di
            List of relations
Schema |           Name            | Type  | Owner  | Table
-------+---------------------------+-------+--------+-------
public | people_email_address_key | index | reuven | people
public | people_pkey               | index | reuven | people
(2 rows)
```

We can also use \d to inspect a particular object more closely. For example, we can look at our People table with `\d People`:

```
linux=# \d People
                     Table "public.people"
Column        |      Type                     |  Modifiers
--------------+-------------------------------+----------
id            | integer                       | not null default
nextval('people_id_seq'::regclass)
first_name    | text                          | not null
last_name     | text                          | not null
email_address | text                          | not null
added_at      | timestamp without time zone | not null default now()
Indexes:
"people_pkey" PRIMARY KEY, btree (id)
"people_email_address_key" UNIQUE, btree (email_address)
```

There are several things to notice in the above output:

- First, PostgreSQL sees the table as public.people, not just people. This is because every object must exist inside of a schema, or namespace, and the default schema is called public. We can use schemas to partition the namespace within a particular database or to handle partitions. This means we don't need to split data across two more databases just to deal with conflicting permissions and names.
- The table name, as well as all column names, are displayed in lowercase letters. That's because PostgreSQL tries to adhere to the SQL standard as best as possible, and the standard says that identifiers should be case-insensitive. If you really want case-sensitive names (and you probably don't), use double quotes around the identifiers.
- Our id column has been transformed, as expected, into an integer column with a default value taken from a sequence.

Constraints

There are some problems with our table definition. Although we have effectively stopped people from storing NULL values in our TEXT columns, we haven't done anything to stop them from entering empty strings. In addition, we might want to ensure that the email_address column looks at least something like an e-mail address.

We can do this by adding constraints to our columns—tiny functions that check the value being inserted or updated. If the new value doesn't fit the definition, PostgreSQL refuses to allow its insertion. Here's a new definition of our table, with some constraints defined:

```
CREATE TABLE People (
  id     SERIAL    NOT NULL,
  first_name     TEXT     NOT NULL  CHECK (first_name <> ''),
  last_name     TEXT     NOT NULL CHECK (last_name <> ''),
  email_address     TEXT NOT NULL CHECK (email_address ~* '.@.+\\\.'),
  added_at     TIMESTAMP  NOT NULL  DEFAULT NOW(),

  PRIMARY KEY(id),
  UNIQUE(email_address)
);
```

If we inspect our table definition, it has changed somewhat, to include the constraints:

```
linux=# \d people
                   Table "public.people"
    Column    |              Type              |      Modifiers
--------------+-----------------------------+----------------------
id            | integer                     | not null default
nextval('people_id_seq'::regclass)
first_name    | text                        | not null
last_name     | text                        | not null
email_address | text                        | not null
added_at      | timestamp without time zone | not null default now()
  Indexes:
  "people_pkey" PRIMARY KEY, btree (id)
  "people_email_address_key" UNIQUE, btree (email_address)
  Check constraints:
  "people_email_address_check" CHECK (email_address ~* '.@.+\\.'::text)
  "people_first_name_check" CHECK (first_name <> ''::text)
  "people_last_name_check" CHECK (last_name <> ''::text)
```

Let's see what happens if we violate these constraints:

```
linux=# insert into people (first_name , last_name, email_address)
    values ('', 'Lerner', 'reuven@lerner.co.il');
ERROR:  new row for relation "people" violates check constraint
    "people_first_name_check"

linux=# insert into people (first_name , last_name, email_address)
    values ('Reuven2', 'Lerner2', 'reuven');
ERROR:  new row for relation "people" violates check constraint
    "people_email_address_check"
```

Sure enough, our constraints help to ensure that our database is in order.

The most common type of constraint is a foreign key, in which one table points to another. For example:

```
CREATE TABLE Appointments (
    person_id     INTEGER      NOT NULL  REFERENCES People,
    starting_time     TIMESTAMP NOT NULL,
    duration            INTERVAL NOT NULL,
    notes TEXT NULL
);
```

If we try to create an appointment that refers to a non-existent person, we will be rejected:

```
INSERT INTO Appointments (person_id, starting_time, duration, notes)
    VALUES (5000, '2007-Feb-12 13:00', interval '1 hour', 'Lunch');

ERROR: insert or update on table "appointments" violates foreign key
constraint "appointments_person_id_fkey"
DETAIL: Key (person_id)=(5000) is not present in table "people".
```

Foreign-key constraints help in the other direction as well. If you try to drop a row to which a foreign key points, PostgreSQL will refuse the request, indicating that you must first delete the foreign key. You can adjust the rules for these constraints by setting the ON UPDATE or ON DELETE modifiers to the foreign key definition.

Other Features

This list of features is just the tip of the iceberg. And that's part of the magic of PostgreSQL—out of the box, it's straightforward and easy to use, but you almost always can redefine and extend existing functionality with your own code and data. The built-in operators, along with the flexible ways in which they can be combined and further enhanced with your own functions and definitions, make for a powerful combination. I don't often use unions or intersections, but I do often use views.

For example, one of my favorite features is the ability to use subselects just about anywhere you would have a value. If you have someone's e-mail address, you can use that to INSERT a row into Appointments in a single command:

```
INSERT INTO Appointments (person_id, starting_time, duration, notes)
    VALUES ((SELECT id
                FROM People
              WHERE email_address = 'reuven@lerner.co.il'),
            '2007-Feb-12 13:00', interval '1 hour', 'Lunch');
```

If the existing data types aren't enough, we can construct our own. PostgreSQL already comes with a number of existing data types, including geometric shapes, IP addresses and even ISBNs.

If we want to create more than one table with similar characteristics, we can take advantage of PostgreSQL's object-oriented features. Thus, we could have a People table and a Managers table, in which the definition of Manager inherits the characteristics of People and adds its own extensions.

You also can create your own server-side functions, in a variety of different languages—from PostgreSQL's own Pl/pgsql to specialized versions of Perl, Python, Tcl, Java, Ruby and the R statistical language. These functions can return individual values or entire tables, and can be used in triggers. You also can use these functions to rewrite the rules for inserting, updating and deleting data from a table or even a view.

But, perhaps the most important feature of all is the built-in support for transactions. Transactions are an essential part of database programming, in that they allow us to combine multiple queries into one all-or-nothing action. The classic example of a transaction is the movement of money from one bank account to another; if the power goes out, you want to be sure that the money was moved, or that it wasn't. It would be unacceptable for the money to disappear altogether or for it to appear in both accounts when the lights come back on.

Recent versions of PostgreSQL have enhanced its transactional capabilities. Not only can you commit or roll back a transaction, but you also can define savepoints inside a transaction. If something goes wrong, you can either roll back the entire transaction or merely go to the previous savepoint. Moreover, PostgreSQL now supports two-phase commits, making it possible to synchronize distributed processes that require communication and coordination.

If anything goes wrong, PostgreSQL also provides a PITR (point-in-time recovery) through a write-ahead log (WAL), ensuring that even if the power is cut off at the most critical moment, transactions will be committed or rolled back, and that as many transactions as possible will be committed.

You might have noticed that I haven't mentioned locking at all. That's because, for the most part, PostgreSQL users don't have to worry about locking. The lack of locking is handled using a system known as MVCC (multiversion concurrency control), which has only one drawback, namely the creation of many unused and cast-off database rows. The traditional way to handle this in PostgreSQL is to VACUUM the database regularly, removing old rows and clearing up space. Recent versions now include an auto-vacuum agent, reducing or even eliminating the need to VACUUM on a regular basis.

Finally, recent versions of PostgreSQL include support for tablespaces. This means you can spread tables across different directories and filesystems, rather than keep everything under the directory defined by your installation. This can boost performance or reliability significantly, particularly on large databases.

Conclusion

Don't think of PostgreSQL as a powerful open-source database. Rather, think of it as a powerful database that happens to be released under an open-source license. It has a wealth of features that make it scalable for large systems and needs, but it is easily approachable by novices who want to begin their journey into the world of relational databases.■

Reuven M. Lerner, a longtime Web/database consultant, is a PhD candidate in Learning Sciences at Northwestern University in Evanston, Illinois. He currently lives with his wife and three children in Skokie, Illinois. You can read his Weblog at altneuland.lerner.co.il.

Resources

The main Web site for PostgreSQL is **www.postgresql.org**. This site contains links to software, documentation, FAQs and a host of mailing lists.

My favorite book about PostgreSQL is simply called *PostgreSQL*, 2nd ed., written by Korry Douglas, and published by Sams (ISBN 0672327562).

The PostgreSQL community mailing lists are also invaluable sources of help and information. It's not unusual for one of the core developers to answer a question that someone has posed or admit that there is a bug that needs fixing.

SUPERMICRO®

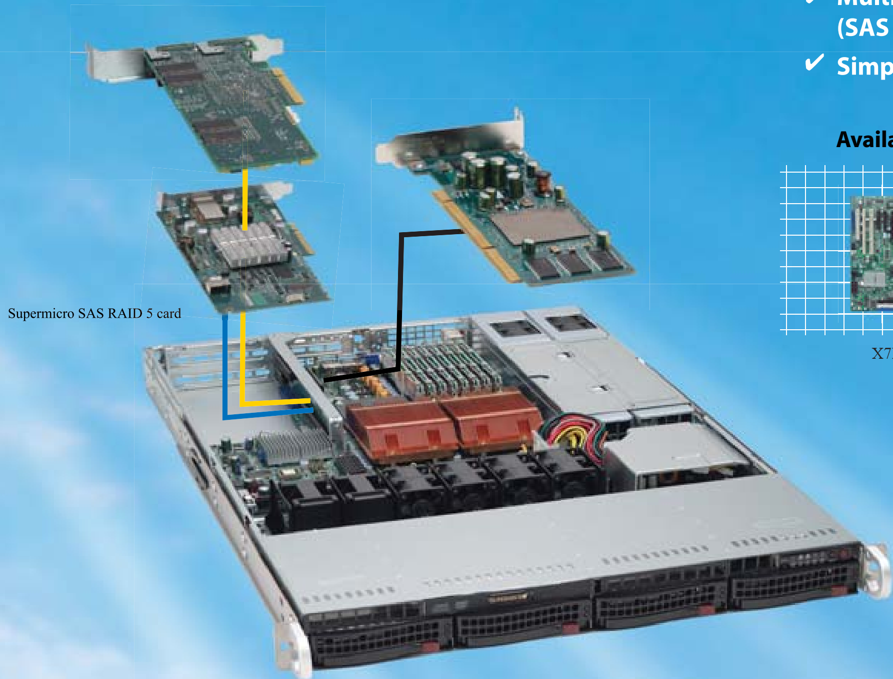
UIO

Universal I/O

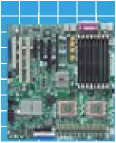
Maximized I/O Expandability & Flexibility

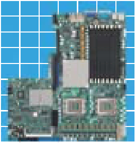
The Freedom to Build Exactly What You Need.

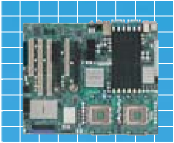
- ✔ 3/7 Add-on Cards for 1U/2U
- ✔ Highly Upgradeable
- ✔ High Efficiency Power (up to 90%+)
- ✔ Multiple Expansion Card Options (SAS RAID 5, 10-G, IB...up to 20 choices)
- ✔ Simplify Your Inventory Management

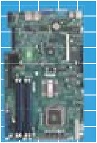


Available Motherboard Options


X7DBN


X7DBU


X7DAL-E+/X7DAL-E


PDSMU

Optimized Chassis Solutions

SC815TQ-R650U

SC825TQ-R700U

Supermicro UIO servers allow users to select from a wide range of I/O options to provide the ultimate in storage and networking flexibility. The UIO card becomes a part of the serverboard, allowing the system to retain all of it's PCI Express and PCI X slots for expansion cards. As a result, future upgrades can be acheived by replacing the UIO card and/or expansion cards instead of replacing the entire system. This versatility helps to minimize the amount of different servers that customers need to operate their business.

intel® Xeon® inside™ ™

Quad-core. Unmatched.

For more information visit us at www.supermicro.com

* Our new generation power supply efficiency measures 90%+ or more under a typical loading operation.

AMAX	Arrow Electronics	ASI	Bell Micro	Ingram Micro	MA LABS	Synnex	Tech Data
1-800-800-6328	1-888-427-2250	1-800-2000-ASI	1-800-232-9920	1-800-456-8000	1-408-941-0808	1-800-756-5974	1-800-237-8931
www.amax.com	www.arrownacp.com	www.asipartner.com	www.bellmicro.com	www.ingrammicro.com	www.malabs.com	www.synnex.com	www.techdata.com

Features

Ajax Simplified PAGE 55 | **Writing Web Applications with Web Services and Ajax** PAGE 58
MochiKit by Example PAGE 64 | **Four Cool Ajax Plugins for WordPress** PAGE 70

Ajax Simplified

NICHOLAS PETRELEY

Ajax can become complex as far as implementation, but the concept is quite simple.

THIS IS A simple tutorial on Ajax that I hope will ease the fears of those of you who think Ajax can be intimidating. Despite the meaning of the term (Asynchronous JavaScript and XML), Ajax really revolves around a very simple principle. It lets you manipulate the contents of a Web page without having to reload the page. Here are the key steps involved that exploit the power of Ajax:

- Capture an event (such as when a user changes an edit field or presses a button).
- The event triggers JavaScript code, which sends a query to the Web server.
- The JavaScript code retrieves results from the server.
- The JavaScript code uses the results to change the contents of the Web page.

JavaScript accesses the Document Object Model (DOM) to change the contents of a Web page without reloading the Web page. For example, suppose your Web page contains the following element:

```
Total: <input type="text" id="total" />
```

The id portion of the HTML tag creates an element called total in the DOM, the contents of which you can change via JavaScript with the following JavaScript code:

```
document.getElementById('total').value = <some value>;
```

Web designers have been using this capability for a long time. The real power in Ajax comes from the ability to calculate the value for the total at the server side rather than at the client. To keep it simple, here's an example that doesn't really involve any server activity other than returning a result. This example presents a simple form that lets you type in a zip code. When you change the value of the zip-code field, the JavaScript code executes a PHP script at the server side that returns the shipping cost to that zip code. The JavaScript code then modifies the totalshipping field to reflect the server response.

The example page shown in Listing 1 contains only the most basic elements of an Ajax page—the primary functions being getHTTPObject, handleHttpResponse and updateShipping. The onChange event is what triggers the JavaScript function updateShipping. You could use onBlur instead, which would call updateShipping when you simply leave the zip-code field and it loses focus.

The getHTTPObject function is what allows you to make a page request via JavaScript, and the updateShipping function performs the page request. The handleHttpResponse function receives the input from the page request and extracts the information in order to modify an element in the page (in this case, the totalshipping field). These are the three basic functions you need to perform an Ajax operation.

This first example avoids XML entirely. The following line of code grabs the result as plain text:

```
results = http.responseText;
```

If you try out this code, you'll find that when you type a zip code (or virtually anything, because the code does no error checking) and then leave the field, the JavaScript automatically retrieves the value $5.00 from the PHP application and places the value in the Total Shipping field.

Keep in mind that the above example takes as many shortcuts as possible to keep it simple. There is no error checking or error handling whatsoever. There aren't even any HTML tag names, only ids. For example, it would be more common to create an input field that reads <input type="text" name="totalshipping" id="totalshipping" />. You probably wouldn't place the shipping cost in a field that a person could edit (although your form could re-validate the shipping when the person clicked "purchase" to correct any user changes). In addition, the example doesn't actually calculate a shipping cost. The URL in the above code points to a simple PHP script that returns the text value "$5.00" (Listing 2). A real application would take the zip code and use it to calculate the shipping cost and return that value. In short, the example cuts every possible corner to isolate how Ajax works rather than how one should code an Ajax application.

Enter XML

Technically, you can create a full Ajax application without ever using XML, but you will find XML to be a virtual necessity as your Web application grows in complexity. Here is how to do the same simple Web page with XML, once again cutting every corner for the sake of simplicity.

Notice in Listing 3 that we now grab the response with the code http.responseXML and extract the value we want with the code xmlDocument.getElementsByTagName('shipping'). Note also that the XML refers to the total with the tag shipping instead of totalshipping. This difference is unnecessary, but the purpose in this tutorial is to avoid the possible implication that the XML tag name and the HTML

Listing 1. index.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >

<head> <title>Example</title>

<script language="javascript" type="text/javascript">

var url = "getShipping.php?zipcode=";

function handleHttpResponse() {
  if (http.readyState == 4) {
    results = http.responseText;
    document.getElementById('totalshipping').value = results;
  }
}

function updateShipping() {
  var zipValue = document.getElementById("zip").value;

  http.open("GET", url + escape(zipValue), true);
  http.onreadystatechange = handleHttpResponse;
  http.send(null);
}

function getHTTPObject() {
  var xmlhttp;
  xmlhttp = new XMLHttpRequest();
  return xmlhttp;
}

var http = getHTTPObject();

</script>

</head>

<body>
<form action="post">
<p>
ZIP code: <input type="text" size="5" id="zip"
 onChange="updateShipping();" />
</p>
Total Shipping: <input type="text" id="totalshipping" />
</form>
</body>
</html>
```

Listing 2. getShipping.php

```
<?php
echo "$5.00";
?>
```

Listing 3. index-xml.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >

<head> <title>Example</title>

<script language="javascript" type="text/javascript">

var url = "getShippingXML.php?zipcode=";

function handleHttpResponse() {
  if (http.readyState == 4) {
    var xmlDocument = http.responseXML;
    var shipping = xmlDocument.getElementsByTagName('shipping')
➥.item(0).firstChild.data;
    document.getElementById('totalshipping').value = shipping;
  }
}

function updateShipping() {
  var zipValue = document.getElementById("zip").value;

  http.open("GET", url + escape(zipValue), true);
  http.onreadystatechange = handleHttpResponse;
  http.send(null);
}

function getHTTPObject() {
  var xmlhttp;
  xmlhttp = new XMLHttpRequest();
  return xmlhttp;
}

var http = getHTTPObject();

</script>

</head>

<body>
<form action="post">
<p>
ZIP code: <input type="text" size="5" name="zip" id="zip"
onChange="updateShipping();" />
</p>
Total Shipping: <input type="text" id="totalshipping" />
</form>
</body>
</html>
```

input field id must match in order to make the application work. They do not have to match.

The only thing left is to modify our PHP code to return XML instead of plain text. See Listing 4 for the PHP code. In addition to the XML content itself, note the line of code that sends a header identifying the content as XML before returning the XML content itself. The XML places the shipping amount as a child of <order>, along with the unused data, <total>. This is simply a baby step toward representing a more realistic set of data that the page should return.

Believe it or not, that's all there is to Ajax. Just about everything else that adds complexity to Ajax application development falls into the following categories.

Listing 4. GetShippingXML.php

```
<?php
$shipping="$5.00";
$total="$505.00";
$return_value = '<?xml version="1.0" standalone="yes"?>
<order>
 <shipping>'.$shipping.'</shipping>
 <total>'.$total.'</total>
</order>';
header('Content-Type: text/xml');
echo $return_value;
?>
```

Validation and Error Handling

A real Ajax application would not assume that the PHP file exists. It also would check the validity of the zip code before attempting to send it as a parameter to the server in order to find the shipping cost. (You also could have the server validate the zip code or do minimal validation at the client side, such as ensuring that the user entered five full digits and then perform full validation of the zip code at the server side.)

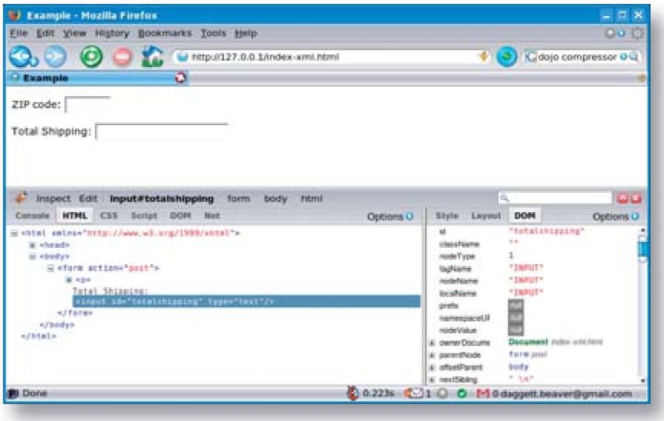
The above example eschews all error handling in order to keep the focus on the bare bones of how Ajax works. Obviously, you need to include input validation, error detection and error handling in a real application.

Accounting for the Differences between Browsers

The above sample code works with Firefox, but there's no guarantee it will work in any other browser. If you want to write all your Ajax code from scratch, taking into account the variations between Firefox, IE and Opera, buy lots of ibuprofen—you'll need it. Fortunately, a plethora of Ajax libraries exist that manage the differences for you. One of my favorites is Dojo (see Resources).

Managing the Elements of the Document via the DOM

Ajax relies on the DOM to address the various elements within a page. As your page becomes more complex, it gets harder to keep track of all the elements, their names and ids. Firefox has a built-in DOM inspector that is enormously helpful. If that's not enough, you can install the Firebug add-on to Firefox. Firebug not only provides you with a way to examine the DOM, it also helps you debug your JavaScript code and manage your cascading stylesheets (see Resources for a link to the add-on). Figure 1 shows the XML example page as viewed through Firebug. [Reuven Lerner covers Firebug in this month's At the Forge on page 22.]


Example - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://127.0.0.1/index-xml.html
dojo compressor
Example
Inspect Edit input#totalshipping form body html
Console HTML CSS Script DOM Net Options
Style Layout DOM Options
Done
0.223s
0 daggett.beaver@gmail.com

ZIP code:

Total Shipping:

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
  <body>
    <form action="post">
      <p>
        Total Shipping:
        <input id="totalshipping" type="text"/>
      </form>
    </body>
</html>
```

```
id              "totalshipping"
className       ""
nodeType        1
tagName         "INPUT"
nodeName        "INPUT"
localName       "INPUT"
prefix
namespaceURI
nodeValue
ownerDocument   Document index-xml.html
parentNode      form post
offsetParent    body
nextSibling     "\n"
```

Figure 1. View elements of the DOM with Firebug.

Performing the Server-Side Calculations and Operations

As for what you must do to handle the services at the server side, that's entirely up to your choice of Web application language and choice of database, among other things. Use what you know best, or take the time to learn other Web application languages you suspect will ease the burden of writing server-side code.

Optimizing Your Application Performance

JavaScript code optimization is an art, but it always helps to compress your JavaScript code. For example, indent all your code for readability, but when you're finished, the tabs and spaces are simply more bytes users will have to download. You can squeeze your JavaScript down to fewer bytes with one of many JavaScript compressors. The Dojo library is compressed for you, and Dojo provides a compressor you can use on your own code. You even can compress your code on-line via Dojo Shrinksafe (see Resources).

