LINUX
JOURNAL

Since 1 _____ of the Linux Community

LJ Interviews
Marten Mickos
of MySQL

REAL-TIME
MULTIPROCESSING AND
EMBEDDED SYSTEMS

NEW TECH TIPS COLUMN
WITH GNULL AND VOYD

A LINUX
LAPTOP
WHERE
EVERYTHING
WORKS?

QTOPIA VS. NANO-X
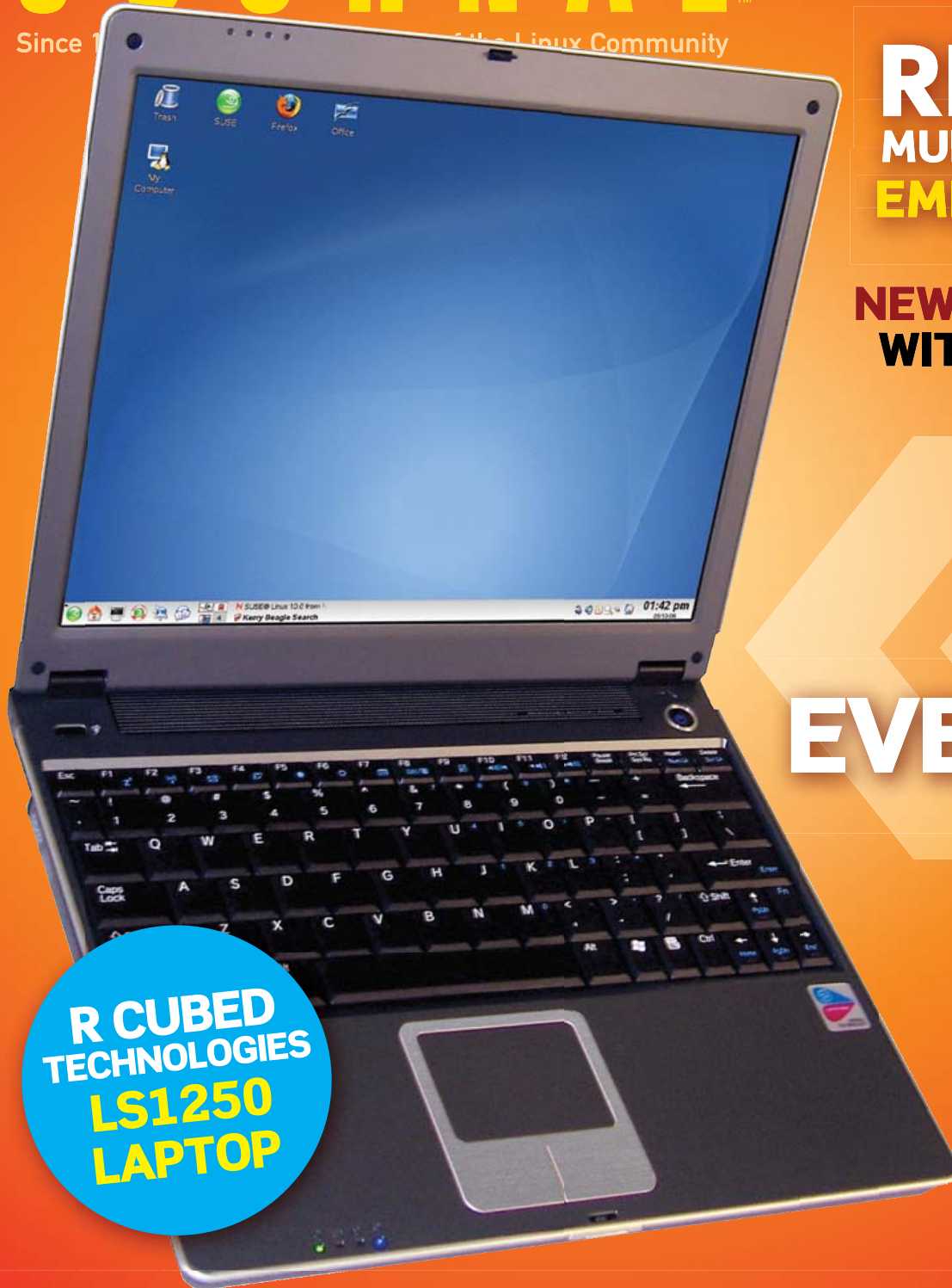
LINUX ON PICO E12

R CUBED
TECHNOLOGIES
LS1250
LAPTOP

JANUARY 2007 | ISSUE 153
www.linuxjournal.com

Today, Carlo restored a failed router in Miami, rebooted a Linux server in Tokyo, and remembered someone's very special day.

With Avocent centralized management solutions, the world can finally revolve around you. Avocent puts secure access and control right at your fingertips – from multi-platform servers to network routers, your local data center to branch offices. Our "agentless" out-of-band solution manages your physical