Finally, keep an eye on what you manage at the server side and what you manage at the client side. Depending on what your Ajax Web application does, you may find some performance gains by storing certain information in cookies, or you may speed up performance by storing the information at the server side. Use common sense and experiment between the two approaches to see which performs best.

It's not always easy to build a killer Ajax application, but hopefully this tutorial on the simplicity of how Ajax works will encourage you to give it a try. Now grab a toolkit and go!■

Nicholas Petreley is Editor in Chief of *Linux Journal* and a former programmer, teacher, analyst and consultant who has been working with and writing about Linux for more than ten years.

Resources

Dojo: **dojotoolkit.org**

Dojo JavaScript Compressor: **dojotoolkit.org/docs/compressor_system.html**

Dojo Shrinksafe: **alex.dojotoolkit.org/shrinksafe**

Firebug: **https://addons.mozilla.org/firefox/1843**

Writing Web Applications with Web Services and Ajax

MIKE DIEHL

An Ajax primer with Perl and PostgreSQL.

IF YOU'VE DONE any Web development at all recently, you've no doubt heard the buzz going on about Web Services and Ajax. The industry hype is so prevalent that you'd almost think people were talking about the next Microsoft operating system. Fortunately, they're not. Web Services and Ajax are two Web technologies that allow developers to create more interesting Web applications as well make the development easier and less error-prone.

Now that I've added to the hype, let me take some time to outline what we mean when we say "Web Services and Ajax".

A Web Service is a program that is accessible over the Internet and provides a specific service, such as searching a library's collection or getting bid history from eBay. We're not talking about a full-fledged application, but rather a Web-based Application Programming Interface (API) that can can be called over the Internet by a given program to perform a needed function. Often, the results of the call to a given Web Service are returned in XML format, which the calling program can manipulate easily.

When people discuss Web Services, they often mention things like JSON, REST, SOAP or XML-RPC. These are simply a few of the protocols available for calling a Web Service. Being familiar with these protocols lets you make use of some of the really powerful Web Services being provided by the likes of Amazon, Google and eBay. However, for my personal Web development, I've found these protocols to be a bit heavy.

Ajax is a mechanism that allows a Web page to make calls back to the server without refreshing the page or using hidden frames. For example, if a user changes the value of a form field, the Web page could tell the server to make a change to a database, without having to refresh the Web page, as would be needed in the case of a standard CGI script. From the user's perspective, the update just happens.

In this article, I outline a set of very primitive Web Services that perform specific functions on a database. The calls to the Web Services will be done via Ajax. Essentially, we're going to build a simple contact management program that stores a person's first name, last name and phone number. We'll be able to move up and down through the database, make additions and corrections and delete records. The neat thing is that once the page is initially loaded, we won't have to refresh it again, even when we make changes.

Listing 1. Preparing a PostgreSQL Sequence and Table for the Project

```
create sequence contacts_id_seq;

create table contacts (
   id      integer default nextval('contacts_id_seq') not null,
   first   varchar(20),
   last    varchar(20),
   phone   varchar(20)
);
```

Before we can get started though, we need to have a table in a database in which to store the information. I happen to use PostgreSQL as my preferred DBMS. For our simple application, we need only one table (Listing 1).

The snippet of SQL in Listing 1 creates a sequence and a table. The table structure is pretty straightforward for our simple application. The only thing worth mentioning is the id field. By default, when records are inserted into our contacts table, the value of the id field is set to the next number in the contacts_id_seq sequence. The end result is that each of our contact records has a unique integer number that can be used to locate it.

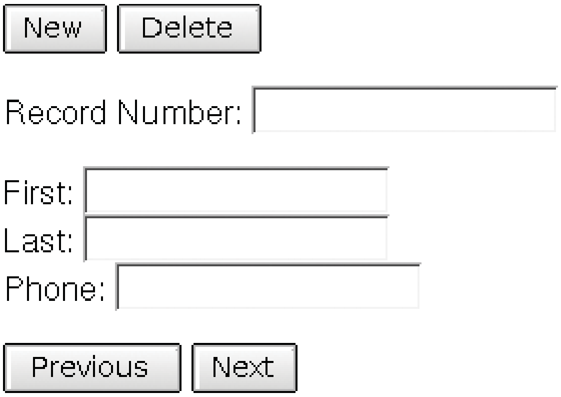


Figure 1. The No-Frills Web Page for This Sample Application

Listing 2. The Basic HTML for the Application

```
<html>
<head>
   <title>Contact Application</title>
   <script src=http://contacts.js></script>
</head>
<body>
<form method=POST name=main>
<input type=button name=new value="New"
     onclick="insert_record();">
<input type=button name=delete value="Delete"
     onclick="delete_record(main.id.value);">
<p>
Record Number: <input id=id name=id>
<p>
First: <input id=first name=first
   onChange="update_record(main.id.value,
   'first', main.first.value);">
<br>
Last: <input id=last name=last
   onChange="update_record(main.id.value,
   'last', main.last.value);">
<br>
Phone: <input id=phone name=phone
   onChange="update_record(main.id.value,
   'phone', main.phone.value);">
<p>
<input type=button name=previous value="Previous"
   onClick="select_record(main.id.value-1);">
<input type=button name=next value="Next"
   onClick="select_record(Number(main.id.value) + 1);">
</form>
</body>
</html>
```

Now that we have the database table defined, we can start to flesh out the actual application. Listing 2 shows the HTML for our simple application, and Figure 1 shows what the application looks like in a Web browser.

As you can see, our simple application is just that, simple. I've stripped it down to the bare necessities to make our discussion easier.

Figure 1 shows how our application allows us to insert a new contact record or delete the current record by pressing the buttons at the top. At the bottom of the application, we can move to the previous or next record in the database. Of course, we have input fields to hold the first and last name as well as the phone number. We also have a form field to display the record id number. In a real application, I'd probably make this a hidden field, but for the sake of instruction, I've left it visible.

Referring back to Listing 1, you can see that the page is fairly straightforward. Aside from importing the contacts.js JavaScript, the first part of the page is standard boilerplate. Things get interesting when we get to the form buttons and fields.

Let's look at the "New" button:

```
<input type=button name=new value="New"
     onclick="insert_record();">
```

The neat thing is that once the page is initially loaded, we won't have to refresh it again, even when we make changes.

This button simply calls a JavaScript function called insert_record() any time a user presses the button. The Delete, Previous and Next buttons all work similarly. The magic is in the JavaScript. Let's look at the JavaScript first (Listing 3).

The insert_record() JavaScript function, which is called when a user presses the New button, is the simplest of the JavaScript functions. All insert_record() does is use the send_transaction() function to call the insert.pl Web Service. In fact, the insert_record(), delete_record(), select_record() and update_record() functions are all wrappers for send_transaction().

The send_transaction() function is where the Ajax comes into our application. This function takes the URL of the service that needs to be called as well as any parameters that need to be passed to the service via HTTP's GET method. Then, the function creates an object that allows the service to be called. We have to jump through a small hoop, because Microsoft chose to call this object ActiveXObject while

Listing 3. JavaScript code handles the database actions and data processing.

```
var req;

function insert_record () {
    send_transaction("/cgi-bin/insert.pl");
    return 1;
}

function select_record (i) {
    send_transaction("/cgi-bin/select.pl?id=" + i);
    return 1;
}

function delete_record (i) {
    send_transaction("/cgi-bin/delete.pl?id=" + i);

    var id = document.getElementById("id");
    select_record(id);
    return;
}

function update_record (i, field, value) {
    send_transaction("/cgi-bin/update.pl?id=" + i +
    "&field=" + field + "&value=" + value);
    return 1;
}

function send_transaction (i) {

    if (window.XMLHttpRequest) {
        req = new XMLHttpRequest();
    } else if (window.ActiveXObject) {
        req = new ActiveXObject("Microsoft.XMLHTTP");
    }

    if (req) {
        req.onreadystatechange = process_results;
        req.open("GET", i, true);
        req.send(null);
    }
}

function process_results () {
    var name = "";
    var value = "";
    var fields;
    var i;
    var length;

    if (req.readyState < 4) { return 1; } // transaction
                                             // not done, yet

    var xml = req.responseXML;
    var result = xml.getElementsByTagName("result").item(0);

    fields = result.getElementsByTagName("field");
    length = fields.length;

    for (i=0; i<length; i++) {
        var field = fields[i];

        name = field.getAttribute("name");
        value = field.getAttribute("value");


        var form_field = document.getElementById(name);
        form_field.value = value;
    }

    return 1;
}
```

almost every other browser calls it XMLHttpRequest. Once the object is created, by whatever name, we tell the object to call our Web Service and then call our process_results() function when the call has returned its results. This is done in the line that assigns the function name to the object's onreadystatechange property.

Well, I lied a little bit. It turns out that the browser will call our process_results() function up to four times at various stages during the service request. Each time the function is called, the value of the readyState property is changed to reflect what phase of the transaction is occurring. Unfortunately, there doesn't seem to be much agreement on when the function is called. The only thing that all browsers seem to agree on is that when the transaction is complete, the readyState property is set to 4. Checking for this value is the first thing our process_results() function does. If the transaction isn't complete, we simply return quietly.

The Web Services simply become bricks that are glued together with JavaScript to build applications, and this is what makes using Web Services such an elegant method of Web development.

Once the transaction is complete, we can recover the resulting XML from the request object's responseXML property. Once we have the XML, we loop over each field element, making a note of both the field name and value. Then we find the corresponding field in the HTML document and assign the new value to it. So by sending the appropriate XML, the Web Services can arrange for any, or all, of the Web form fields to be updated.

If you think the JavaScript was easy to follow, wait until you see the Perl scripts that implement the Web Services; they're even easier to understand and debug. The insert.pl

Listing 4. The server-side Perl script handles the database insert action.

```
#!/usr/bin/perl

use DBI;

$dbh = DBI->connect("dbi:Pg:dbname=database",
 ➥"postgres", "password");

$dbh->do("insert into contacts (first,last,phone) values
(NULL,NULL,NULL)");

$sth = $dbh->prepare("select last_value from
 ➥contacts_id_seq");
$sth->execute();
($index) = $sth->fetchrow_array();

print "Content-type: text/xml\n\n\n";

print "<result>\n";
print "<field name=\"id\" value=\"$index\"></field>\n";
print "</result>\n";
```

Listing 5. Resulting XML

```
<result>
<field name="id" value="25"></field>
</result>
```

program is shown in Listing 4.

All this program does is connect to a database, insert an empty record into the contacts table, retrieve the id of the newly created record and return the results in a block of XML with a text/xml MIME type. The resulting XML resembles that shown in Listing 5.

The select.pl, delete.pl and update.pl services are very similar, as shown in Listings 6, 7 and 8, respectively.

The select.pl service shown in Listing 6 takes a single parameter—the id number of the record to be retrieved. The result is an XML file containing all the fields in the record and the appropriate values. This allows us to call the function with a record id and retrieve all the fields of that record for later manipulation.

The delete.pl service shown in Listing 7 takes a record id and deletes the record with that id. Then, the program finds the next lowest record number and returns that record id.

Finally, the update.pl service shown in Listing 8 takes a record id, a field name and a new value as parameters. Then, the program updates the given field of the selected record with the new value. The new field value is then returned via XML.

Granted, our little application is fairly trivial, but it does perform all of the basic functions that need to be performed on a database: insert, delete, update and search. More important, no single element of this application is difficult to write, debug or understand. In fact, with a few improvements that I outline next, the Web Service scripts and much of the JavaScript can be reused for other parts of a larger application or even many different applications. The Web Services simply become bricks that are glued together with JavaScript to build applications, and this is what makes using Web Services such an elegant method of Web development.

From the user's perspective, using Ajax to perform the database functions is a major win. As mentioned before, once the application is loaded, users never have to incur the cost of re-downloading it and having their browsers re-render it. On more complex pages, this can be a significant delay. Furthermore, because the results of a given operation are returned in small snippets of XML, the bandwidth requirements are minimal. It's arguable that not only would users perceive this type of application as faster, but it also would put lower demands on the server and network infrastructure that provide the application.

But, how hard would it be to add a new field, perhaps an e-mail address, to our application? Well, we'd have to add an appropriate field to our database table scheme. Then, we'd have to add the field,

Listing 6. The Perl Script for Selecting Data

```
#!/usr/bin/perl

use CGI;
use DBI;

$dbh = DBI->connect("dbi:Pg:dbname=database",
 ➥"postgres", "password");

print "Content-type: text/xml\n\n\n";
print "<result>\n";

$cgi = new CGI;

$id = $cgi->param("id");

$sth = $dbh->prepare("select * from contacts where id=$id");
$sth->execute();

$a = $sth->fetchrow_hashref();

foreach $key (keys %$a) {
   print "<field name=\"$key\" value=\"$a-
>{$key}\"></field>\n";
}

print "</result>\n";
```

Listing 7. The Perl Script for Deleting a Record

```
#!/usr/bin/perl

use CGI;
use DBI;

$dbh = DBI->connect("dbi:Pg:dbname=database",
 ➥"postgres", "password");

$cgi = new CGI;

$id = $cgi->param("id");

$dbh->do("delete from contacts where id=$id");

$sth = $dbh->prepare("select max(id) from contacts where
id<$id");
$sth->execute();
($index) = $sth->fetchrow_array();

print "Content-type: text/xml\n\n\n";

print "<result>\n";
print "<field name=\"id\" value=\"$index\"></field>\n";
print "</result>\n";
```

Listing 8. The Perl Script for Updating a Record

```
#!/usr/bin/perl

use CGI;
use DBI;

$dbh = DBI->connect("dbi:Pg:dbname=database",
 ➥"postgres", "password");

$cgi = new CGI;

$id = $cgi->param("id");
$field = $cgi->param("field");
$value = $cgi->param("value");

$dbh->do("update contacts set $field=\'$value\' where
id=$id");

print "Content-type: text/xml\n\n\n";
print "<result>\n";
print "<field name=\"$field\" value=\"$value\"></field>\n";
print "</result>\n";
```

with the same name, to our HTML document. We could use the other form fields as a template of course. And, that should just about do it.

So, how could we improve our code? First, we'd need to take care of some glaring security issues. Our Web Services should use some form of authentication to make sure that only authorized users can perform database functions. More subtly though, the Web Services need to perform some basic validation on the parameters they receive. The delete.pl service accepts a record number in the form of id=25 as a parameter. What if someone wanted to be mean and, instead, sent id=25 or 1=1 to our service? Well, our database would be gone because 1=1 is always true, and our program would delete *all* records. So, we would have to take care of such issues before we could use these services in the wild.

You may have noticed that all of the fields in our database are of type varchar(20). That's not very flexible or efficient. To be truly useful, our services would need to be able to query the database to determine what data type a given field was and act appropriately. For example, chars and varchars need to be quoted, but integers and booleans do not. The service should be able to determine how to handle these situations.

Finally, by simply sending the name of the table as one of the parameters, we can build a Web Service that can modify database tables other than our contacts table. We'd be able to use the same services to update a shopping list, inventory or calendar. Generalizing our Web Services like this would make our simple contacts application easy to write as well as any other application in which we chose to use them.

So, by coupling Ajax with our own brand of Web Services, we're able to write applications that are more responsive to user input, less taxing on the server infrastructure, and much easier to write and maintain.■

Mike Diehl is a contractor at Sandia National Laboratories in Albuquerque, New Mexico, where he writes network management software. Mike lives with his wife and two small boys and can be reached via e-mail at mdiehl@diehlnet.com.

OSBC | OPEN SOURCE BUSINESS CONFERENCE

MAY 22-23, 2007
PALACE HOTEL I SAN FRANCISCO, CA
OSBC.com

SAVE $200
off current pricing when you register and use priority code
OSBC57LJ
at OSBC.com

Open Source Business Conference (OSBC) is the industry s only forum for senior business leaders, IT professionals, lawyers and venture capitalists to collaborate on emerging business models, strategies and profitability for open source software use. OSBC explores the legal, investment, vendor and customer risks, rewards and opportunities presented by open source software.

TRACKS AND TOPICS INCLUDE:

What s Next

Open source business models (cutting edge models and details on how the leading companies do business), emerging issues (Attribution, effective download conversion strategies, etc.), and other topics of interest to the rising tide of open source vendors.

What s Now

Examine CIO issues – implementation strategies, cost savings, etc., as well as ways to work with open source communities and get quality support.

What s Legal

How to take a proprietary product into the open, the mechanics of setting up an open source review board for enterprises, what GPLv3 means for the industry, and more (CLE credentials).

FEATURED SPEAKERS:

Marc West
CIO, H&R Block

Tim Golden
SVP Linux Engineering, Bank of America

Marten Mickos
CEO, MySQL

Eben Moglen
General Counsel of the Freedom Law Center and Professor, Columbia Law School

Matthew Szulik
CEO, Red Hat

Jon Stumpf
CTO, AIG

PLATINUM SPONSORS:
ORACLE
SUGARCRM COMMERCIAL OPEN SOURCE

GOLD SPONSORS:
EMC² where information lives
intel
Jefferies
GOODWIN PROCTER
TROLLTECH
UNISYS imagine it. done.

PRODUCED BY:
InfoWorld

View complete program details and register today at www.OSBC.com

RYAN WILCOX

mochikit by Example

An overview of MochiKit with three real-life examples.

INTERACTIVE WEB PAGES, long dreamed of by designers, are finally here—Web interfaces that respond instantly to user commands, with minimal page redraw. All this and more is possible via Ajax (Asynchronous JavaScript and XML), which recently has come into vogue with the Web illuminati.

In JavaScript, and programming in general, the best code is code you don't have to write. For serious projects, this often means using a framework—a collection of useful and reusable code that is tested, optimized and (ideally) peer-reviewed. The better frameworks have automated unit tests to make certain they keep working. Having a good, high-level JavaScript framework, for example, means more time to push the boundaries and less time writing boring building-block code (and then reworking it when the inevitable cross-browser incompatibilities poke their heads up).

MochiKit (**www.mochikit.com**) is a JavaScript framework that provides tools for dealing with asynchronous requests (Ajax), DOM (Document Object Model) functionality, functional programming tools, dates and times, string formatting, colors, visual effects, events, drag-and-drop ability, sorting and many other features. MochiKit 1.3 is targeted to work on Safari 2.0.2, Firefox (1.0.7, 1.5 and 2.0), Opera 8.5 and Internet Explorer (6 and 7).

This article provides a quick introduction to MochiKit, explains how to get started with MochiKit (with interesting stops along the way) and describes three walk-through examples of varying complexity that also are generic enough to use in your Web applications right now. Ever wanted to round element corners in HTML? Make a link that is clickable only once? Create a dynamic login mechanism? Keep reading!

What's in MochiKit?

Included in MochiKit are algorithms for data structures (including serialization), functional programming, iteration, DOM and CSS manipulation, asynchronous server communication, a signal/slot mechanism for JavaScript events and logging tools. At this point, MochiKit sounds like the C++ STL of JavaScript. Above and beyond what the STL provides C++, MochiKit also provides event handling, drag and drop, colors and visual effects.

On the topic of data structure algorithms, MochiKit provides the powerful iteration tools of filter(), which returns only list elements that match a criteria; find(); map(), which returns the result of list elements run through an operation; and more. MochiKit also provides tools to translate to and from JSON (JavaScript Object Notation) syntax: serializeJSON() and evalJSON(). Above all, MochiKit gives you the power to hook your own objects into MochiKit's magic.

MochiKit's tools for functional programming allow functions to be created dynamically, or they may simply provide more extensive (or less broken) behaviors for functionality provided already in JavaScript. MochiKit's partial() and bind() functions create a version of a function that requires less parameters or rebind JavaScript's this parameter, respectively. In a nutshell, these tools let you create functions dynamically. These two different functions aren't obviously useful right now, but they are powerful when combined with MochiKit's iteration tools.

In addition to these data structure tools, MochiKit allows you to create DOM elements dynamically, convert DOM objects to strings, retrieve elements matching class or type attributes, and swap DOM objects for other DOM objects.

Getting Started with MochiKit

Want to see some MochiKit magic? The Demos page on **mochikit.com** has several interesting samples. One of MochiKit's examples is an interactive JavaScript Interpreter, executing whatever JavaScript code you enter. In addition, this interpreter provides documentation for MochiKit functions—via help()—returning a clickable link to the passed function.

MochiKit's documentation page also uses some (MochiKit-enabled) JavaScript to display a list of the sub-namespaces of MochiKit. When the main page loads, it dynamically creates a list of sub-namespaces (MochiKit.Async, MochiKit.Base, MochiKit.DOM and so on). Clicking any item in the list expands or collapses the documentation for the namespace. No actual list of functions exists—it's all computed dynamically by (asynchronously) requesting each documentation page from the server, then parsing the DOM of each one. It is worth noting that the documentation on-line corresponds with the release of MochiKit currently in development and not the current “stable” version; each function also lists the version of MochiKit in which it appeared.

Good frameworks have tests to validate their functionality, and MochiKit is no different. The MochiKit test page (see Resources) has almost 800 tests validating MochiKit. This automated framework allows MochiKit to be validated easily on all the supported browsers.

Example 1: MochiKit in a Simple Page

The MochiKit package is easy to install. If you downloaded the zip version, move the Mochikit folder in the lib folder to your Web space. To use the Subversion version, copy the Mochikit folder from your checkout.

Create the following HTML page:

```
<!DOCTYPE html PUBLIC
"-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"
    xml:lang="en" lang="en">
<head>
    <meta http-equiv="Content-Type"
        content="text/html; charset=utf-8" />

    <title>MochiKit Example #1</title>
    <script type="text/javascript" charset="utf-8"
        src="Mochikit/MochiKit.js" />
    <script type="text/javascript" charset="utf-8"
        src="example1.js" />
</head>

<body>
        <p style="background: red;
            padding-top: 1em;">
        Hello world this is Mochikit!</p>
</body>
</html>
```

Notice the following line in your <head> section, which loads MochiKit:

```
<script type="text/javascript" charset="utf-8"
    src="Mochikit/MochiKit.js" />
```

Also notice that in the body is a paragraph with a red background. This box is plain-looking. Wouldn’t it be great if it had rounded corners? MochiKit makes it easy.

First, we want to execute a JavaScript function when we load the page. The MochiKit function for that is MochiKit.DOM.addLoadEvent(). In JavaScript, the namespace specifiers are optional, but for clarity, we include them here.

We create a separate file for our JavaScript (a separate file is best). In the file, we have one function and a call to addLoadEvent():

```
function myLoadFunction()
{
    MochiKit.Visual.roundClass('p', null);
};

MochiKit.DOM.addLoadEvent(myLoadFunction);
```

MochiKit’s roundClass allows you to specify an entire class type, and it rounds all of the elements of that class. You also can round elements selectively with MochiKit.Visual.roundElement(), which accepts either a string specifying the id or an element object.

Example 2: Turning Clicked Links into SPANs

Our second example uses MochiKit.Base.map(), MochiKit.Base.partial(), MochiKit.Signal and MochiKit.DOM to create a link that can be clicked only once and then goes away:

```
function myLoadFunction(eventObj)
{
    /*
        Find all A elements whose class is
        "onepush" and make them all to call
        handleJSHREFClick() in response to click
    */

    elementsToApplyOn =
        MochiKit.DOM.getElementsByTagAndClassName(
            "a", "onepush");

    /*now that we have all of the elements that
    match our transformation query run our function
    that connects everything, calling it once
    for every item.*/
    MochiKit.Base.map(
        connectOneClickOnly, elementsToApplyOn );
}

MochiKit.Signal.connect(window, "onload", myLoadFunction);
//end main and load
```

MochiKit’s Signal module allows us to have functions called when events happen (it’s based off Qt’s signal/slot mechanism). In the case of the last line of code here, we are having the window object’s onload event call our load function. Careful readers will remember MochiKit.DOM.addLoadEvent() used in the first example, and yes, we are using similar functionality with MochiKit.Signal. Be aware that once you choose one method of handling the load event, you cannot, in the same script, use the other method—they are incompatible in that way.

In JavaScript, and programming in general, the best code is code you don’t have to write.

When the document loads in the browser, myLoadFunction() is called. This function gathers all the A elements whose class is oneclick and passes them, one by one, to the connectOneClickOnly function. The MochiKit.Base.map function could be seen as a convenience in writing the following pattern:

```
for (i = 0; i < elementsToApplyOn.length; i++)
```

```
        connectOneClickOnly(elementsToApplyOn[i]);
```

Next, we examine the real sweet spot for partial() and bind()—providing parameters to callback functions:

```
function connectOneClickOnly(linkElement)
{
    /*
        This function gets called for each A of
        type "oneclick" we have. Hook it up so
        that our handleJSHREFClick gets called
        (properly) when a user clicks the
        linkElement object
    */

    /*Each of our calls to handleJSHREFClick, in
    addition to getting the event object passed to
    it via MochiKit.Signal, also gets called with
    the object to call to create our replacement.
    */

    newH = partial(handleJSHREFClick, makeNewObj);

    MochiKit.Signal.connect( linkElement,
        'onclick', newH );
} //end connectOneClickOnly
```

Remember, MochiKit.Base.partial() and MochiKit.Base.bind() allow for runtime creation of functions that are based on other functions. These wrapper functions can provide parameters or even remap the JavaScript for this variable to the functions they are wrapping.

In this case, we use MochiKit.Base.partial(), because there is no way to provide arbitrary arguments to functions called via MochiKit.Signal (or any other MochiKit methods that call back to user functions). Using MochiKit.Base.partial(), we can pass as many parameters as we want, and MochiKit is none the wiser. In this case, we supply a function, which creates the replacement SPAN element, to our event handler callback. We have MochiKit.Signal call our function when the user clicks on our element:

```
function makeNewObj(target)
{
    /*
        Create a new item to replace our target
        with.
        Return the created element
    */

    makeNew = SPAN({});

    inHTMLStr = "One Click Only!";

    makeNew.innerHTML = inHTMLStr;

    return makeNew;
}

function handleJSHREFClick(makeNewF, eventObj)
{
    /*
        When one of our "oneclick" elements have
            been clicked, this function runs.
    */

    ourTarget = eventObj.target();

    /*stop the event right here (don't let it go to
    the href listed in the A)
    Here also so the event is stopped if we have
    errors further on*/

    eventObj.stop();

    //call our function that creates new elements
    makeNew = makeNewF(ourTarget);

    swapDOM(ourTarget, makeNew);
} //end click functionality code
```

The handleJSHREFClick() function is called, as previously mentioned, when a user clicks on our oneClick A elements. Normally, this function would accept only one parameter: the eventObj parameter passed by MochiKit.Signal. Because we used MochiKit.Base.partial(), the function is passed another parameter (in this case, the function to call to create our replacement object).

MochiKit.Signal takes care of the hard work of handling events. No matter what browser the user is using (or what event modal that browser uses), the JavaScript code doesn't have to change—the custom event object from MochiKit.Signal takes care of all that for you. Through the passed event object, you can get the key state, the mouse state, the object that triggered the event, the object connect()ed to the event, what type of event happened, and even stop the event from propagating further by preventing the default action of the DOM object.

handleJSHREFClick() swaps the item the user clicked on with the element we created. First it stops the event, because (in this instance) we don't want to go any further (that would follow the HREF element of the A, something we don't want to happen in

MochiKit's tools for functional programming allow functions to be created dynamically, or they may simply provide more extensive (or less broken) behaviors for functionality provided already in JavaScript.

this particular example).

The code that creates our replacement span is in the obvious place: makeNew = SPAN({}), yet this monster requires some explanation. DOM elements are created through MochiKit via functions in MochiKit.DOM. Like other MochiKit modules, there's a lot here. Functions to create, functions to query, swap and even convert DOM elements can be found in this module—getElement(), getElementsByTagAndClassName(), currentDocument(), currentWindow() and createDOM() to list a few. MochiKit.DOM.createDOM() is what is (indirectly) used here. MochiKit.DOM includes shortcuts to create common DOM elements (A, BUTTON, BR, CANVAS, DIV, FIELDSET, FORM, H1, H2, H3, HR, IMG, INPUT, LABEL, LEGEND, LI, OL, OPTGROUP, OPTION, P, PRE, SELECT, SPAN, STRONG, TABLE, TBODY, TD, TEXTAREA, TFOOT, TH, THEAD, TR, TT and UL, at the time of this writing). These are called with the attributes specified in the associative array parameter. In the passed array, each key corresponds to an attribute of the HTML element. For example, to create a link to example.com, the code would be:

```
makeNew = A({'href':'http://www.example.com'});
```

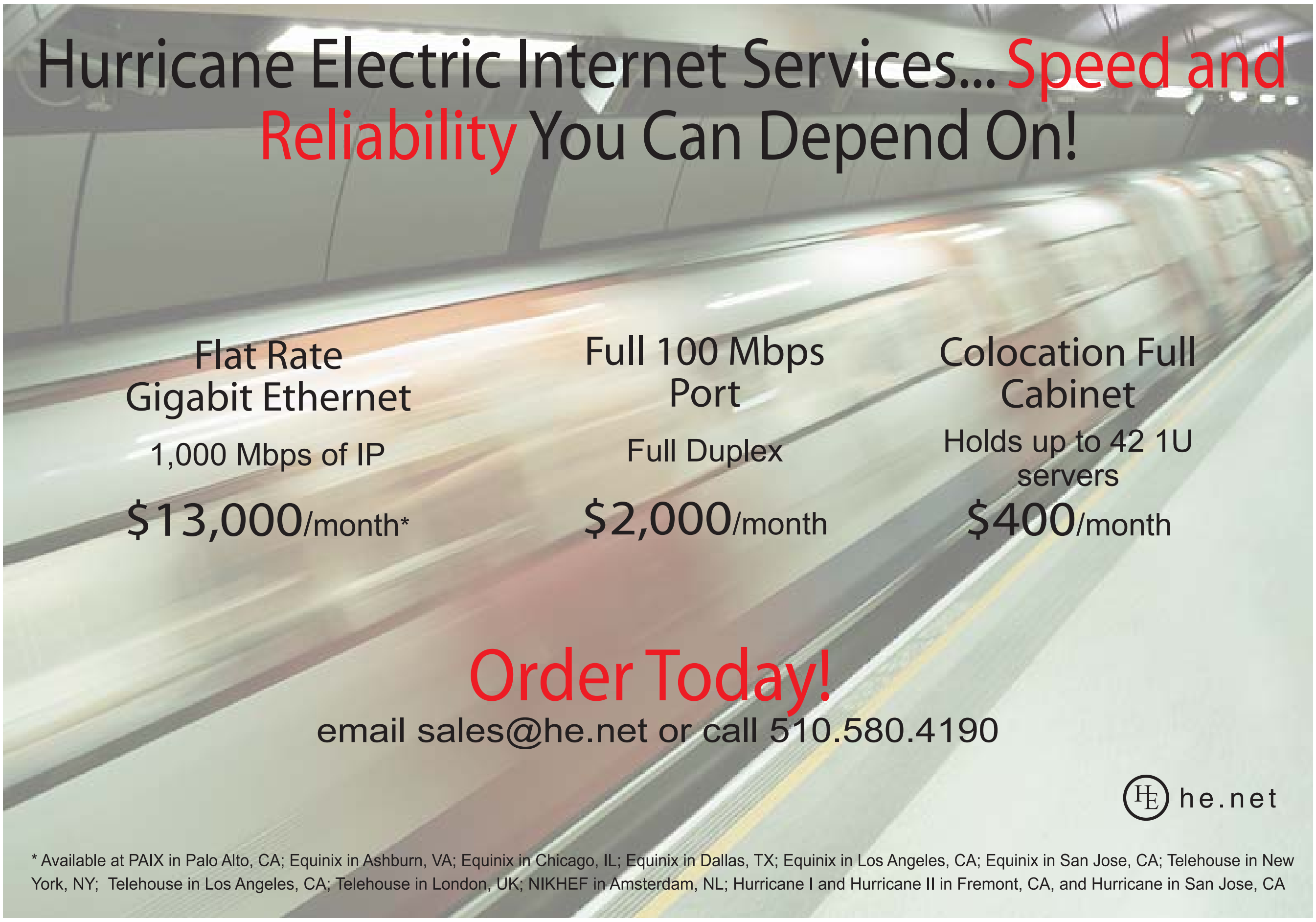
This example covered a lot of ground—MochiKit.Signal.connect(), MochiKit.Base.partial() and MochiKit.DOM.createElement(). However, we've scratched only the surface of these, and there's a whole lot more of MochiKit to cover. The next example takes the normal login box found all over the Web and "Web 2.0's" it up.

Example 3: Simple User Name/Password UI with Ajax

Our final example creates a dynamic login screen. The idea is to provide feedback for an incorrect password without refreshing the entire Web page. On success, the main menu screen loads without requiring a full page reload.

MochiKit's logging functionality isn't only for debugging. Instead, it can be co-opted to be an easy error-reporting mechanism. We use this mechanism to report both incorrect user name/password errors and errors with the login script on the server (server down and so on):

```
function fatalLog(sendLogTo, logInst)
    /*handles our logError calls, displays in
        element param #1, displaying the error in
        yellow then fading it out after 5 seconds*/
{
    var errStr = logInst.info.join(" ");

    if (errStr.length == 0)
        errStr = "Unknown error";
    sendLogTo.innerHTML =
        "<pre> We're sorry an error occurred: " +
        errStr + ". Please try again. </pre>";

    Highlight( sendLogTo, {delay: 1, duration: 5} );
    //Yellow Fade Technique
```



Many of MochiKit's visual effects were ported from Scriptaculous for MochiKit 1.4.

```
    //see Visual.DefaultOptions documentation for
    //associative array options that affect
    //MochiKit.Visual

} //end fatalLog
```

Co-opting MochiKit's log functionality means that all errors can be reported using MochiKit.Logging calls. During debugging of this site, something else could be set up to handle the errors—perhaps logging more information or breaking into a source-level debugger.

We have an issue here: when parts of the page redraw, it's sometimes not obvious that part of the page contains new information. The user could be waiting for something to happen when, in fact, it already has. The Ajax community has solved this by highlighting changed elements in yellow (traditionally) and fading back to the normal background after an amount of time. This technique (the yellow fade technique) is used, in particular, by 37 Signals (the minds behind Ruby on Rails). The technique is used here to prompt users to react to the error (for example, try the password again). MochiKit makes this technique very easy with MochiKit.Visual.Highlight(). We can specify how long to delay before starting the effect, how long it should last and other options, all specified via an associative array (see the keys outlined in MochiKit.Visuals.DefaultOptions). Many of MochiKit's visual effects were ported from Scriptaculous for MochiKit 1.4.

As shown in the next code sample, our load function plugs our fatalLog function into the MochiKit.Logging framework, then sets up the environment to call our subClicked handler when the submit button is pressed:

```
function subClicked(eventObj)
    /*checks the username/pw*/
{
    d = MochiKit.Async.doSimpleXMLHttpRequest(
        cgiLoginLocation,
        { 'username': getElement('uname').value,
          'passw': getElement('pword').value } );

    d.addCallback(handleServerResult_Login);
    d.addErrback(handleServerError);

    getElement('waitMsg').innerHTML =
        "Please wait...";

    clearError();
    //clear the old error message, it doesn't apply

}   //end subClicked

function myLoadFunction()
{
    /*first create our Logging listener, and direct
    our generic function the errMsg span we have*/
    fatalLogTo = partial(fatalLog,
        MochiKit.DOM.getElement('errorMsg'));

    MochiKit.Logging.logger.addListener('ERRORONLY',
        null, fatalLogTo);

    MochiKit.Signal.connect( 'submit', 'onclick',
        subClicked );

    //now hide the place where our main menu will be
    MochiKit.Style.hideElement("Result");
}

/*connect our event handlers right off*/
MochiKit.Signal.connect(window, "onload",
    myLoadFunction);
//end script
```

MochiKit.Async.doSimpleXMLHttpRequest is the simplest way in MochiKit to do an Ajax request. It accepts the URI to which to send the request, the GET parameters for that URI, and it returns a MochiKit.Async.Deferred object. For advanced requirements, MochiKit.Async also provides functionality for sending POSTs instead of GETs, obtaining JSON documents and more. All MochiKit.Async Ajax functions return a Deferred object, so (beyond the construction) these functions behave exactly as in our example doSimpleXMLHttpRequest().

By the time we have our Deferred object, the Ajax event already is sent off. Because the call is asynchronous, it may be several seconds until an answer is received—plenty of time to set up functions (callbacks) that will handle error or success. The MochiKit.Async.Deferred object is merely a guarantee that something will happen—what happens is up to us. When a Deferred object comes back from a MochiKit Ajax request, execution of the script continues while we wait. This allows us to set up our callbacks and do whatever other housekeeping is required. When a response comes back from the server, good or bad, the appropriate callback set up in the Deferred object is called.

The success and failure callback functions get one status parameter from MochiKit.Async. Additional parameters can be sent to the callback functions by passing them to MochiKit.Async.addCallback()/MochiKit.Async.addErrback(). The status parameter always is the last parameter provided to the callback function. Success functions are called if the HTTP status code from the XMLHttpRequest is 200, 201, 204 or 304. The error function is called if the status is any other number. Success functions get a standard XMLHttpRequest object as the status object. If you've never seen one of these, the important items are responseText and status. Failure functions get an XMLHttpRequestError parameter. The important items of this object are message and number.

With all this background, we can handle our XMLHttpRequest. Remember, the idea behind this example is to do all the work required in a login screen without refreshing the page. To accomplish this, first we send an XMLHttpRequest to check the user name and password and return a session ID. In our case, the responseText from the CGI is a string whose contents are formatted as a JavaScript array. With this formatting, we could run JavaScript's eval() to get the result as a JavaScript array object, or we could simply call MochiKit.Async.evalJSONRequest() to do the same thing. In our case, the array contains (in order): a success boolean, a failure message (or an empty string, if the call succeeded) and a session ID (or 0, in the case of failure).

The success callback handleServerResult_Login() should check the success passed back from the CGI. If we have a success, it sends a second Ajax request to load the main menu. During this second request, the server checks the session ID, making sure it is valid, then returns HTML code for the main menu (or an error message). When this (second) request succeeds, the handleServerResult_Manage() clears the login controls away (which we do with a cool transition effect, courtesy of MochiKit.Visual) and inserts the main menu code. On success, cookies can be set to save the session ID (see **www.quirksmode.org/js/cookies.html** for cookie manipulation functions; future versions of MochiKit may include cookie manipulation functions):

```
function handleServerError(err)
{
    getElement("waitMsg").innerHTML = "";
    logError( err.message + " (error #" +
        err.number + ")" );
    //err.message will be like "Request Failed"
}   //end handleServerError

function handleServerResult_Manage(sessionID, res)
{
    //get rid of our login controls - we're very
    //much validated by this point
    slideUp( getElement('loginDlg') );

    //our responseText will be the HTML for the
    //"main menu"
    whereTo = getElement("Result");
    whereTo.innerHTML = res.responseText;
    MochiKit.Visual.appear( whereTo, {delay: 1} );

    createCookie("sessionID", sessionID, 1);

    getElement("waitMsg").innerHTML =
        "Cookie value = " + readCookie('sessionID');

}   //end handleServerResult_Manage


function handleServerResult_Login (res)
{
    getElement("waitMsg").innerHTML = "";
    //no more waiting required!

    //res.responseText contains our result. Our CGI
    //returns it as a JS array inside a string
    //but just let MochiKit handle it for us
    resList = MochiKit.Async.evalJSONRequest(res);
    success = resList[0];
    failMsg = resList[1];
    sessionID = resList[2];

    if (success)
    {
        //send off _another_ AJAX request
        //(passing session id), this time to get
        //the main menu screen
        d2 = MochiKit.Async.doSimpleXMLHttpRequest(
            cgiMainMenuLocation,
            {'sessionID': sessionID } );

        d2.addCallback( handleServerResult_Manage,
            sessionID );
        d2.addErrback(handleServerError);

    }
    else
    {
        logError(failMsg);
    }
}   //end handleServerResult_Login
```

Conclusion

MochiKit is a powerful toolkit, making advanced features easy in JavaScript. From visual effects, event handling and functional tools to Ajax functionality, MochiKit puts amazing features at your fingertips.■

Ryan Wilcox is the founder of Wilcox Development Solutions (www.wilcoxd.com) specializing in cross-platform application development and Web solutions. He also considers himself a "general practitioner" of programming languages. His only hope is never having to pull out the memory cores from a spaceship's insane computer.

Resources

The MochiKit Screencast: **mochikit.com/screencasts/MochiKit_Intro-1.html**. Although it covers only MochiKit 1.1, it should give an idea about what is possible with MochiKit. As an additional feature, the entire screencast uses the JavaScript Interactive Interpreter from the MochiKit sample pages.

The MochiKit Test Page: **www.mochikit.org/tests/index.html**. Run the tests for the MochiKit framework by visiting this page.

The MochiKit Subversion Repository: **svn.mochikit.com/mochikit**. Grab the current or in-development version of MochiKit, or stay up to date with the latest update releases.

JavaScript World Cup: **www.sitepoint.com/article/javascript-library/2**. Overview of all the current heavyweights (as of this writing) in the JavaScript library space: Dojo, Prototype/script.aculo.us, MochiKit and the Yahoo UI Library.

Zebra Table Showdown: **jquery.com/blog/2006/10/18/zebra-table-showdown**. Makes a table with alternating background rows (like the iTunes playlist) with the major JavaScript library players.

Turbogears: **www.turbogears.org**. MochiKit is part of the TurboGears Python Web application framework, but, as we've seen here, it can be used completely separate from TurboGears.

Scriptaculous: **script.aculo.us**. Some of MochiKit's visual effects functions were ported from Scriptaculous.

Four Cool Ajax Plugins for WordPress

MARCO FIORETTI

Here is how to install and use four dynamite plugins for the WordPress content management system.

WORDPRESS IS PROBABLY the most popular free software solution for publishing and managing a dynamic personal Web site. It installs easily and quickly, it has a lot of plugins that extend its functionality, and it can be used with practically every Web hosting provider on the planet. Ajax (Asynchronous JavaScript and XML) is a mix of Web technologies used for building Web sites that respond much more quickly and smoothly to user input. Combining WordPress and Ajax to build a snappier Web site in a few minutes is much easier than it may seem at first, but you need to follow the instructions carefully to avoid a few little traps.

This article discusses four Ajax plugins that can make your WordPress-based Web site more dynamic, friendly and fun for visitors. The only prerequisite is that you have an already-working WordPress installation (we used version 2.1) and, of course, a complete backup of it just in case something goes wrong.

AjaxWp: Raising the Speed Limit

Let's start with a plugin that doesn't generate impressive snapshots, but makes your site less boring by making navigation faster. AjaxWp improves the responsiveness of your WordPress pages with a relatively simple trick; it dynamically replaces all the internal links to other parts of your Web site with onclick() JavaScript function calls.

When visitors click on these modified links, their browsers launch the scripts embedded in the AjaxWp code. These scripts then request the new page, all by themselves, in the background. In the meantime, the visitors' browsers will not go blank; the header, footer, sidebars—basically every part of your Web site that is common both to the current page and the one just requested—remain fully readable where they are.

The part to be replaced, and that only, gradually vanishes, and the block of new content takes its place as soon as the AjaxWp scripts have it ready. During this phase, to show that it is actually doing something, AjaxWp superimposes an animated GIF of a rotating wheel to the area it is replacing. The animation with which AjaxWp moves from the old page to the new one, courtesy of the Script.aculo.us library, can be set to appear, slide or blind.

How slowly or quickly all this happens depends on the speed of the Internet connection, the load on your Web server and the speed of your visitors' computers. If something goes wrong, after a programmable timeout, AjaxWp simply lets the browser load the page in the standard mode.

AjaxWp depends on a few JavaScript libraries that are included in the distribution. To use this plugin, download the latest tarball from the home page, unpack it, and move its JavaScript folder, the animated GIF and a PHP file called, you guessed it, ajax-wp.php, inside your WordPress installation. Then the fun begins.

AjaxWp can work in two modes: Quick, which is easier to configure and use, or Optimized. Whichever mode you choose, the home page and the README file describe in detail all the actual steps of the installation procedure, but I summarize the main points here.

In Quick mode, every AjaxWp call requests a whole new page from the server and then extracts from it the single area that must be refreshed in the browser window. Other than the steps described above, you need to add only a few lines of PHP code to the header file of your WordPress theme to start using Quick AjaxWp.

Optimized mode is faster and more efficient, because only the pieces of the pages that *have* to change are requested from the server and dropped as they arrive in the right part of the browser window. To make this work, however, you have to create an AjaxWp version of your theme—that is, add to each of its pages the snippets of PHP code described in the on-line documentation. Depending on your theme, this may take a bit of tweaking to get right.

Regardless of which mode you set up, once everything is in place, users who have JavaScript enabled in their browsers will enjoy a faster or at least much smoother navigation of your pages. Users without JavaScript enabled still will be able to load and read the pages in the old, pre-Ajax way.

Some advice: keep a copy of all the original WordPress files and restart from those if you configure Quick AjaxWp and decide to switch to Optimized mode later. If you mix or repeat installation steps or JavaScript calls in the code, strange things will happen.

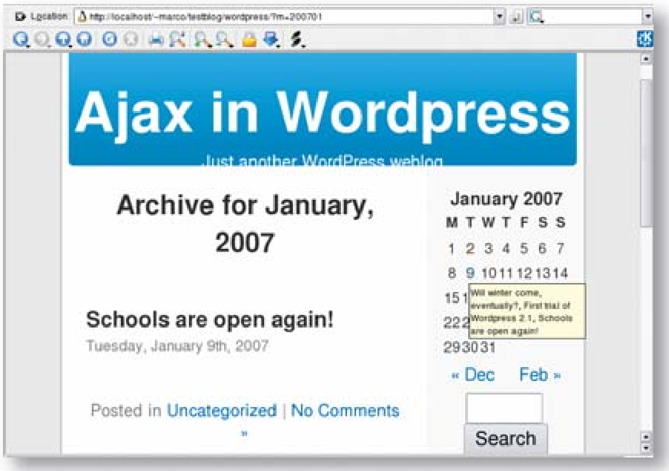
Whether you choose Quick or Optimized mode, don't forget to spend a few minutes checking the configuration variables of AjaxWp to adapt them to your taste and, more important, to your theme and general WordPress setup. The two most important options are ajax_wp_blog_base_path, which is the relative path from the root of your Web server to your WordPress installation folder, and the list of pages (ajax_wp_ignore_links) that should be loaded normally rather than through AjaxWp.

Calendar

WordPress publishes and archives authors' posts in chronological order without any effort. It often comes naturally, both for you and your readers, to track and retrieve such posts through a calendar in the home page. WordPress does have a built-in calendar, but the Ajax-based one shown in Figure 1 is more dynamic and pleasant to use.

The installation is simple, but not without a few issues. According to the README file distributed with the code, you must uncompress the tarball only in the WordPress plugins folder, activate it in the Administration pages, and add, in the index.php files of your WordPress theme, these three lines of code right where you want to place the calendar:

```
<div id="calendar">
<?php get_calendar(); ?>
</div>
```


Location http://localhost/~marco/testblog/wordpress/?m=200701

Ajax in Wordpress

Just another WordPress weblog

Archive for January, 2007

Schools are open again!

Tuesday, January 9th, 2007

Posted in Uncategorized | No Comments »

January 2007

M T W T F S S

1 2 3 4 5 6 7

8 9 10 11 12 13 14

15 1

22 2

29 30 31

Will winter come, eventually?, First trial of Wordpress 2.1, Schools are open again!

« Dec Feb »

Figure 1. The Calendar Plugin for WordPress

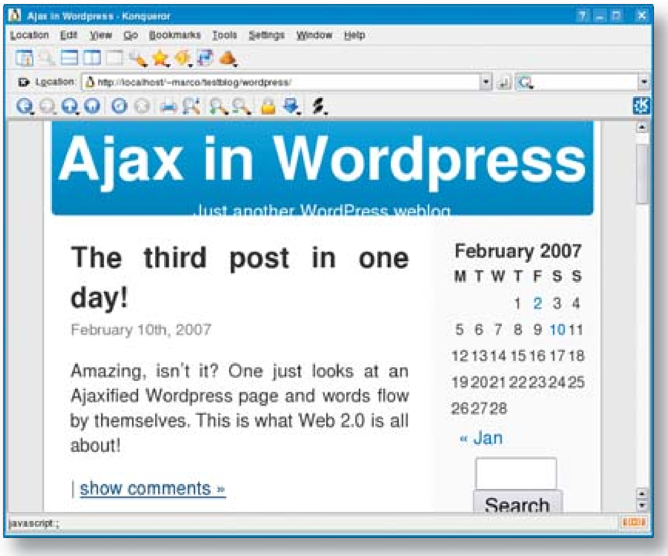
After saving the file, you should configure the plugin only in the Options→AjaxCal Administration page. In practice, this may depend on your theme and WordPress version. The calendar you see in the screenshots for this article, for example, appeared by placing the get_calendar() function call, without the opening and closing <div> tags, inside the sidebar div element of the sidebar.php file.

Once the calendar appears, go to the Options→AjaxCal page in the Administration section to configure it. You can, for example, choose how to display the links to all the posts on a given day or give the whole calendar a completely different appearance from the rest of the Web site. Apart from eye candy, don't forget to set the display of Future and Draft posts according to your preference. The most important option of the calendar, however, may very well be the Load prototype.js tick box; this is a JavaScript library, and there is no need to load it twice, lest it confuse the browser. Therefore, if you place the call to the calendar in the PHP code *after* your WordPress theme or some other plugin already has loaded prototype.js, deselect that box.

What Did People Say?

The joy of reading a blog or any dynamic Web site is being able to add a comment to each page or, even more often, to read what other visitors had to say. The normal way to do this is to follow the link to the whole page or to its comment section, but Ajax comes to the rescue to speed up even this WordPress task.

The Inline Comments plugin makes all the comments to a specific post appear or disappear in the home page, according to each visitor's preference. This happens in the usual Ajax fashion—that is, without freezing the browser or blanking the whole window. After you have installed and activated this plugin, the home page should look like the one shown in Figure 2. One click on the show comments link loads all the comments for that post and makes them slowly appear, right where you want them to be, as shown in Figure 3. After that, another click on hide comments returns the page to its original state. The beauty of the plugin is that all the comments to a post are downloaded only once and cached in the browser. If visitors hide them and then decide they want to read them again, they reappear instantly. The order in which comments are displayed (newest or oldest first) can be set in the Plugins→Inline Ajax Comments page. As with the calendar, the CSS styles for the comment box are customizable separately from the rest of the theme.


Ajax in Wordpress - Konqueror
Location Edit View Go Bookmarks Tools Settings Window Help
Location: http://localhost/~marco/testblog/wordpress/

Ajax in Wordpress

Just another WordPress weblog

The third post in one day!

February 10th, 2007

Amazing, isn't it? One just looks at an Ajaxified Wordpress page and words flow by themselves. This is what Web 2.0 is all about!

| show comments »

February 2007

M T W T F S S

1 2 3 4

5 6 7 8 9 10 11

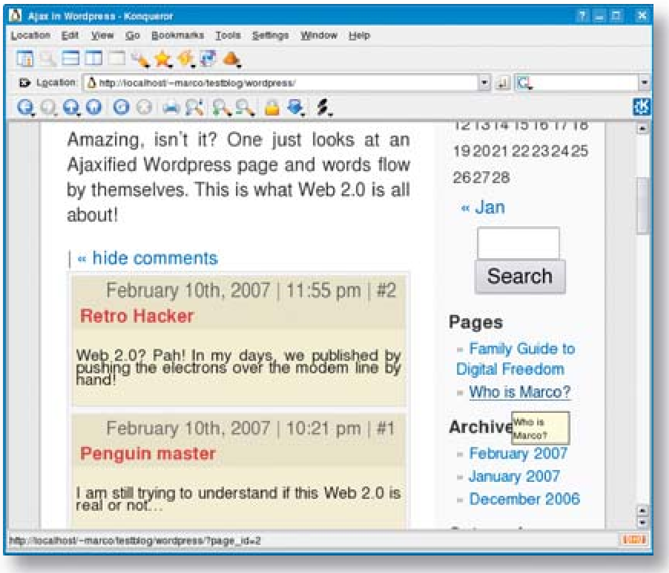
12 13 14 15 16 17 18

19 20 21 22 23 24 25

26 27 28

« Jan

Figure 2. This plugin adds the show comments link.


Ajax in Wordpress - Konqueror
Location Edit View Go Bookmarks Tools Settings Window Help
Location: http://localhost/~marco/testblog/wordpress/

Amazing, isn't it? One just looks at an Ajaxified Wordpress page and words flow by themselves. This is what Web 2.0 is all about!

| « hide comments

February 10th, 2007 | 11:55 pm | #2

Retro Hacker

Web 2.0? Pah! In my days, we published by pushing the electrons over the modem line by hand!

February 10th, 2007 | 10:21 pm | #1

Penguin master

I am still trying to understand if this Web 2.0 is real or not...

12 13 14 15 16 17 18

19 20 21 22 23 24 25

26 27 28

« Jan

Pages

- Family Guide to Digital Freedom
- Who is Marco?

Archives

- February 2007
- January 2007
- December 2006

Figure 3. You can hide the comments too.

To add inline comments, download the plugin, place all its files in the plugins folder of your WordPress installation, and activate this function in the WordPress Plugins Administration page. After that, you need to add two new lines of code to the index.php file of your theme—one creates the link that opens or hides the comments box:

```
<?php ajax_comments_link(); ?>
```

The other instruction needs to go where the comment box itself must appear:

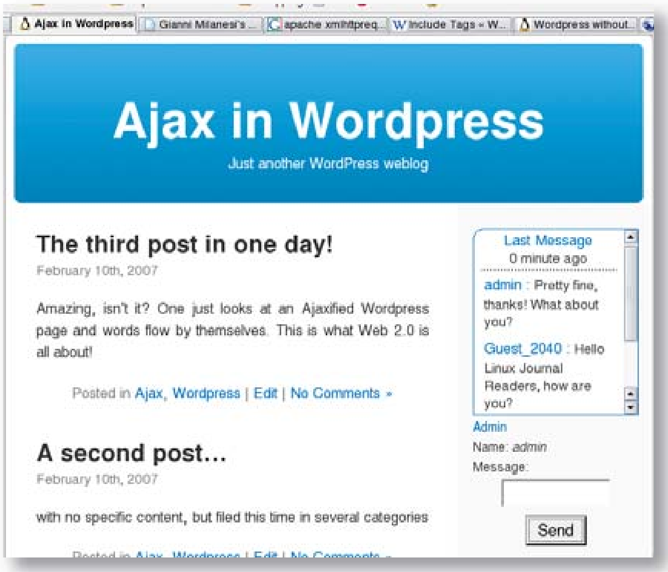
```
<?php ajax_comments_div(); ?>
```

Let's Chat!

WordPress and all other popular blogging packages have many features specifically designed to make conversations between authors and readers as easy as possible. If you are used to instant messaging, however, you may think nothing is better than instant messaging for a quick on-line conversation. Don't worry; there's no need to leave your beloved WordPress home page to have such conversations. The Ajax-based Wordspew/Shoutbox plugin adds real-time chat functionality to any WordPress Web site. Installation is possibly the simplest one of all the plugins described in this article: unpack the tarball in the WordPress plugins directory, and call Shoutbox with this line of code in the piece of the theme where you want it to appear:

```
<?php jal_get_shoutbox(); ?>
```

Figure 4 shows the result. Whenever anyone writes some text in the Message input field, everyone else who is visiting the home page at that moment will see it, without doing anything, the next time the Shoutbox area refreshes itself, and everyone will be able to answer in the same way. Shoutbox users also can add their name and home URL, if they choose, as well as use emoticons or links in the message text.



Ajax in Wordpress

Just another WordPress weblog

The third post in one day!

February 10th, 2007

Amazing, isn't it? One just looks at an Ajaxified Wordpress page and words flow by themselves. This is what Web 2.0 is all about!

Posted in Ajax, Wordpress | Edit | No Comments »

A second post...

February 10th, 2007

with no specific content, but filed this time in several categories

Last Message
0 minute ago

admin : Pretty fine, thanks! What about you?

Guest_2040 : Hello Linux Journal Readers, how are you?

Admin

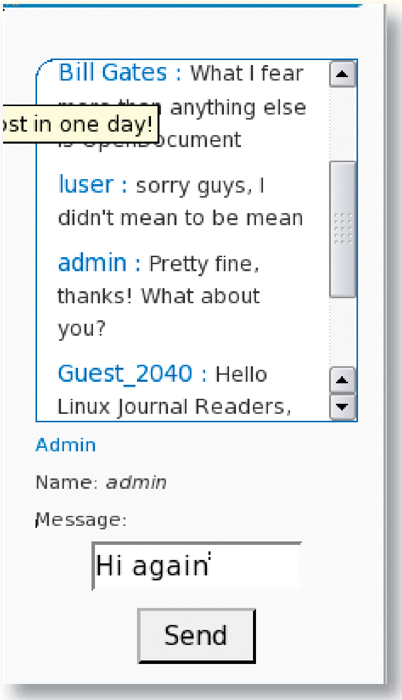
Name: admin

Message:

Figure 4. Shoutbox lets people chat without refreshing the page.

The scrollbar on the right allows newcomers to follow the whole conversation (Figure 5). The refresh interval is programmable. When it expires, new posts appear highlighted in a different color, which then fades away in the background after an equally programmable interval.

Almost everything else in the Shoutbox is configurable. You can set all the options from the Manage→Live Shoutbox page. The screenshots here show the vanilla version, but you can change the colors of user names, text and background of all comments. Even the one-line input area can be replaced with a larger field, but this obviously uses more space and may ruin the overall layout of your pages. One Shoutbox feature you might want to disable as soon as possible is the sound alert when new messages are loaded. You don't want your coworkers to know when you're chatting in your browser, do you?



Bill Gates : What I fear more than anything else is OpenDocument

st in one day!

luser : sorry guys, I didn't mean to be mean

admin : Pretty fine, thanks! What about you?

Guest_2040 : Hello Linux Journal Readers,

Admin

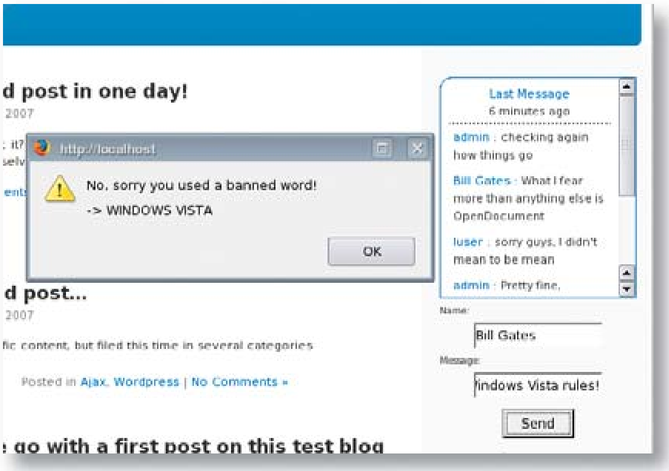
Name: admin

Message:

Hi again

Figure 5. You can scroll back to view previous chat messages.

Security-conscious readers will immediately spot the potential for abuse here, but Shoutbox has two configurable mechanisms to prevent spammers from filling it with garbage. One is a place (Options→Discussion→Comments moderation) where you can enter a list of banned words, URLs and sentences. The package includes a sample word list; to add new ones, simply type them in the right place on the list. People trying to use banned words will see the alert box shown in Figure 6. The drawback in using the banned word list is that it requires continuous monitoring and maintenance. To avoid this burden, it is much better, without giving up the list itself, to set Shoutbox to accept comments only



d post in one day!

2007

http://localhost

No, sorry you used a banned word!
-> WINDOWS VISTA

d post...

2007

fic content, but filed this time in several categories

Posted in Ajax, Wordpress | No Comments »

go with a first post on this test blog

Last Message
6 minutes ago

admin : checking again how things go

Bill Gates : What I fear more than anything else is OpenDocument

luser : sorry guys, I didn't mean to be mean

admin : Pretty fine,

Name:

Bill Gates

Message:

indows Vista rules!

Figure 6. You can filter profanity, among other things.

from registered users. Anonymous visitors still will be able to see the chat in progress.

Besides English, the Wordspew Shoutbox is also available in about ten other European languages. If your language is already supported, simply download the corresponding PO-MO files from the plugin home page, and place them in the Wordspew folder on the server. Otherwise, the author welcomes localizations in other languages.

Final Tips

Ajax still is a relatively new technology. Depending on which WordPress version you run, how you configured it and which theme you chose, you may experience messed-up internal links, misaligned blocks or similar problems if you download everything described in this article and simply drop it in your WordPress installation.

One reason for these issues is the simple fact that all these plugins are still under active development. In the Calendar version (0.8.3) we tested, for example, one php tag was missing (surely due to a typing mistake before packaging) from the source file called ajaxcalendarscript.php. To make it work, we had to replace <? with <?php on line 89 of that file. Surely all the plugins will have more stable interfaces by the time you read this article.

Another reason is that, at least at the time of this writing, several plugins are packaged with their own copies of the same (or different) versions of some JavaScript library. This spares you having to find those libraries, but it also might confuse some browsers. Install the plugins one at a time, starting from the one you need the most, and don't move to the next until you're sure everything works as you want, and be sure you have a backup of all your WordPress files.

Another trick that can spare you a lot of frustration, not only with these plugins but with any JavaScript-based Web application, is always to keep two windows open during the installation and testing phases. The first one should show, if you have access to it, the last lines of the error log file of your Web server. This will make it evident if things are going wrong because some file is not in the expected location. The other window should be the JavaScript console of Firefox or Mozilla, which is where these browsers report any problems they have with executing the code embedded in a Web page.■

Marco Fioretti is a hardware systems engineer interested in free software both as an EDA platform and, as the current leader of the RULE Project, as an efficient desktop. Marco lives with his family in Rome, Italy.

Resources

AjaxWp: **www.giannim.com/blog/index.php?page_id=13**

Shoutbox: **pierre.sudarovich.free.fr**

Ajax Calendar: **dunamisdesign.net/?p=7**

Inline Comments: **kashou.net/blog/inline-ajax-comments**

An Ajax-Enhanced Web-Based Ethernet Analyzer

Combine Ruby, Ajax and bash with CGI scripts to monitor server-bound processes.

PAUL BARRY

I've spent the past six months or so playing with Ruby. I blame the July 2006 issue of *Linux Journal* for this hiatus from my programming language of choice, Perl, as that issue opened my eyes to the possibilities of using Ruby as a serious tool. I still love, use and teach Perl, but I'm spending more and more time programming in Ruby.

I follow the same process when learning any new programming technology: I identify a good book and work through it, and then start to use the language to build some of the things I love to build with Perl. Identifying the book was easy. The second edition of *Programming Ruby* by Dave Thomas (known as *The PickAxe*) is as good an introduction as you are likely to find for any programming language, not just Ruby. Once I'd worked my way though *The PickAxe*—creating a Ruby tutorial as I went along (see Resources)—I was itching to write some real code. I started with a type of tool that I enjoy building with Perl: a custom Ethernet analyzer.

Does the World Really Need Another Ethernet Analyzer?

At this point, probably more than a few readers are saying to themselves: why bother creating an Ethernet analyzer when tcpdump and Ethereal/Wireshark already exist? Those solutions are excellent tools—which I use a lot—but, I'm often looking to build something that involves additional processing plus the capturing and decoding of Ethernet packets, and this customization invariably involves resorting to custom code. Luckily, it turns out that the technology that underpins both tcpdump and Ethereal/Wireshark—as well as the hugely popular Snort IDS—is available as a library and that a number of language bindings exist for it. The packet capturing library, called libpcap, is available from the same project that brought the world tcpdump and can be downloaded with ease from the Web. In fact, it may well be included within your distribution's package management system; it is if you are running a recent release of Ubuntu (as I am). Obviously, the intrepid programmer can use C with libpcap, but—let's be honest here—life's far too short to work at the C level of abstraction when something more agile is needed. Thankfully, Perl provides an excellent set of modules that work with libpcap, and I devote one-sixth of my first book to discussing the Perl technology in detail. To my delight, and after a little digging around, I also found a set of Ruby classes that interface to libpcap (see Resources).

Listing 1. The dns-watcher.rb Source Code

```
#! /usr/bin/ruby -w

require 'pcap'
require 'net/dns/packet'

dev      = Pcap.lookupdev
capture = Pcap::Capture.open_live( dev, 1500 )

capture.setfilter( 'udp port 53' )

NUMPACKETS  = 50

puts "#{Time.now} - BEGIN run."

capture.loop( NUMPACKETS ) do |packet|

  dns_data = Net::DNS::Packet.parse(packet.udp_data)

  dns_header = dns_data.header

  if dns_header.query? then

    print "Device #{packet.ip_src} "
    print "(to #{packet.ip_dst}) looking for "
    question = dns_data.question
    question.inspect =~ /^\[(.+)\s+IN/
    puts $1
    STDOUT.flush

  end

end

capture.close

puts "#{Time.now} - END run."
```

Creating a Custom Ethernet Analyzer with Ruby

In order to test the libpcap technology for real, I decided to use Ruby to redevelop a tool I created with Perl a number of years ago, which I wrote about within the pages of *The Perl Review* (see Resources). My Perl tool, called wdw (short for who's doing what?), analyzes requests made to a LAN's DNS service and reports on the site names for which the clients are requesting DNS resolutions. In less than 100 lines of Perl code, I'd written a functioning and useful DNS Ethernet analyzer. I wondered how using Ruby would compare.

Now, I present the 20 or so lines of Ruby I used to re-create wdw (for the entire program, see Listing 1). Do not interpret my numbers as any attempt to claim that Ruby can do what Perl does in one-fifth the number of lines of code. It cannot. It is important to note, however, that Ruby's interface to libpcap is significantly more abstract than the one offered by Perl, so Ruby does more in a single call than Perl does, but that has more to do with the choices made by the creators of each language's libpcap binding, as opposed to any fundamental language difference.

Before executing this code, download and install Ruby's libpcap library. Pop on over to the Ruby libpcap Web site (see Resources), and grab the tarball. Or, if you are using Ubuntu, use the Synaptic Package Manager to download and install the libpcap-ruby1.8 package. If a distribution package isn't available, install the tarball in the usual way.

You also need a Ruby library to decode DNS messages. Fortunately, Marco Ceresa has been working hard at porting Perl's excellent Net::DNS module to Ruby, and he recently released his alpha code to RubyForge, so you need that too (see Resources). Despite being alpha, Marco's code is very usable, and Marco is good at releasing a patched library quickly after any problems are brought to his attention. Once downloaded, install Marco's Net::DNS library into your Ruby environment with the following commands:

```
tar zxvf net-dns-0.3.tgz
cd net-dns-0.3
sudo ruby setup.rb
```

My Ruby DNS analyzer is called dns-watcher.rb, and it starts by pulling in the required Ruby libraries: one for working with libpcap and the other for decoding DNS messages:

```
#! /usr/bin/ruby -w

require 'pcap'
require 'net/dns/packet'
```

I can tell my program which network connection to use for capturing traffic, or I can let libpcap-ruby work out this for me. The following line of code lets Ruby do the work:

```
dev = Pcap.lookupdev
```

With the device identified (and stored in dev), we need to enable Ethernet's promiscuous mode, which is essential if we are to capture all the traffic traveling on our LAN. Here's the

Ruby code to do this:

```
capture = Pcap::Capture.open_live( dev, 1500 )
```

The open_live call takes two parameters: the device to work with and a value that indicates how much of each captured packet to process. Setting the latter to 1500 ensures that the entire Ethernet packet is grabbed from the network every time capturing occurs. The call to open_live will succeed only if the program has the ability to turn on promiscuous mode—that is, it must be run as root or with sudo. With the network card identified and ready to capture traffic, the next line of code applies a packet capturing filter:

```
capture.setfilter( 'udp port 53' )
```

I'm asking the libpcap library to concern itself only with capturing packets that match the filter, which in this case is Ethernet packets that contain UDP datagrams with a source or destination protocol port value of 53. As all Net-heads know, 53 is the protocol port reserved for use with the DNS system. All other traffic is ignored. What's cool about the setfilter method is that it can take any filter specification as understood by the tcpdump technology. Motivated readers can learn more about writing filters from the tcpdump man page.

A constant is then defined to set how many captured packets I am interested in, and then a timestamped message is sent to STDOUT to indicate that the analyzer is up and running:

```
NUMPACKETS = 50
puts "#{Time.now} - BEGIN run."
```

The libpcap-ruby library contains the loop iterator, which provides a convenient API to the packet capturing technology, and it takes a single parameter, which is the number of packets to capture. Each captured packet is delivered into the iterator's body as a named parameter, which I refer to as packet in my code:

```
capture.loop( NUMPACKETS ) do |packet|
```

Within the iterator, the first order of business is to decode the captured packet as a DNS message. The Packet.parse method from Marco's Net::DNS library does exactly that:

```
dns_data = Net::DNS::Packet.parse( packet.udp_data )
```

With the DNS message decoded, we can pull out the DNS header information with a call to the header method:

```
dns_header = dns_data.header
```

For my purposes, I am interested only in queries going to the DNS server, so I can ignore everything else by checking to see whether the query? method returns true or false:

```
if dns_header.query? then
```

Within the body of this if statement, I print out the IP source and destination addresses, before extracting the IP name from the query, which is returned by calling the dns_data.question method. Note the use of a regular expression to extract the IP name from the query:

```
print "Device #{packet.ip_src}
 ➥(to #{packet.ip_dst}) looking for "
question = dns_data.question
question.inspect =~ /^\[(.+)\s+IN/
puts $1
STDOUT.flush
```

The program code concludes with the required end block terminators, and then the capture object is closed, and another timestamp is sent to STDOUT:

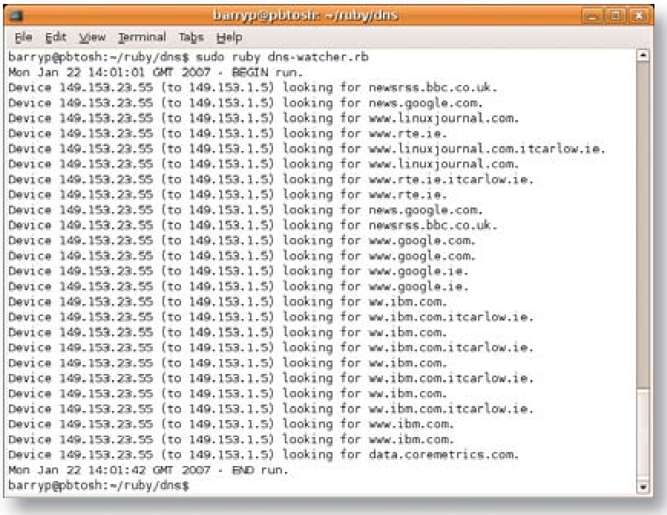
```
end
end
capture.close
puts "#{Time.now} - END run."
```

Running dns-watcher.rb

It's time to give dns-watcher.rb a spin:

```
sudo ruby dns-watcher.rb
```

The output from one such invocation is shown in Figure 1. Note that there are not 50 lines of output, as might be expected. Remember, the program's if statement checks to see whether the captured DNS message is a query going to the server and processes the message only if it is. All other DNS messages are ignored by the program, even though they still contribute to the overall count of DNS packets processed.



```
barryp@pbtosh:~/ruby/dns$ sudo ruby dns-watcher.rb
Mon Jan 22 14:01:01 GMT 2007 - BEGIN run.
Device 149.153.23.55 (to 149.153.1.5) looking for newsrss.bbc.co.uk.
Device 149.153.23.55 (to 149.153.1.5) looking for news.google.com.
Device 149.153.23.55 (to 149.153.1.5) looking for www.linuxjournal.com.
Device 149.153.23.55 (to 149.153.1.5) looking for www.rte.ie.
Device 149.153.23.55 (to 149.153.1.5) looking for www.linuxjournal.com.itcarlow.ie.
Device 149.153.23.55 (to 149.153.1.5) looking for www.linuxjournal.com.
Device 149.153.23.55 (to 149.153.1.5) looking for www.rte.ie.itcarlow.ie.
Device 149.153.23.55 (to 149.153.1.5) looking for www.rte.ie.
Device 149.153.23.55 (to 149.153.1.5) looking for news.google.com.
Device 149.153.23.55 (to 149.153.1.5) looking for newsrss.bbc.co.uk.
Device 149.153.23.55 (to 149.153.1.5) looking for www.google.com.
Device 149.153.23.55 (to 149.153.1.5) looking for www.google.com.
Device 149.153.23.55 (to 149.153.1.5) looking for www.google.ie.
Device 149.153.23.55 (to 149.153.1.5) looking for www.google.ie.
Device 149.153.23.55 (to 149.153.1.5) looking for ww.ibm.com.
Device 149.153.23.55 (to 149.153.1.5) looking for ww.ibm.com.itcarlow.ie.
Device 149.153.23.55 (to 149.153.1.5) looking for ww.ibm.com.
Device 149.153.23.55 (to 149.153.1.5) looking for ww.ibm.com.itcarlow.ie.
Device 149.153.23.55 (to 149.153.1.5) looking for ww.ibm.com.
Device 149.153.23.55 (to 149.153.1.5) looking for ww.ibm.com.itcarlow.ie.
Device 149.153.23.55 (to 149.153.1.5) looking for ww.ibm.com.
Device 149.153.23.55 (to 149.153.1.5) looking for ww.ibm.com.itcarlow.ie.
Device 149.153.23.55 (to 149.153.1.5) looking for www.ibm.com.
Device 149.153.23.55 (to 149.153.1.5) looking for www.ibm.com.
Device 149.153.23.55 (to 149.153.1.5) looking for data.coremetrics.com.
Mon Jan 22 14:01:42 GMT 2007 - END run.
barryp@pbtosh:~/ruby/dns$
```

Figure 1. Running dns-watcher.rb from the Command Line

To run the analyzer for a longer amount of time, change the NUMPACKETS constant to some value greater than 50. As shown in Figure 1, it took the analyzer just more than 40 seconds to process 50 DNS messages (on my PC, on my network segment—your mileage will vary). It is not unreasonable to assume that changing the constant value to something like 250 could result in several minutes of processing. Obviously, piping the output to a disk file or to less allows you to review any results at your leisure.

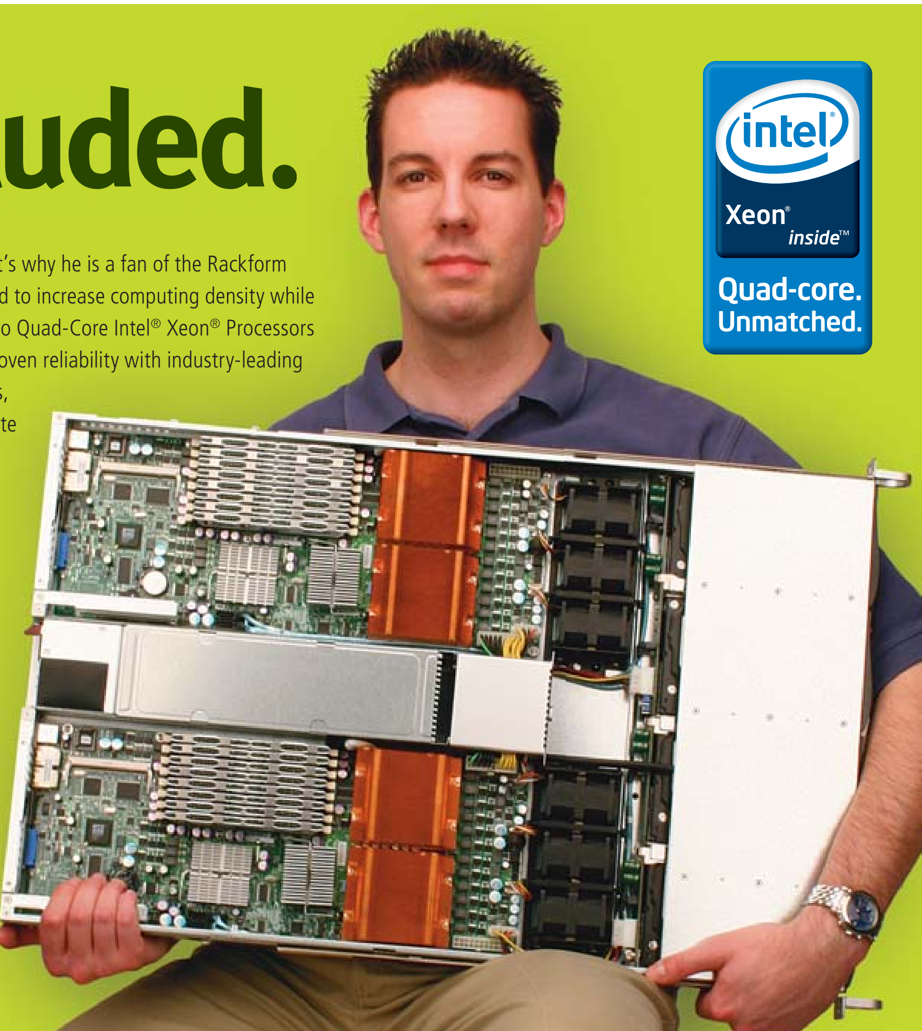
Creating a Web-Based Ethernet Analyzer

With my little analyzer up and running, I started thinking it would be cool if I could provide a Web-based interface to it. As every Web developer knows, long-running, server-bound processes and the Web tend not to go together, as there's nothing worse than waiting at a browser for long periods of time while such a process executes. During the years, a number of solutions to this problem have been proposed, which involve techniques that employ redirection, cookies, sessions and the like. Although such techniques work, I've always thought they were rather clunky, and I've been on the lookout for something more elegant. Having just completed Reuven M. Lerner's excellent series of *LJ* articles on Ajax programming [see the October, November and December 2006 issues of *LJ*], I wondered if I could combine my analyzer with an Ajax-enabled Web page, updating a part of the Web page with the output from the analyzer as and when it was generated.

My strategy is simple enough. I provide a starter Web page that starts the network analysis on the Web server as a backgrounded CGI process, and then redirects to another Web page that displays the results in an HTML text-area widget, updating the text area with the results from the network analysis. The little HTML Web page in Listing 2 gets things moving. All this Web page really does is provide a link that, when clicked, calls the startwatch.cgi script. The latter is itself straightforward CGI, written as a bash script. Here's the entire script:

```
#! /bin/sh

echo "Content-type: text/html"
echo ""

sudo /usr/bin/ruby /var/www/watcher/dns-watcher.rb \
         > /var/www/watcher/dns-watcher.log &

echo '<html><head>'
echo '<title>Fetching results ... </title>'
echo '<meta http-equiv="Refresh" content="1;'
echo 'URL=/watcher.html">'
echo '</head><body>Fetching results ... </body>'
echo '<html>'
```

The key line of script is the one that invokes Ruby and feeds the interpreter the dns-watcher.rb program, redirecting the latter's standard output to a file called dns-watcher.log. Note the trailing ampersand at the end of this command, which runs the analyzer as a background



Listing 2. A Simple HTML Web Page That Starts the Analyzer

```
<html>
  <head>
    <title>Start a new DNS Analysis</title>
  </head>

  <body>
    Click to
    <a href="/cgi-bin/startwatch.cgi">start</a>.
  </body>
</html>
```

Listing 3. The Network Analysis Results Web Page

```
<html>

<head>
  <title>Web-based DNS Watcher</title>

  <script language=javascript src="/js/dns-watcher.js">
  </script>
</head>

<body>

<h1>Web-based DNS Watcher</h1>

Here are the results of your DNS analysis:
<p>
<textarea name="watcherarea" cols="100"
           rows="20" id="watcherarea">
Waiting for results ...
</textarea>

<script>
startWatcher();
</script>

<p>Start
<a href="/startwatcher.html">another analysis</a>
(which stops this one).

</body>
</html>
```

process. The script continues by sending a sort HTML Web page to the browser that redirects to the analysis results page, called watcher.html, which is shown in Listing 3.

The results Web page loads in some JavaScript code (dns-watcher.js) within its header section, and then creates a simple HTML results page that contains an initially empty text-area widget called watcherarea. A call to the startWatcher JavaScript method occurs as soon as the browser loads the body section of the results Web page.

Listing 4 contains the dns-watcher.js code. A lot of what happens here has been covered by Reuven's excellent Ajax articles. The code starts by declaring some global variables that are used throughout the remainder of the code:

```
var capturing = false;
var matchEnd = new RegExp( "END run" );
var r = new getXMLHttpRequest();
```

The capturing boolean is set to true while the analyzer is capturing traffic, and to false otherwise. A regular expression is created to match against a string containing the words "END run". Finally, an Ajax request object is created with a call to the getXMLHttpRequest method, which is taken directly from Reuven's examples.

The startWatcher method starts the heavy lifting by calling the updateCaptureData method every 1.5 seconds and setting capturing to true:

```
function startWatcher() {
  setInterval( "updateCaptureData()", 1500 );
  capturing = true;
}
```

It is within the updateCaptureData method that the Ajax call occurs, with the request object being used to execute another CGI script that accesses the dns-watcher.log disk file and returns its contents. (Listing 5 contains the get_watcher_data.cgi script, which is written in Ruby.) Once the CGI script has been invoked on the Web server, a call to displayCapture occurs:

```
function updateCaptureData() {

  if (capturing) {
    r.open( "GET",
             "/cgi-bin/get_watcher_data.cgi",
             false );
    r.send( null );

    displayCaptureData();
  }
}
```

The displayCaptureData method is adapted from Reuven's code and processes the results of the Ajax call, which are available from the request object. These are used to update the watcherarea text-area widget within the results Web page:

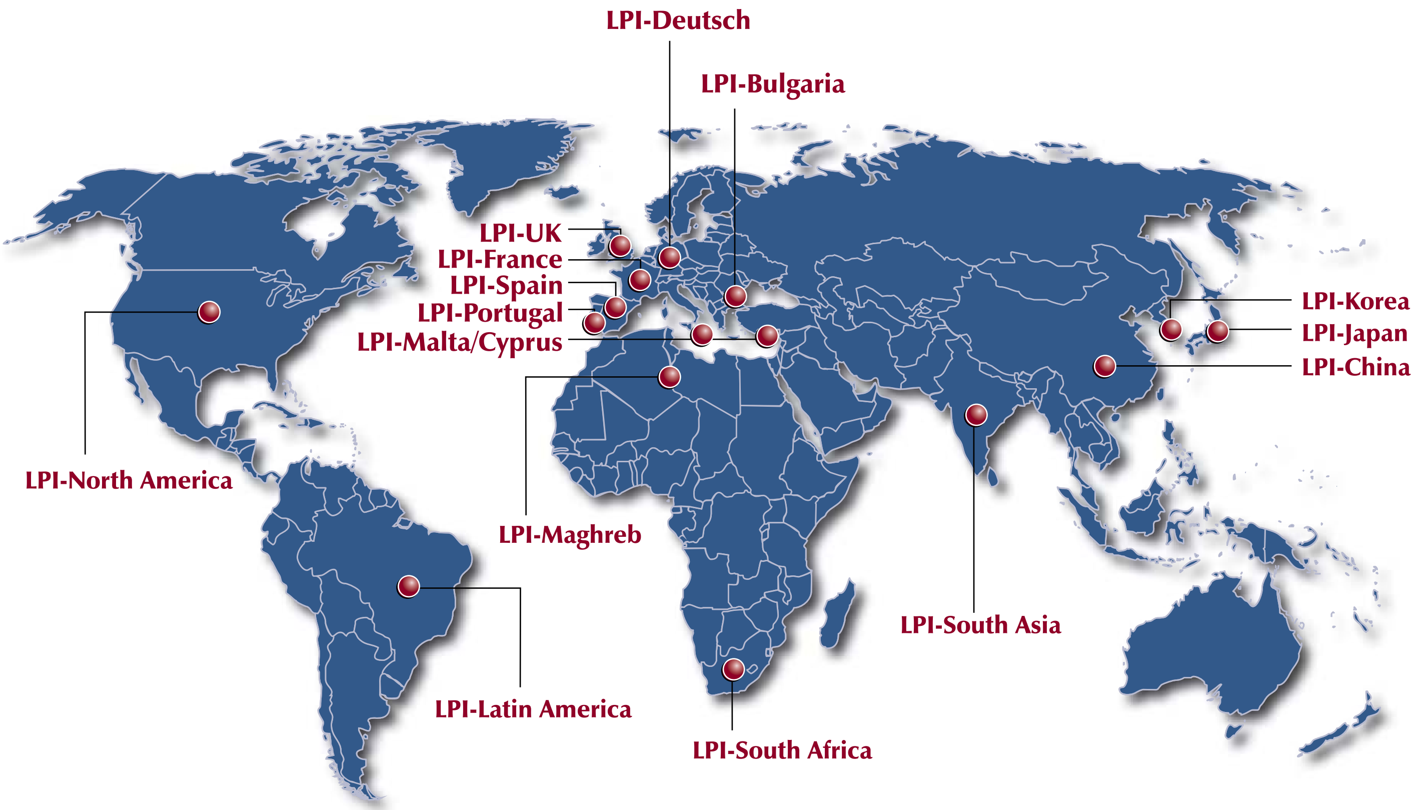
```
te.value = r.responseText;
```

Note the use of the following line of JavaScript to scroll the text area to the bottom of the results:

```
te.scrollTop = te.scrollHeight;
```

And, finally, note that the displayCaptureData method sets the capturing boolean to false as soon as a line that matches the regular expression appears within the data coming from the Ajax request (see

Growing a World of Linux Professionals



We at the Linux Professional Institute believe the best way to spread the adoption of Linux and Open Source software is to grow a world wide supply of talented, qualified and accredited IT professionals.

We realize the importance of providing a global standard of measurement. To assist in this effort, we are launching a Regional Enablement Initiative to ensure we understand, nurture and support the needs of the enterprise, governments, educational institutions and individual contributors around the globe.

We can only achieve this through a network of local "on the ground" partner organizations. Partners who know the sector and understand the needs of the IT work force. Through this active policy of Regional Enablement we are seeking local partners and assisting them in their efforts to promote Linux and Open Source professionalism.

We encourage you to contact our new regional partners listed above.

Together we are growing a world of Linux Professionals.

Listing 4. The Ajax-Enabled JavaScript Code

```
var capturing = false;
var matchEnd = new RegExp( "END run" );
var r = new getXMLHttpRequest();

function startWatcher() {
  setInterval( "updateCaptureData()", 1500 );
  capturing = true;
}

function getXMLHttpRequest() {
  try {
    return new ActiveXObject("Msxml2.XMLHTTP");}
  catch(e) {};
  try {
    return new ActiveXObject("Microsoft.XMLHTTP");}
  catch(e) {}
  try {
    return new XMLHttpRequest(); }
  catch(e) {};

  return null;
}

function updateCaptureData() {

  if (capturing) {
    r.open( "GET",
            "/cgi-bin/get_watcher_data.cgi",
            false );
    r.send( null );

    displayCaptureData();
  }
}

function displayCaptureData() {
  var te = document.getElementById("watcherarea");

  if ( r.readyState == 4 ) {
    if ( r.status == 200 ) {

      te.value = r.responseText;
      te.scrollTop = te.scrollHeight;

      if ( matchEnd.test( te.value ) ) {
        capturing = false;
      }
    }
    else
    {
      te.value =
        "Web-based DNS Analysis unavailable.";
    }
  }
}
```

Listing 5. A Simple CGI to Retrieve the Captured Data

```
#! /usr/bin/ruby -w

puts "Content-type: text/plain\n\n"

IO.foreach("/var/www/watcher/dns-watcher.log") do |l|
  puts l
end
```

Figures 1 and 2 to convince yourself that this in fact matches at the end of the network capture):

```
if ( matchEnd.test( te.value ) ) {
  capturing = false;
}
```

This check is very important. Without it, the Web browser continues to send an Ajax request to the server every 1.5 seconds for as long as the watcher.html results page is displayed within the browser, even after the analyzer has finished and isn't generating any more data. With this check in the code, the Ajax behavior is switched off, reducing the load on the Web server (and keeping the Apache2 access log from quickly growing large).

To deploy my solution, I created a simple shell script to copy the required components into the appropriate directory locations on my Web server (which is Apache2 on Ubuntu):

```
sudo cp watcher.html /var/www/
sudo cp startwatcher.html /var/www/
sudo cp dns-watcher.js /var/www/js/
sudo cp dns-watcher.rb /var/www/watcher/
sudo cp get_watcher_data.cgi /usr/lib/cgi-bin/
sudo cp startwatch.cgi /usr/lib/cgi-bin/
```

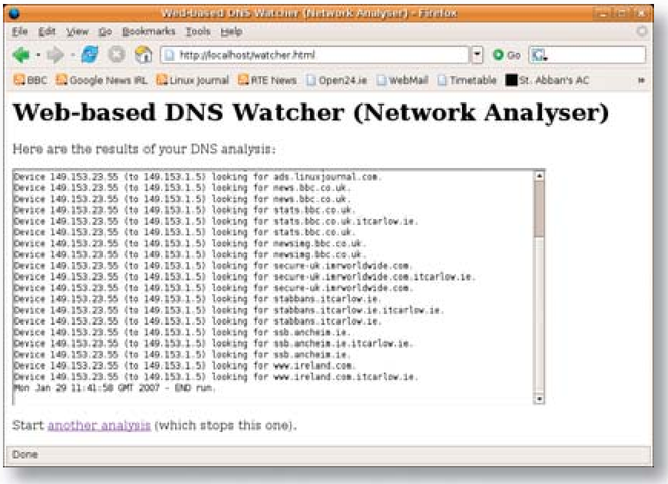
These directory locations may not match those of your Apache2 installation, so adjust accordingly. You also may need to create the js and watcher directories. And, of course, make sure the CGIs have their executable bit set.

Running the Web-Based Network Analyzer

One final wrinkle is that the dns-watcher.rb program needs to be executed with root privilege, in order to switch the Web server's NIC into promiscuous mode. As would be expected, Apache2 does not, by default, execute CGI scripts as a root privilege, and for good reason. To get my Web-based analyzer to work, I added the following line to my /etc/sudoers file:

```
%www-data ALL=(root) NOPASSWD: /usr/bin/ruby
```

This allows the www-data user, which executes Apache2, to execute Ruby with root privilege, as it is the Ruby interpreter that executes the


Web-based DNS Watcher (Network Analyser) - Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/watcher.html
Go
BBC
Google News IRL
Linux Journal
RTE News
Open24.ie
WebMail
Timetable
St. Abban's AC
Done

Web-based DNS Watcher (Network Analyser)

Here are the results of your DNS analysis:

```
Device 149.153.23.55 (to 149.153.1.5) looking for ads.linuxjournal.com.
Device 149.153.23.55 (to 149.153.1.5) looking for news.bbc.co.uk.
Device 149.153.23.55 (to 149.153.1.5) looking for news.bbc.co.uk.
Device 149.153.23.55 (to 149.153.1.5) looking for stats.bbc.co.uk.
Device 149.153.23.55 (to 149.153.1.5) looking for stats.bbc.co.uk.itcarlow.ie.
Device 149.153.23.55 (to 149.153.1.5) looking for stats.bbc.co.uk.
Device 149.153.23.55 (to 149.153.1.5) looking for newsimg.bbc.co.uk.
Device 149.153.23.55 (to 149.153.1.5) looking for newsimg.bbc.co.uk.
Device 149.153.23.55 (to 149.153.1.5) looking for secure-uk.imrworldwide.com.
Device 149.153.23.55 (to 149.153.1.5) looking for secure-uk.imrworldwide.com.itcarlow.ie.
Device 149.153.23.55 (to 149.153.1.5) looking for secure-uk.imrworldwide.com.
Device 149.153.23.55 (to 149.153.1.5) looking for stabbans.itcarlow.ie.
Device 149.153.23.55 (to 149.153.1.5) looking for stabbans.itcarlow.ie.itcarlow.ie.
Device 149.153.23.55 (to 149.153.1.5) looking for stabbans.itcarlow.ie.
Device 149.153.23.55 (to 149.153.1.5) looking for ssb.ancheim.ie.
Device 149.153.23.55 (to 149.153.1.5) looking for ssb.ancheim.ie.itcarlow.ie.
Device 149.153.23.55 (to 149.153.1.5) looking for ssb.ancheim.ie.
Device 149.153.23.55 (to 149.153.1.5) looking for www.ireland.com.
Device 149.153.23.55 (to 149.153.1.5) looking for www.ireland.com.itcarlow.ie.
Mon Jan 29 11:41:58 GMT 2007 - END run.
```

Start another analysis (which stops this one).

Figure 2. Running dns-watcher.rb from the Web

dns-watcher.rb code on behalf of Apache2. Such a situation may not be acceptable to you—due to the security concerns raised—and I'd be interested to know if any reader has a solution that allows me to execute the analyzer with root privilege more safely.

Figure 2 shows the results of a Web-based network analysis. The long-running, server-bound process is started by the Web server, runs in the background and—as results are generated—any and all output appears within the Web-based front end. Thanks to Ajax, the user's experience closely matches that of the command-line execution of the same program—as soon as data is ready, it's displayed. Adapting my solution to other uses is not difficult; all that's required is a mechanism to redirect some long-running, server-bound process' output to a file, and then access the file's contents via a CGI script that executes as a result of a single Ajax call. As I hope I've demonstrated, Ruby and Ajax make for a clean solution to this particular Web development pattern.■

Paul Barry (paul.barry@itcarlow.ie) lectures at the Institute of Technology, Carlow in Ireland. Find out more about the stuff he does at glasnost.itcarlow.ie/~barryp.

Resources

Paul's Ruby Tutorial: **glasnost.itcarlow.ie/~barryp/ruby-tut.html**

tcpdump/libpcap: **www.tcpdump.org**

Ruby's libpcap Library: **raa.ruby-lang.org/project/pcap**

Ruby Net::DNS Page on RubyForge: **rubyforge.org/projects/net-dns**

Ethereal: **www.ethereal.com**

Wireshark: **www.wireshark.org**

The "who is doing what?" Perl Script:
www.theperlreview.com/Issues/v0i6.shtml

Source Code for This Article:
ftp.linuxjournal.com/pub/lj/issue157/9614.tgz

Advertiser Index

**For advertising information, please contact our sales department at 1-713-344-1956 ext. 2 or ads@linuxjournal.com.
www.linuxjournal.com/advertising**

Advertiser	Page #	Advertiser	Page #
Aberdeen, LLC www.aberdeeninc.com	11	MBX www.mbx.com	19
AML www.amltd.com	21	Microway, Inc. www.microway.com	C4, 39
APPRO HPC Solutions appro.com	C2	Mikro Tik www.routerboard.com	25
ASA Computers www.asacomputers.com	31, 49	Open Source Systems, Inc. www.opensourcesystems.com	5
Avocent Corporation www.avocent.com/special	1	OSBC www.osbc.com	63
Cari.net www.cari.net	95	Pogo Linux www.pogolinux.com	15
Coraid, Inc. www.coraid.com	13	Polywell Computers, Inc. www.polywell.com	29
Coyote Point www.coyotepoint.com	3	The Portland Group www.pgroup.com	37
EGI Hosting www.egihosting.com	85	Rackspace Managed Hosting www.rackspace.com	C3
Emac, Inc. www.emacinc.com	59	R Cubed Technologies www.rcubedtech.com	23
EmperorLinux www.emperorlinux.com	35	SYSGO AG www.sysgo.com	17
FairCom www.faircom.com	75	Servers Direct www.serversdirect.com	27
Genstor Systems, Inc. www.genstor.com	45	Silicon Mechanics www.siliconmechanics.com	73, 77
Hurricane Electric www.he.net	67	Supermicro www.supermicro.com	53
Iron Systems www.ironsystems.com	61	Technologic Systems www.embeddedx86.com	10
ISPCON www.ispcon.com	83	TotalView Technologies (formerly Etnus) www.totalviewtech.com	9
LPI www.lpi.org	79	Tyan Computer USA www.tyan.com	7
Linux Journal www.linuxjournal.com	6, 93	Usenix Association www.usenix.org/usenix07/lj	91
LinuxWorld Conference & Expo www.it360.ca	51		

Protecting SSH Servers with Single Packet Authorization

Lock down access to SSH with Single Packet Authorization. MICHAEL RASH

Last month, in the first of a two-part series, I described the theory behind the next generation in passive authentication technologies called Single Packet Authorization (SPA). This article gets away from theory and concentrates on the practical application of SPA with fwknop and iptables to protect SSHD from reconnaissance and attack. With this setup on a Linux system, no one will be able to tell that SSHD is even listening under an nmap scan, and only authenticated and authorized clients will be able to communicate with SSHD.

To begin, we require some information about configuration and network architecture. This article assumes you have installed the latest version of fwknop (1.0.1 at the time of this writing) on the same system where SSHD and iptables are running. You can download fwknop from **www.cipherdyne.org/fwknop** and install either from the source tar archive by running the install.pl script or via the RPM for RPM-based Linux distributions.

Network Architecture

The basic network depicted in Figure 1 illustrates our setup. The fwknop client is executed on the host labeled spa_client (15.1.1.1), and the fwknop server (along with iptables) runs on the system labeled spa_server (16.2.2.2). A malicious system is labeled attacker (18.3.3.3), which is able to sniff all traffic between the spa_client and spa_server systems.

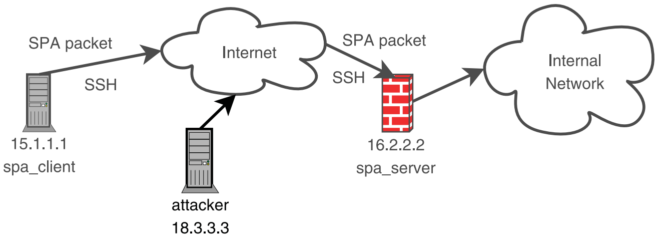


Figure 1. Sample Scenario Where You Use SPA to Protect SSH Communications

Default-Drop iptables Policy

The spa_client system has the IP address 15.1.1.1, and the spa_server system has the IP address 16.2.2.2. On the spa_server system, iptables is configured to provide basic connectivity services for the internal network (192.168.10.0/24) and to log and drop all attempts (via the iptables LOG and DROP targets) from the external network to connect to any service on the firewall itself. This policy is quite simplistic, and it is meant to show only that the firewall does not advertise any services (including SSHD) under an nmap scan. Any serious deployment of iptables for a real network would be significantly more complicated. One important feature to note, however, is that the connection tracking facilities provided by Netfilter are used to keep state in the iptables policy. The end result is that connections initiated through the firewall (via the FORWARD chain) and to the firewall (via the INPUT chain) remain open without additional ACCEPT rules to allow packets required to keep the connections established (such as TCP acknowledgements and the like). The iptables policy is built with the following basic firewall.sh script:

```
[spa_server]# cat firewall.sh
#!/bin/sh
IPTABLES=/sbin/iptables
$IPTABLES -F
$IPTABLES -F -t nat
$IPTABLES -X
$IPTABLES -A INPUT -m state --state
 ➥ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A FORWARD -m state --state
 ➥ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -t nat -A POSTROUTING -s
 ➥192.168.10.0/24 -o eth0 -j MASQUERADE
$IPTABLES -A INPUT -i ! lo -j LOG --log-prefix
 ➥"DROP "
$IPTABLES -A INPUT -i ! lo -j DROP
$IPTABLES -A FORWARD -i ! lo -j LOG --log-prefix
 ➥"DROP "
$IPTABLES -A FORWARD -i ! lo -j DROP
echo 1 > /proc/sys/net/ipv4/ip_forward
echo "[+] iptables policy activated"
exit
[spa_server]# ./firewall.sh
[+] iptables policy activated
```

With iptables active, it is time to see what remote access we might have. From the spa_client system, we use nmap to see if SSHD is accessible on the spa_server system:

```
[spa_client]$  nmap -P0 -sT -p 22 16.2.2.2

Starting Nmap 4.01 ( http://www.insecure.org/nmap/ )
at 2007-02-09 23:55 EST
Interesting ports on 16.2.2.2:
PORT   STATE    SERVICE
22/tcp filtered ssh

Nmap finished: 1 IP address (1 host up) scanned in
12.009 seconds
```

As expected, iptables is blocking all attempts to communicate with SSHD, and the remaining ports (both TCP and UDP) are similarly protected by the iptables policy. It does not matter if an attacker has a zero-day exploit for the particular version of OpenSSH that is deployed on the spa_server system; all attempts to communicate up the stack are being blocked by iptables.

fwknop SPA Configuration

Confident that iptables is protecting the local network with a Draconian stance, it is time to configure the fwknop server dæmon (fwknopd) on the spa_server system. The file /etc/fwknop/fwknop.conf controls important configuration parameters, such as the interface on which fwknopd sniffs traffic via libpcap, the e-mail address(es) to which fwknopd sends informational alerts and the pcap filter statement designed to sniff SPA packets off the wire. By default, fwknop sends SPA packets over UDP port 62201, so the pcap filter statement in /etc/fwknop/fwknop.conf is set to `udp port 62201` by default. However, SPA packets can be sent over any port and protocol (even over ICMP), but the filter statement would need to be updated to handle SPA communications over other port/protocols. More information can be found in the fwknop man page. Although the defaults in this file usually make sense for most deployments, you may need to tweak the PCAP_INTF and EMAIL_ADDRESSES variables for your particular setup.

The /etc/fwknop/access.conf file is the most important fwknopd configuration file—it manages the encryption keys and access control rights used to validate SPA packets from fwknop clients. The following access.conf file is used for the remainder of this article:

```
[spa_server]# cat /etc/fwknop/access.conf
SOURCE: ANY;
OPEN_PORTS: tcp/22;
FW_ACCESS_TIMEOUT: 30;
KEY: LJ07p2rbga;
GPG_DECRYPT_ID: ABCD1234;
GPG_DECRYPT_PW: p2atc1l30p;
GPG_REMOTE_ID: 5678DEFG;
GPG_HOME_DIR: /root/.gnupg;
```

The SOURCE variable defines the IP addresses from which fwknopd accepts SPA packets. The value ANY shown above is a wild card to examine SPA packets from any IP address, but it can be restricted to specific IP addresses or subnets, and comma-separated lists are supported (for example, 192.168.10.0/24, 15.1.1.1). The OPEN_PORTS variable informs fwknopd about the set of ports that should be opened upon receiving a valid SPA packet; in this case, fwknopd will open TCP port 22.

Although not shown above, fwknopd can be configured to allow the fwknop client to dictate the set of ports to open by including the PERMIT_CLIENT_PORTS variable and setting it to Y. FW_ACCESS_TIMEOUT specifies the length of time that an ACCEPT rule is added to the iptables policy to allow the traffic defined by the OPEN_PORTS variable. Because the iptables policy in the firewall.sh script above makes use of the connection tracking capabilities provided by Netfilter, an SSH connection will remain established after the initial ACCEPT rule is deleted by fwknopd.

The remaining variables define parameters for the encryption and decryption of SPA packets. This article illustrates the usage of both symmetric and asymmetric ciphers, but only one encryption style is required by fwknop.

All of the GPG_* variables can be omitted if there is a KEY variable and vice versa. The KEY variable defines a shared key between the fwknop client and fwknopd server. This key is used to encrypt/decrypt the SPA packet with the Rijndael symmetric block cipher (see Resources). For asymmetric encryption, GPG_DECRYPT_ID defines the local fwknopd server GnuPG key ID. This key is used by the fwknop client to encrypt SPA packets via an encryption algorithm supported by GnuPG (such as the ElGamal cipher).

GPG_DECRYPT_PW is the decryption password associated with the fwknop server key. Because this password is placed within the access.conf file in clear text, it is not recommended to use a valuable GnuPG key for the server; a dedicated key should be generated for the purpose of decrypting SPA packets. The fwknop clients sign SPA packets with a GnuPG key on the local key ring, and the password is supplied by the user from the command line and never stored within a file (as we will see below). Hence, any GnuPG key can be used by the fwknop client; even a valuable key used for encrypting sensitive e-mail communications, for example.

The GPG_REMOTE_ID variable defines a list of key IDs that the fwknopd server will accept. Any SPA packet encrypted with the

fwknopd server public key must be signed with a private key specified by the GPG_REMOTE_ID variable. This allows fwknopd to restrict the set of people who can gain access to a protected service (SSHD in our case) via a cryptographically strong mechanism. Instructions for creating GnuPG keys for use with fwknop can be found at **www.cipherdyne.org/fwknop/docs/gpghowto.html**.

With the /etc/fwknop/access.conf file built, it is time to start fwknopd on the spa_server system and put fwknop to work for us:

```
[spa_server]# /etc/init.d/fwknop start
 * Starting fwknop ...                          [ ok ]
```

SPA via Symmetric Encryption

On the spa_client system, we use fwknop to build an SPA packet encrypted via Rijndael and send it on its way to the spa_server system. We want access to SSHD, and the -A argument below encodes the desired access within the SPA packet. The -w argument resolves the IP address of the client system by querying http://www.whatismyip.com (this is useful if the fwknop client is behind a NAT device), the -k argument is the IP address of the destination SPA server, and -v runs in verbose mode so we can view the raw packet data:

```
[spa_client]$ fwknop -A tcp/22 -w -k 16.2.2.2 -v
[+] Starting fwknop in client mode.
    Resolving external IP via: http://www.whatismyip.com/
    Got external address: 15.1.1.1

[+] Enter an encryption key. This key must match a key
    in the file /etc/fwknop/access.conf on the remote system.

Encryption Key:

[+] Building encrypted single-packet authorization
    (SPA) message...
[+] Packet fields:

        Random data: 7764880827899123
        Username:    mbr
        Timestamp:   1171133745
        Version:     1.0.1
        Action:      1 (access mode)
        Access:      15.1.1.1,tcp/22
        MD5 sum:     yzxKgnAxwUA5M2YhI8NTFQ

[+] Packet data:

U2FsdGVkX1+BvzxXj5Zv6gvfCFXwJ+iJGKPqe2whdYzyigkerSp \
2WtvON/xTd8t6V6saxbg1v4zsK+YNt53BE8EInxVCgpD7y/gEBI \
g8sd+AvU1ekQh9vwJJduseVxDxjmAHx3oNnClo2wckBqd8zA

[+] Sending 150 byte message to 16.2.2.2 over udp/62201...
```

As you can see from the Packet data section above, the SPA packet is a completely unintelligible blob of encrypted data. On the spa_server system, the following syslog message is generated indicating that an ACCEPT rule has been added for the source IP (15.1.1.1) that generated the SPA packet. Note that the source IP is put within the SPA packet by the fwknop client. In this case, the SPA packet was not spoofed, so the real source address and the source address embedded in the SPA packet match. SPA packets can be spoofed by fwknop with the `--Spoof-src` command-line argument (requires root):

```
Feb 10 13:55:44 spa_server fwknopd: received valid Rijndael \
encrypted packet from: 15.1.1.1, remote user: mbr
Feb 10 13:55:44 spa_server fwknopd: adding FWKNOP_INPUT ACCEPT \
rule for 15.1.1.1 -> tcp/22 (30 seconds)
```

So, for 30 seconds after sending the SPA packet, the iptables policy on the spa_server allows the spa_client system to establish an SSH session:

```
[spa_client]$ ssh -l mbr 16.2.2.2
mbr@spa_server's password:
```

After 30 seconds has expired, knoptm (a dæmon responsible for deleting iptables rules added by fwknopd to the iptables policy) deletes the ACCEPT rule and writes the following messages to syslog:

```
Feb 10 13:52:17 spa_server knoptm: removed iptables \
FWKNOP_INPUT ACCEPT rule for 15.1.1.1 -> tcp/22, \
30 second timeout exceeded
```

Our SSH session remains established after the ACCEPT rule is deleted because of the state tracking rules in the iptables policy (see the firewall.sh script above). These rules allow packets that are part of an established TCP connection to pass unimpeded.

SPA via Asymmetric Encryption

To use GnuPG to encrypt and sign an SPA packet, you can execute the fwknop command below. In this case, the key ID of the fwknopd server is specified on the command line with the --gpg-recipient argument, and the key ID used to sign the SPA packet is given with the --gpg-signing-key argument (the output below has been abbreviated):

```
[spa_client]$ fwknop -A tcp/22 --gpg-recipient ABCD1234 \
--gpg-signing-key 5678DEFG -w -k 16.2.2.2

[+] Sending 1010 byte message to 16.2.2.2 over udp/62201
```

As you can see, the length of the application portion of the SPA packet has increased to more than 1,000 bytes, whereas it was only 150 bytes for the Rijndael example. This is because the key length of GnuPG keys (in this case 2,048 bits) and the characteristics of asymmetric ciphers tend to inflate the size of small chunks of data after being encrypted. There is no strict correspondence between the size of clear-text and cipher-text data as in block ciphers such as Rijndael.

Again, on the spa_server system, fwknop adds the ACCEPT rule for us. This time fwknopd reports that the SPA packet is encrypted with GnuPG, and that a valid signature for the required key ID 5678DEFG is found:

```
Feb 10 14:38:26 spa_server fwknopd: received valid GnuPG
encrypted packet (signed with required key ID: "5678DEFG")
```

```
from: 15.1.1.1, remote user: mbr
Feb 10 14:38:26 spa_server fwknopd: adding
FWKNOP_INPUT ACCEPT rule for 15.1.1.1 -> tcp/22 (30 seconds)
```

Thwarting a Replay Attack

Suppose that the SPA packet from the first example above was sniffed off the wire en route by a crafty individual on the system labeled attacker in the network diagram in Figure 1. The SPA packet always can be placed back on the wire in an effort to gain the same access as the original packet—this is known as a replay attack. There are several ways to acquire the packet data and replay it. One of the most common is to use tcpdump to write a pcap file (in this case `tcpdump -i eth0 -l -nn -s 0 -w SPA.pcap port 62201` would work) and then use tcpreplay (see **tcpreplay.synfin.net/trac**) to copy the SPA packet back onto the wire. Another method, after the packet has been captured, is to use the echo command along with netcat:

```
[attacker]$ echo "U2FsdGVkX1+BvzxXj5Zv6gvfCFXwJ+iJGKP \
qe2whdYzyigkerSp2WtvON/xTd8t6V6saxbg1v4zsK+YNt53BE8EI \
nxVCgpD7y/gEBIg8sd+AvU1ekQh9vwJJduseVx \
DxjmAHx3oNnClo2wckBqd8zA" |nc -u 16.2.2.2 62201
```

On the fwknopd server, the duplicate SPA packet is monitored, but because the MD5 sum matches that of the original SPA packet, no access is granted, and the following message is written to syslog on the spa_server system:

```
Feb 10 14:14:24 spa_server fwknopd: attempted \
message replay from: 18.3.3.3
```

Conclusion

Single Packet Authorization provides an additional layer of security for services such as SSHD, and this layer strikes at the first step that an attacker must accomplish when trying to compromise a system: reconnaissance. By using iptables in a default-drop stance and fwknop to sniff the wire for specially constructed (that is, encrypted and non-replayed) packets, it is difficult even to tell that a service is listening, let alone communicate with it. The end result is that it is significantly harder to exploit any vulnerabilities a protected service might have.■

Michael Rash holds a Master's degree in applied mathematics with a concentration in computer security from the University of Maryland. Michael is the founder of cipherdyne.org, a Web site dedicated to open-source security software for Linux systems, and he works as Security Architect on the Dragon Intrusion Detection System for Enterasys Networks. He is the author of the upcoming book *Linux Firewalls: Attack Detection and Response*, published by No Starch Press.

Resources

fwknop: **www.cipherdyne.org/fwknop**

An excellent source of additional theoretical information about both port knocking and Single Packet Authorization can be found in Sebastien Jeanquier's Master's thesis at the Royal Holloway College, University of London. The thesis can be downloaded from **web.mac.com/s.j**, and it includes an excellent argument for why SPA is not "security through obscurity".

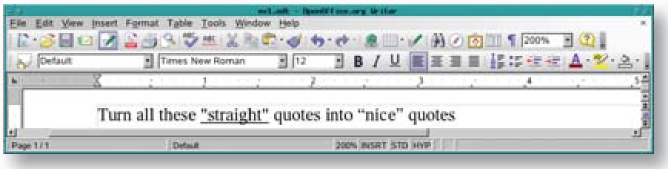
The Rijndael cipher was selected in 2001 for the Advanced Encryption Standard (AES) as the successor to the aging Data Encryption Standard (DES). A good writeup can be found at **en.wikipedia.org/wiki/Advanced_Encryption_Standard**.

GnuPG is the GNU Privacy Guard, and is an open-source implementation of the OpenPGP standard. More information can be found at **www.gnupg.org**.

OpenOffice.org ODF, Python and XML

Combine Python with the open format of ODF files to manipulate fine details. COLLIN PARK

My wife is a writer, which today means she uses a word processing program. It's a sophisticated, powerful program—OpenOffice.org Writer—but occasionally it won't do something that she wants it to do. In this article, we take a look at the structure of OpenDocument Format (ODF) files and see how Python, with its XML libraries, can help. Figure 1 shows an example.



Turn all these "straight" quotes into “nice” quotes

Figure 1. Converting Quotation Marks

It's not hard to convert quotation marks on a few paragraphs by hand—or even on a few pages, if I'm doing it only once. But having to repeat such manual operations on subsequent revisions becomes tedious, especially on a longer document, such as a poetry collection or novel. (We might have to repeat these operations after importing plain text from an e-mail message, for example.)

Fortunately, ODF is *open*, so we should be able to manipulate the file contents outside the word processing program.

Let's see if we can do that manually, just to make sure we know what we're doing. Once we can do that, we'll create a script to do some more ambitious things with the document.

Cracking the OpenDocument Format—A Simple Example

I read somewhere that an ODF file is a zip archive of XML files. So, let's see if it really is one—and if so, what's inside:

```
% unzip -l ex1.odt
Archive:  ex1.odt
  Length     Date   Time    Name
 --------    ----   ----    ----
       39  11-15-06 01:55   mimetype
        0  11-15-06 01:55   Configurations2/statusbar/
        0  11-15-06 01:55   Configurations2/accelerator/current.xml
        0  11-15-06 01:55   Configurations2/floater/
        0  11-15-06 01:55   Configurations2/popupmenu/
        0  11-15-06 01:55   Configurations2/progressbar/
        0  11-15-06 01:55   Configurations2/menubar/
        0  11-15-06 01:55   Configurations2/toolbar/
        0  11-15-06 01:55   Configurations2/images/Bitmaps/
        0  11-15-06 01:55   Pictures/
     2872  11-15-06 01:55   content.xml
     9786  11-15-06 01:55   styles.xml
     1109  11-15-06 01:55   meta.xml
      878  11-15-06 01:55   Thumbnails/thumbnail.png
     6611  11-15-06 01:55   settings.xml
     2037  11-15-06 01:55   META-INF/manifest.xml
 --------                   -------
    23332                   16 files
%
```

Good news—it *is* a zip archive.

So, the plan is this: unpack it, modify a file (or files) and pack everything back up again. We'll pack up files in the same order, just in case it matters. So, we need to save the file list.

The listing from running unzip has that file list, along with some other stuff. Let's select only the lines that have filenames (in this case, the lines with a : followed by digits) and print only the filenames. A single command to sed does that:

```
% unzip -l ex1.odt | sed -n '/:[0-9][0-9]/s|^.*:.. *||p'
mimetype
Configurations2/statusbar/
Configurations2/accelerator/current.xml
Configurations2/floater/
Configurations2/popupmenu/
Configurations2/progressbar/
Configurations2/menubar/
Configurations2/toolbar/
Configurations2/images/Bitmaps/
Pictures/
content.xml
styles.xml
meta.xml
Thumbnails/thumbnail.png
settings.xml
META-INF/manifest.xml
%
```

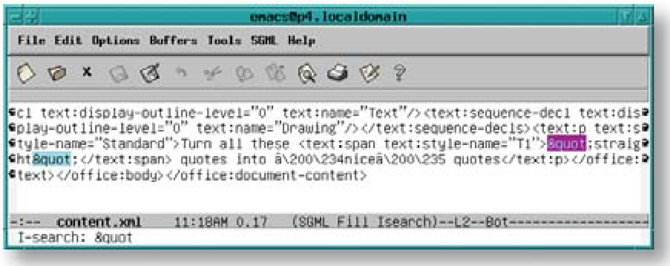
Looks good. Let's save the list in a shell variable—we'll use F (for files):

```
% F=$(unzip -l ex1.odt | sed -n '/:[0-9][0-9]/s|^.*:.. *||p')
```

With that settled, the next question is, which file to modify? To find out, let's find the file or files containing the word quotes, which appeared in the document. We'll unpack ex1.odt into an empty directory and ask grep, remembering to check files in subdirectories as well:

```
% cd TMP
% unzip -q ~/oo/ex1.odt
% find . -type f | xargs grep -l quote
./content.xml
%
```

Okay, content.xml is it. Text editors provide one way to manipulate content.xml, so let's give that a try. The relevant part looked like Figure 2 in Emacs.

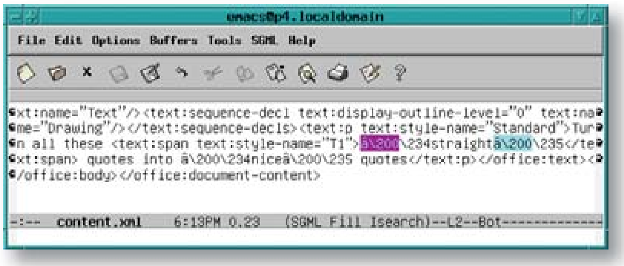


```
cl text:display-outline-level="0" text:name="Text"/><text:sequence-decl text:dis
play-outline-level="0" text:name="Drawing"/></text:sequence-decls><text:p text:s
tyle-name="Standard">Turn all these <text:span text:style-name="T1">&quot;straig
ht&quot;</text:span> quotes into â\200\234niceâ\200\235 quotes</text:p></office:
text></office:body></office:document-content>

-:--  content.xml     11:18AM 0.17    (SGML Fill Isearch)--L2--Bot------------------
 I-search: &quot
```

Figure 2. Editing XML in Emacs

The two occurrences of " (partially highlighted in Figure 2) represent the straight quotation marks.

I changed the straight quotes to the appropriate curly or smart quotes (found on either side of the word nice), as shown in Figure 3. The changed areas are, again, partially highlighted.

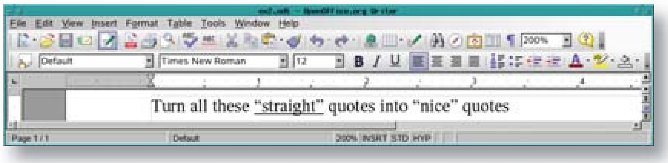


```
xt:name="Text"/><text:sequence-decl text:display-outline-level="0" text:na
me="Drawing"/></text:sequence-decls><text:p text:style-name="Standard">Tur
n all these <text:span text:style-name="T1">â\200\234straightâ\200\235</te
xt:span> quotes into â\200\234niceâ\200\235 quotes</text:p></office:text><
/office:body></office:document-content>

-:--  content.xml     6:13PM 0.23    (SGML Fill Isearch)--L2--Bot-------------
```

Figure 3. Edited XML with Smart Quotes

With that done, let's zip the files (the list saved in $F) to create ex2.odt, and see what OpenOffice.org Writer thinks about it:

```
% zip -q ~/oo/ex2.odt $F
% oowriter ~/oo/ex2.odt
```


ex2.odt - OpenOffice.org Writer
File Edit View Insert Format Table Tools Window Help
Default
Times New Roman
12
Page 1 / 1
Default
200% INSRT STD HYP

Turn all these “straight” quotes into “nice” quotes

Figure 4. Writer Recognizes the New Quotes

It worked (Figure 4)! The formerly straight quotes around the word straight are now curly quotes, and they're even curled in the right direction. So, to review what we've done so far:

- Created a list of the files in ex1.odt (saving it in $F).
- Unpacked ex1.odt.
- Made a simple change, manually, in content.xml.
- Created ex2.odt (using $F).
- Validated ex2.odt using OpenOffice.org Writer.

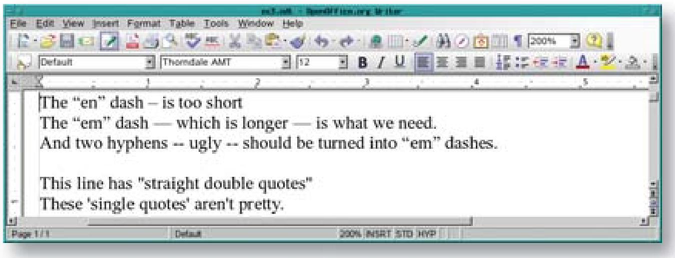
A Real-Life Example

That exercise proved the concept, so now we can get to work. My wife's poetry book was about 60 pages long, and it needed these issues addressed:

1. Those straight quotes, which came from plain-text e-mail messages or other word processors.
2. Apostrophes (or single quotes), which also were straight rather than curled the right way.
3. Double hyphens and shorter dashes (the en dash), which should all be changed into the longer em dash.

OpenOffice.org Writer has keystroke sequences for creating the en dash as well as the longer em dash. Sometimes the wrong sequence was typed, so an en dash appeared instead of the desired em dash. Plain text imported from e-mail messages sometimes had double hyphens (that is, --).

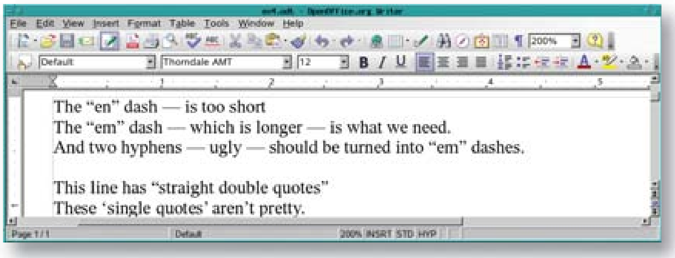
Concretely, we want to transform what's shown in Figure 5 into what's shown in Figure 6.


ex3.odt - OpenOffice.org Writer
File Edit View Insert Format Table Tools Window Help
Default
Thorndale AMT
12
Page 1 / 1
Default
200% INSRT STD HYP

The “en” dash – is too short
The “em” dash — which is longer — is what we need.
And two hyphens -- ugly -- should be turned into “em” dashes.

This line has "straight double quotes"
These 'single quotes' aren't pretty.

Figure 5. Before...


ex4.odt - OpenOffice.org Writer
File Edit View Insert Format Table Tools Window Help
Default
Thorndale AMT
12
Page 1 / 1
Default
200% INSRT STD HYP

The “en” dash — is too short
The “em” dash — which is longer — is what we need.
And two hyphens — ugly — should be turned into “em” dashes.

This line has “straight double quotes”
These ‘single quotes’ aren’t pretty.

Figure 6. ...and After

Let's develop the automated script in two pieces, and let's do it top-down. The top layer will create a temporary directory, unpack the original document and then run the bottom layer, a program (designated fixit.py) to modify content.xml. Afterward, it will pack up the files into the new document and clean up.

The Top Layer: a Shell Script

I want to use the highest-level language reasonable for each task; for this top layer, that's probably the shell. This script, called fixit.sh, turned out to be longer than I thought it would be, mostly because of all the error checking:

```
#!/bin/bash
# Script to fix up OpenDocument Text (.odt) files
# "cd" to the directory containing "fixit.py".

# Make $TMPDIR, a new temporary directory

TMPDIR=/tmp/ODFfixit.$(date +%y%m%d.%H%M%S).$$
if rm -rf $TMPDIR && mkdir $TMPDIR; then
   : # Be happy
else
   echo >&2 "Can't (re)create $TMPDIR; aborting"
   exit 1
fi

OLDFILE=$1
NEWFILE=$2

# Check number of parameters.
# Ensure $NEWFILE's dir exists and is writable.
# Quietly Unzip $OLDFILE. Whine and abort on error.

if [[ $# -eq 2 ]] &&
   touch $NEWFILE && rm -f $NEWFILE &&
                    unzip -q $OLDFILE -d $TMPDIR ; then
   : # All good; be happy.
else

   # Trouble! Print usage message, clean up, abort.

   echo >&2 "Usage: $0 OLDFILE NEWFILE"
   echo >&2 "  ... both OpenDocument Text (odt) files"
   echo >&2 "Note: 'OLDFILE' must already exist."
   rm -rf $TMPDIR
   exit 1
fi

# Save file list in $F; is content.xml there?

F=$(unzip -l $OLDFILE |
       sed -n '/:[0-9][0-9]/s|^.*:.. *||p')
if echo "$F" | grep -q '^content\.xml$'; then
   : # Good news; we have content.xml
else
   echo >&2 "content.xml not in $OLDFILE; aborting"
   echo >&2 TMPDIR is $TMPDIR
   exit 1
fi

# Now invoke the Python program to fix content.xml

mv $TMPDIR/content.xml $TMPDIR/OLDcontent.xml
if ./fixit.py $TMPDIR/OLDcontent.xml > \
                    $TMPDIR/content.xml; then
   : # It worked.
else
   echo >&2 "fixit.py failed in $TMPDIR; aborting"
   exit 1
fi

if (cd $TMPDIR; zip -q - $F) | cat > $NEWFILE; then
   # Everything worked! Clean up $TMPDIR
   rm -rf $TMPDIR
else # something Bad happened.
   echo >&2 "zip failed in $TMPDIR on $F"
   exit 1
fi
```

It's long but straightforward, so I explain only a few things here.

First, the temporary directory name includes the date and time (the `date +%` stuff), and the shell's process ID (the $$) prevents name collisions.

Second, the grep line looks the way it does because I want it to accept content.xml but not something like discontent.xml or content-xml.

Finally, we clean up the temporary directory ($TMPDIR) except in some error cases, where we leave it intact for debugging and tell the user where it is.

We can't run this script yet, because we don't yet have fixit.py actually modify content.xml. But, we can use a stub to validate what we have so far. The fixit.sh script assumes fixit.py will take one parameter (the original content.xml's pathname) and put the result onto stdout. This just happens to match the calling sequence for /bin/cat with one parameter; hence, if we use /bin/cat as our fixit.py, fixit.sh should give us a new document with the same content as the old. So, let's give it a whirl:

```
% ln -s /bin/cat fixit.py
% ./fixit.sh ex1.odt foo.odt
% ls -l ex1.odt foo.odt
-rw-r--r--  1 collin users 7839 2006-11-14 17:50 ex1.odt
-rw-r--r--  1 collin users 7900 2006-11-14 19:45 foo.odt
% oowriter foo.odt
```

The new file, foo.odt, is slightly larger than ex1.odt, but when I looked at it with OpenOffice.org Writer, it had the right stuff.

As far as writing a program for manipulating content.xml—well, back in the 1990s, I probably would have spent many hours with yacc (or bison)—but today, Python with its XML libraries is a more natural choice.

The Bottom Layer: a Python/XML Script

My desktop distribution (SUSE 9.3) includes the packages python-doc-2.4-14 and python-doc-pdf-2.4-14. You also can get documentation from **www.python.org**. In either case, we want the Library Reference, which contains information on the Python XML libraries; they are described in the chapter on "Structured Markup Processing Tools" (currently Chapter 13).

Several modules are listed, and I noticed one labeled lightweight: xml.dom.minidom—Lightweight Document Object Model (DOM) implementation.

Lightweight sounded good to me. The library reference gives these examples:

```
from xml.dom.minidom import parse, parseString

dom1 = parse('c:\\temp\\mydata.xml') # parse an XML file by name

datasource = open('c:\\temp\\mydata.xml')
dom2 = parse(datasource)   # parse an open file
```

So, it looks like parse can take a filename or a file object.

Exploring content.xml

Once we create a dom object, what can we do with it? One nice thing about Python is the interactive shell, which lets you try things out one at a time. Let's unpack the first example and look inside:

```
% mkdir TMP
% unzip -q -d TMP ex1.odt
% python
Python 2.4 (#1, Mar 22 2005, 21:42:42)
[GCC 3.3.5 20050117 (prerelease) (SUSE Linux)] on linux2
Type "help", "copyright", "credits" or "license"
 for more information.
>>> import xml.dom.minidom
>>> dom=xml.dom.minidom.parse("TMP/content.xml")
>>> dir(dom)
[  --- a VERY long list, including ---
'childNodes', 'firstChild', 'nodeName', 'nodeValue', ...  ]
>>> len(dom.childNodes)
1
>>> c1=dom.firstChild
>>> len(c1.childNodes)
4
>>> for c2 in c1.childNodes: print c2.nodeName
...
office:scripts
office:font-face-decls
office:automatic-styles
office:body
>>>
```

Notice how Python's dir function tells what fields (including methods) are in the object. The childNodes field looks interesting, and indeed, it appears that the document has a tree structure. After a little more manual exploration, I discovered that text is contained in the data field of certain nodes. So, I coded up a naive script, fix1-NAIVE.py:

```
#!/usr/bin/python -tt
import xml.dom.minidom
import sys

DEBUG = 1
def dprint(what):
   if DEBUG == 0: return
   sys.stderr.write(what + '\n')

def handle_xml_tree(aNode, depth):
   if aNode.hasChildNodes():
      for kid in aNode.childNodes:
         handle_xml_tree(kid, depth+1)
   else:
      if 'data' in dir(aNode):
         dprint(("depth=%d: " + aNode.data) % depth)

def doit(argv):
   doc = xml.dom.minidom.parse(argv[1])
   handle_xml_tree(doc, 0)
   # sys.stdout.write(doc.toxml('utf-8'))

if __name__ == "__main__":
   doit(sys.argv)
```

The dprint routine prints debugging information on stderr; later we'll set DEBUG=0, and it'll be silent. Okay, let's try that on the content.xml above:

```
% ./fix1-NAIVE.py TMP/content.xml
depth=5: Turn all these
depth=6: "straight"
Traceback (most recent call last):
  File "./fix1-NAIVE.py", line 24, in ?
    doit(sys.argv)
  File "./fix1-NAIVE.py", line 20, in doit
    handle_xml_tree(doc, 0)
  File "./fix1-NAIVE.py", line 13, in handle_xml_tree
    handle_xml_tree(kid, depth+1)
  File "./fix1-NAIVE.py", line 13, in handle_xml_tree
    handle_xml_tree(kid, depth+1)
  File "./fix1-NAIVE.py", line 13, in handle_xml_tree
    handle_xml_tree(kid, depth+1)
  File "./fix1-NAIVE.py", line 13, in handle_xml_tree
    handle_xml_tree(kid, depth+1)
  File "./fix1-NAIVE.py", line 13, in handle_xml_tree
    handle_xml_tree(kid, depth+1)
  File "./fix1-NAIVE.py", line 16, in handle_xml_tree
    dprint(("depth=%d: " + aNode.data) % depth)
  File "./fix1-NAIVE.py", line 8, in dprint
    sys.stderr.write(what + '\n')
UnicodeEncodeError: 'ascii' codec can't encode character
u'\u201c' in position 22: ordinal not in range(128)
%
```

What's that error about? When trying to print that string on stderr, we hit a non-ASCII character—probably one of those curly quotes. A quick Web search gave this possible solution:

```
sys.stderr.write(what.encode('ascii', 'replace') + '\n')
```

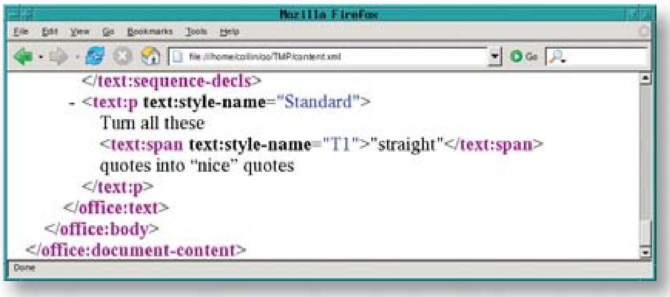
It says that if a non-ASCII Unicode character appears, replace it with something in ASCII—an equivalent, or at least something printable.

Replacing line 8 with that yields this output:

```
% ./fix1.py TMP/content.xml
depth=5: Turn all these
depth=6: "straight"
depth=5:  quotes into ?nice? quotes
%
```

So the curly quotes were replaced with ? characters, which is fine for our debugging output. Note that the text doesn't necessarily all come at the same depth in the tree.

The document's structure also can be seen by typing the full filename of the content.xml file into a Firefox window (Figure 7). That's good for displaying the data; the point, however, is to change it!


Mozilla Firefox
File Edit View Go Bookmarks Tools Help
file:///home/collin/oo/TMP/content.xml
Go
Done

```
    </text:sequence-decls>
  - <text:p text:style-name="Standard">
      Turn all these
      <text:span text:style-name="T1">"straight"</text:span>
      quotes into "nice" quotes
    </text:p>
  </office:text>
 </office:body>
</office:document-content>
```

Figure 7. Firefox presents the XML more clearly.

Simple String Replacement

Let's take fix1.py and make an easy modification. Whenever two hyphens appear, replace them with the em dash. Then, when we're done, write the XML to stdout—that's exactly what the shell script (fixit.sh) expects.

We'll specify the em dash by giving its hex value; to find it, locate the em dash in OpenOffice.org Writer's Insert→Special Character dialog (Figure 8).

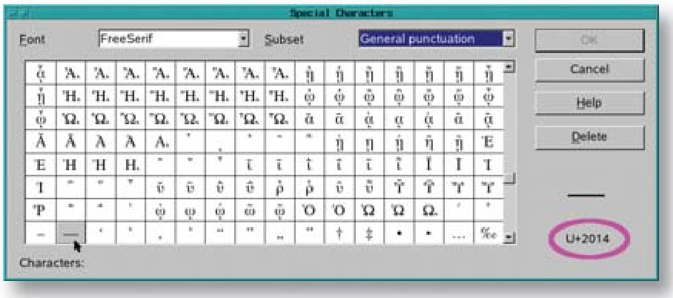


Figure 8. Selecting and Inserting Special Characters

When I select the long dash (the em dash), its Unicode value appears in the lower-right corner, where I've put a purple ellipse; that's the value to put into the string in place of the double hyphens. Let's call this script fix2.py:

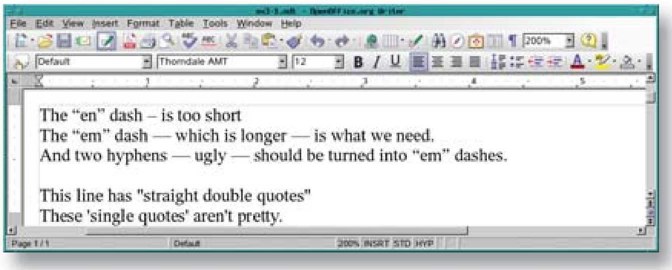
```
#!/usr/bin/python -tt
import xml.dom.minidom
import sys

DEBUG = 1
def dprint(what):
    if DEBUG == 0: return
    sys.stderr.write(what.encode('ascii',
                                 'replace') + '\n')

emDash=u'\u2014'

def fixdata(td, depth):
    dprint("depth=%d: childNode: %s" %
           (depth, td.data))
    # OK, so '--' becomes em dash everywhere
    td.data = td.data.replace('--', emDash)

def handle_xml_tree(aNode, depth):
    if aNode.hasChildNodes():
        for kid in aNode.childNodes:
            handle_xml_tree(kid, depth+1)
    else:
        if 'data' in dir(aNode):
            fixdata(aNode, depth)

def doit(argv):
    doc = xml.dom.minidom.parse(argv[1])
    handle_xml_tree(doc, 0)
    sys.stdout.write(doc.toxml('utf-8'))

if __name__ == "__main__":
    doit(sys.argv)
```

Notice how easy Python makes it to replace a pattern in a string. Strings in recent Python versions have a built-in method, replace, that causes one substring to be replaced by another:

```
td.data = td.data.replace('--', emDash)
```

Let's plug fix2.py into fixit.sh to see how well it works:

```
% ln -sf fix2.py fixit.py
% ./fixit.sh ex3.odt ex3-1.odt
depth=5: childNode: The ?en? dash ? is too short
depth=5: childNode: The ?em? dash ? which is longer ?
 is what we need.
depth=5: childNode: And two hyphens -- ugly -- should
 be turned into ?em? dashes.
depth=5: childNode: This line has "straight double quotes"
depth=5: childNode: These 'single quotes' aren't pretty.
% oowriter ex3-1.odt
%
```


ex3-1.odt - OpenOffice.org Writer
File Edit View Insert Format Table Tools Window Help
Default
Thorndale AMT
12
200%
Page 1 / 1
Default
200% INSRT STD HYP

The “en” dash – is too short
The “em” dash — which is longer — is what we need.
And two hyphens — ugly — should be turned into “em” dashes.

This line has "straight double quotes"
These 'single quotes' aren't pretty.

Figure 9. This looks like a job for Python.

Success! Now for the rest. Besides the double hyphen, we want to change the en dash into an em dash. That syntax is just like the double hyphen replacement.

Replacement Using Regular Expressions

Replacing straight quotes with curly ones is more complicated though, because we have to decide between a starting double quote and an ending double-quote character. How to tell? Well, if the quote character is at the start of the string, and there’s a nonspace character afterward, it’s a left (or start of quote) curly quote. Ditto if there’s a blank before it and a nonspace afterward.

That’s the easy way to describe it. We could code it like that, or we could simply write a regular expression. I looked at the section titled “re -- Regular expression operations” in Chapter 4 of Python’s library documentation and eventually came up with this:

```
sDpat = re.compile(r'(\A|(?<=\s))"(?=\S)', re.U)
```

Let me explain this left to right. We are creating sDpat, the pattern for a starting double quote or Starting Double-quote PATtern. We do that by calling the method compile in the re module (for regular expressions). That analyzes the pattern once and creates a regular expression object. We’ll use sDpat to match straight double quotes that should be turned into nice curly quotes at the start of a quotation.

Now, about the pattern—the pattern contains a double-quote character (") so we delimit it with single quotes, ‘like this’. Also, we’ll pass some escapes (such as \A and \s) to re.compile, so let’s make this a raw string by putting an r in front of it.

(A little explanation for Perl users: in Python, \ escapes are interpolated except in raw strings, whether single-quoted or double-quoted; the delimiters don’t affect interpolation as they do in Perl.)

We can see how raw strings work by using Python’s shell:

```
>>> print 'normal string: \n is a newline'
normal string:
 is a newline
>>> print r'raw string: \n is not a newline'
raw string: \n is not a newline
>>>
```

So, what’s in that raw string? It consists of three parts:

1. The part before the quote character (`\A|(?<=\s)`). What we are doing is matching something (the ‘"’ in this case), but only if it occurs at the beginning of the string or if it’s preceded by a whitespace character. The \A means “match beginning of the string”, the | means “or” and (?<=\s) means “match if immediately preceded by whitespace (a blank, tab or newline), but don’t include that whitespace itself in the match”. The enclosing parentheses denote grouping.

2. The straight double quote itself: ". That’s what we’re matching.

3. The part after the ‘"’: (?=\S). What we’re doing is adding another condition—that the quote character be followed by a non-whitespace character.

If all three conditions are met—that is, if a quote is there (condition 2), and it’s either at the start of the string or preceded by whitespace (condition 1), and it’s followed by some non-whitespace character (condition 3), we want to replace it by an opening double-quote character.

Besides the pattern, you also can pass flags to re.compile. We pass re.U to make certain escapes dependent on the Unicode character database. Because we’re parsing a Unicode string, I think we want that.

Let’s call this fix3.py:

```
#!/usr/bin/python -tt
import xml.dom.minidom
import sys
import re                            # new in fix3.py

DEBUG = 1
def dprint(what):
   if DEBUG == 0: return
   sys.stderr.write(what.encode('ascii',
                              'replace') + '\n')

emDash=u'\u2014'
enDash=u'\u2013'                  # new in fix3.py
sDquote=u'\u201c'                 # new in fix3.py

# sDpat: pattern for starting dbl quote, as
#         "Go! <-- the quote there
#         We look for it either at start (\A) or
#         after a space (\s), and we want it to be
#         followed by a non-space
sDpat = re.compile(r'(\A|(?<=\s))"(?=\S)', re.U) # new in fix3.py

def fixdata(td, depth):
   dprint("depth=%d: childNode: %s" %
           (depth, td.data))
   # OK, so '--' becomes em dash everywhere
   td.data = td.data.replace('--', emDash)
   # Change 'en' dash to 'em' dash
   td.data = td.data.replace(enDash , emDash)  # new in fix3.py
   # Make a nice starting curly-quote          # new in fix3.py
   td.data = sDpat.sub(sDquote, td.data)       # new in fix3.py

def handle_xml_tree(aNode, depth):
   if aNode.hasChildNodes():
      for kid in aNode.childNodes:
         handle_xml_tree(kid, depth+1)
   else:
      if 'data' in dir(aNode):
         fixdata(aNode, depth)

def doit(argv):
   doc = xml.dom.minidom.parse(argv[1])
   handle_xml_tree(doc, 0)
   sys.stdout.write(doc.toxml('utf-8'))

if __name__ == "__main__":
   doit(sys.argv)
```

Note that the syntax for replacing a regular expression differs from that of substring replacement: we use the sub (substitute) method of

LINUX JOURNAL
1994-2006
ARCHIVE
ISSUES 1–152 of Linux Journal

the regular expression object (sDpat in this case):

```
td.data = sDpat.sub(sDquote, td.data)
```

Here we're taking td.data, the data in this particular node in the XML tree, looking for the regular expression specified by sDpat, and replacing whatever matched it (the straight " character in the appropriate context) with the starting double quote, sDquote.

Now, if we try fixit.sh with fix3.py as the lower-level program:

```
% ln -sf fix3.py fixit.py
% ./fixit.sh ex3.odt ex3-2.odt
depth=5: childNode: The ?en? dash ? is too short
depth=5: childNode: The ?em? dash ? which is longer ?
 is what we need.
depth=5: childNode: And two hyphens -- ugly -- should be
 turned into ?em? dashes.
depth=5: childNode: This line has "straight double quotes"
depth=5: childNode: These 'single quotes' aren't pretty.
% oowriter ex3-2.odt
%
```

OpenOffice.org Writer showed what we expected. Both the double hyphen and the en dash changed into em dashes, and the starting double quote curves the right way.

Now, here's the rest. The expression to deal with the ending double quote is the mirror image of the starting double quote. In order to write an ending/closing double quote, we require the quote character either to be at the end of the string (\Z) or followed by whitespace. Again, when we do the replacement, we want to replace only the quote itself, not the whitespace. Hence, the Ending Double-quote PATtern (eDpat) is given by:

```
eDpat = re.compile(r'("\Z)|("(?=\s))', re.U)
```

By the way, we compile all these patterns because we're going to use them over and over again when processing documents.

To handle single quotes ('like these'), we basically can do the same thing, except for a couple of issues. First, is the problem of contractions. When handling a double-quote character, we didn't cover the case where it was surrounded on both sides by non-whitespace. With single quotes (or apostrophes), we can't avoid that, because of words such as can't. Therefore, although the starting single-quote pattern can match the starting double-quote pattern, the other one, which doubles as an apostrophe in contractions, has a looser pattern. Here's what I came up with:

```
eSpat = re.compile(r"(?<=\S)'", re.U)
```

Because the pattern has an apostrophe in it, we delimit the pattern string using double-quote characters. This expression matches a single quote, but only when preceded immediately by a non-whitespace character.

The second issue, which the code doesn't address, is that of contractions beginning with an apostrophe, such as 'tis the season or stick 'em up.

The script treats the leading apostrophe like the start of a single-quoted phrase, and the single quote will face the wrong way. This probably will need a manual work-around.

Putting all this together, we have fix4.py:

```
#!/usr/bin/python -tt
import xml.dom.minidom
import sys
import re                   # new in fix3.py

DEBUG = 1
def dprint(what):
    if DEBUG == 0: return
    sys.stderr.write(what.encode('ascii',
                                   'replace') + '\n')

emDash=u'\u2014'
enDash=u'\u2013'           # new in fix3.py
sDquote=u'\u201c'          # new in fix3.py
eDquote=u'\u201d'          # new in fix4.py
sSquote=u'\u2018'          # new in fix4.py
eSquote=u'\u2019'          # new in fix4.py

# sDpat: pattern for starting dbl quote, as
#         "Go! <-- the quote there
#         We look for it either at start (\A) or
#         after a space (\s), and we want it to be
#         followed by a non-space
sDpat = re.compile(r'(\A|(?<=\s))"(?=\S)', re.U)  # new in fix3.py
eDpat = re.compile(r'("\Z)|("(?=\s))', re.U)      # new in fix4.py
sSpat = re.compile(r"(\A|(?<=\s))'(?=\S)", re.U)  # new in fix4.py
eSpat = re.compile(r"(?<=\S)'", re.U)             # new in fix4.py

def fixdata(td, depth):
    dprint("depth=%d: childNode: %s" %
           (depth, td.data))
    # OK, so '--' becomes em dash everywhere
    td.data = td.data.replace('--', emDash)
    # Change 'en' dash to 'em' dash
    td.data = td.data.replace(enDash , emDash)   # new in fix3.py
    # Make a nice starting curly-quote           # new in fix3.py
    td.data = sDpat.sub(sDquote, td.data)        # new in fix3.py
    td.data = eDpat.sub(eDquote, td.data)        # new in fix4.py
    # Make nice curly single-quote characters
    td.data = sSpat.sub(sSquote, td.data)        # new in fix4.py
    td.data = eSpat.sub(eSquote, td.data)        # new in fix4.py

def handle_xml_tree(aNode, depth):
    if aNode.hasChildNodes():
        for kid in aNode.childNodes:
            handle_xml_tree(kid, depth+1)
    else:
        if 'data' in dir(aNode):
            fixdata(aNode, depth)

def doit(argv):
    doc = xml.dom.minidom.parse(argv[1])
    handle_xml_tree(doc, 0)
    sys.stdout.write(doc.toxml('utf-8'))
```

```
if __name__ == "__main__":
   doit(sys.argv)
```

Let’s try that on our example:

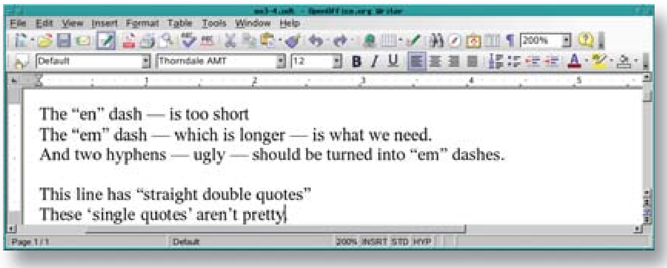
```
% ln -sf fix4.py fixit.py
% ./fixit.sh ex3.odt ex3-4.odt
depth=5: childNode: The ?en? dash ? is too short
depth=5: childNode: The ?em? dash ? which is
 longer ? is what we need.
depth=5: childNode: And two hyphens -- ugly -- should
 be turned into ?em? dashes.
depth=5: childNode: This line has "straight
 double quotes"
depth=5: childNode: These 'single quotes'
 aren't pretty.
% oowriter ex3-4.odt
```

Let’s review what we’ve done here:

- Wrote scripts to unpack and repack ODF files.
- Learned about using Python to understand the structure of ODF files.
- Wrote a Python program to perform useful transformations on an OpenOffice.org Writer file, using regular expressions and the built-in string methods.

What Next?

I hope this introduction has been useful, but it’s only the beginning of how Python/XML can work with ODF files.



The “en” dash — is too short
The “em” dash — which is longer — is what we need.
And two hyphens — ugly — should be turned into “em” dashes.

This line has “straight double quotes”
These ‘single quotes’ aren’t pretty

Figure 10. Python string handling gets results.

For example, I had an OpenDocument spreadsheet, and I wanted to add up the values of all cells having a yellow background, which Python/XML allowed me to do. I’ve also had the need to get all the e-mail addresses from one column of a spreadsheet, except for those in italic or strikeout type. I don’t think OpenOffice.org will let me do that, but Python/XML will.■

Collin Park has been a computer engineer since 1976 and currently works for Network Appliance. He runs Linux on four computers at home, which he shares with his wife and two teenage daughters.

Resources

Current Python Library Reference:
docs.python.org/lib

Older (pre-2.5) Versions of Python Documentation:
www.python.org/doc/versions

Dave Taylor’s Work the Shell columns in *Linux Journal* provide a terrific introduction to shell scripting.

“Why Not Python?” (the old C hacker drags himself into the late 1990s):
linuxjournal.com/article/8794, **linuxjournal.com/article/8729**, **linuxjournal.com/article/8858** and **linuxjournal.com/article/8859**

They're Ba-ack

The Network Computing revolution rears its beautiful head once again, thanks to Ajax.

Nick Petreley, Editor in Chief

What is Ajax, really? There's Ajax the technology, and then there's the fact that Ajax makes it possible to provide a rich client experience on a Web browser. Think about the latter for a moment. Wasn't that the basic idea behind Network Computing?

Does anyone recall when IBM and Oracle pushed the concept of Network Computers? Network Computing was all about delivering a rich client experience without the price tag of a PC and a commercial PC operating system. It also was about centralized storage and client management, which would bring the total cost of ownership way down.

The concept was so logical and compelling that it struck fear in the hearts of Microsoft and mainstream computer journalists. Microsoft had the most to lose. The Network Computing environment promoted by IBM and Oracle was, by necessity, platform-neutral. The goal was to base everything on browsers and Java, making the hardware platform and operating system irrelevant. You could participate in the revolution with a powerful Windows PC equipped with Java, but the fact that you ran Windows was incidental. It wasn't a necessary component.

The revolution self-destructed, however. Despite how sensible the concept may have been, there were two things wrong with it. First, hardware and Java weren't ready. Java applications were buggy, and most Network Computing appliances walked software, they didn't run it.

But here's what really killed the movement. IBM, Sun and Oracle discovered it was incredibly hard to make good money by selling truckloads of cheap computing appliances and only a handful of powerful servers. It's much more profitable to sell people massive computing power at the client as well as the server end, even if the average user never takes full advantage of the client machine's power.

The economic impact is also the reason why most computer journals hated Network Computers. The success of Network Computing would scale down the computing economy so much that advertising revenues would plummet. As a result, the mainstream computing press printed reams of anti-NC propaganda and hung on Microsoft's every word about the NetPC and Zero-Administration Windows. Remember those? Right, these reactive "initiatives" by Microsoft vanished the moment it became obvious that the Network Computing revolution wasn't going to get off the ground.

Wasn't going to get off the ground—yet. I used to go show-hopping with a presentation about Network Computing. I repeatedly predicted that Network Computing was so sensible you could count on the success of a Network Computing revolution, whether it happened that year or in decades. I also predicted that it would be based on Java, but I was careful to add that Java wasn't necessary. If Java flopped, some other platform-neutral technique of delivering applications and content would emerge in its place.

Hello Ajax. Ajax-based office suites are popping up everywhere, some free as in FOSS, some free as in service, some nonfree and some free with upgrade options. You can get a taste of the experience if you sign up at **www.ajax13.com** for free access to a suite of Ajax-based office applications. Or, you can try out Google's Docs and Spreadsheets at **docs.google.com**. Better still, you'll find out why I still prefer Java over Ajax by trying out the ThinkFree office suite beta at **www.thinkfree.com**. ThinkFree lets you choose between a lightweight and heavy-duty application. The lightweight applications are Ajax-based, and the heavy-duty applications are Java-based. Both types of applications are terrific, but the Java-based applications, such as its heavy-duty word processor, is much more slick and polished than the Ajax equivalent.

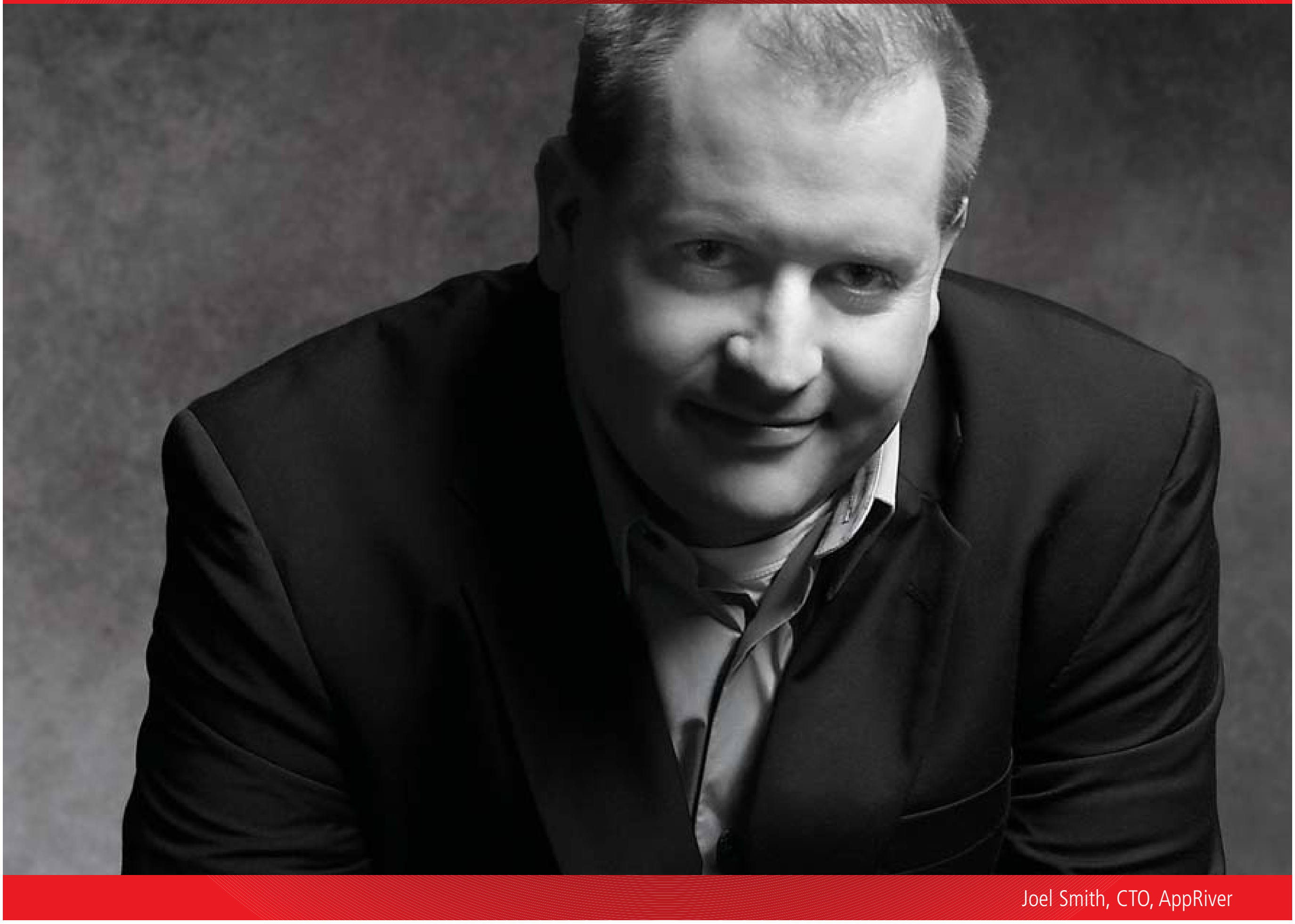
Here's why these efforts are much more likely to lead to a successful Network Computing revolution. They take advantage of the relative platform neutrality of browsers, but the success of these Web-based suites is not tied to any hardware platform. In other words, the NC revolution as pitched by IBM, Oracle and Sun expected you to buy a truck full of cheap clients. These Ajax and Java Web-based applications will work on a cheap client, but that's purely coincidental. This approach to the Network Computing revolution doesn't hinge upon changing what you buy, thus enabling hardware companies to keep selling you faster boxes with decent profit margins.

That's where Linux comes in. If this Network Computing revolution succeeds, OEMs will have one less reason to pay more to sell a Windows box than a Linux box. If people begin to depend on Web-based office applications, why pay Microsoft an OS tax on every unit when people can get the same experience with Linux and Firefox?

Many people will voice most of the same fears and objections as they did during the previous attempt to push Network Computing back in the late 1990s. If Web-based office suites pick up enough steam, you'll see these fears dissipate.

I'm still a bigger fan of Java than Ajax, and the fact that Java is going GPL may change the future of Web-based suites. But, even if we end up with Java, we probably will thank Ajax for getting it started. Regardless, I maintain that we will see a Network Computing revolution, whether it's today or decades from now. And, when it happens, sooner or later, it will be great for Linux.■

Nicholas Petreley is Editor in Chief of *Linux Journal* and a former programmer, teacher, analyst and consultant who has been working with and writing about Linux for more than ten years.

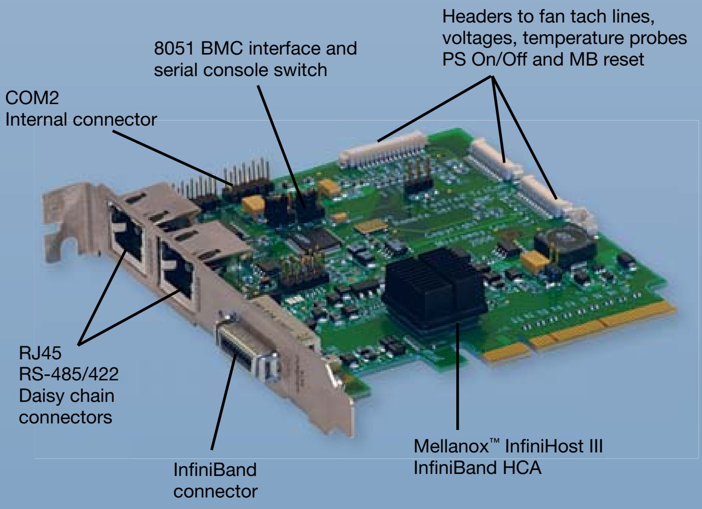


Affordable InfiniBand Solutions

4 Great Reasons to Call Microway NOW!

TriCom™

- DDR/SDR InfiniBand HCA
- "Switchless" serial console
- NodeWatch web enabled remote monitor and control


Headers to fan tach lines, voltages, temperature probes PS On/Off and MB reset
8051 BMC interface and serial console switch
COM2 Internal connector
RJ45 RS-485/422 Daisy chain connectors
InfiniBand connector
Mellanox™ InfiniHost III InfiniBand HCA

FasTree™

- DDR InfiniBand switches
- Low latency, modular design
- 24, 36 and 48 port building blocks

InfiniScope™

- Monitors ports on HCA's and switches
- Provides real time BW diagnostics
- Finds switch and cable faults
- Lane 15 interface
- Logs all IB errors

ServaStor™

- Extensible IB based storage building blocks
- Redundant and scalable
- Parallel file systems
- Open source software
- On-line capacity expansion
- RAID 0,1,1E, 3, 5, 6, 10, 50

Upgrade your current cluster, or let us design your next one using Microway InfiniBand Solutions.

To speak to an HPC expert
*call **508 746-7341** and ask*
for technical sales or email
sales@microway.com
www.microway.com

Microway®
Technology you can count on℠