and virtual connections (KVM, serial, integrated power, embedded service processors, IPMI and SoL) from a single console. You have guaranteed access to your critical hardware even when in-band methods fail. Let others roll crash carts to troubleshoot – with Avocent, trouble becomes a thing of the past, so you can focus on the present.

**Visit www.avocent.com/special to download *Data Center Control:*
Guidelines to Achieve Centralized Management white paper.**

Avocent.

The Power of Being There®

CONTENTS JANUARY 2007
Issue 153

FEATURES

COVER PHOTO COURTESY R CUBED TECHNOLOGIES

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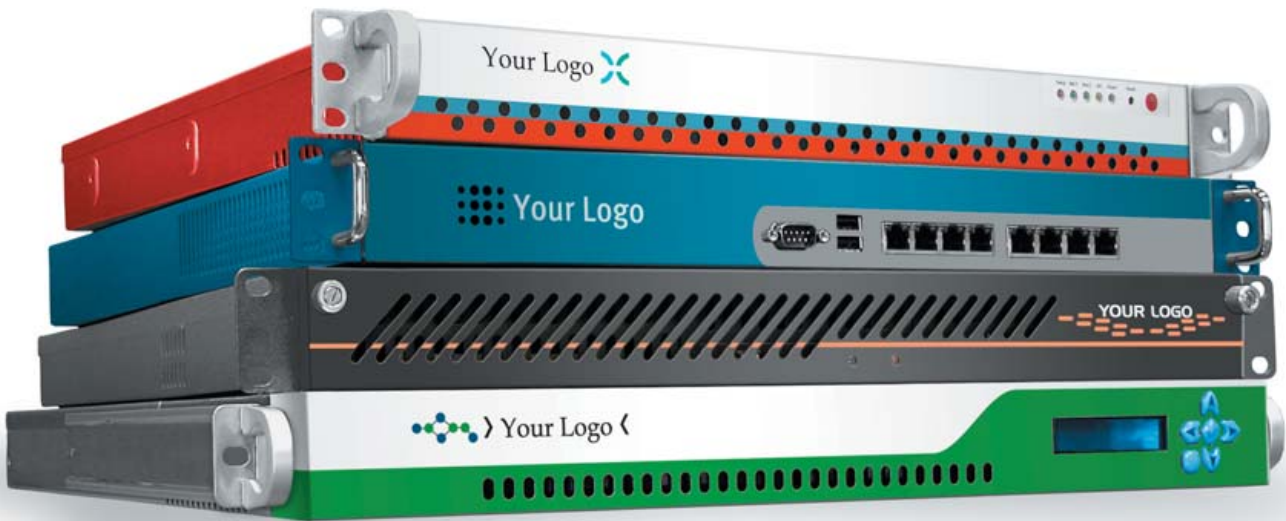
Custom server appliances or off the shelf reference platforms,
built with your image and software starting under $1,000.
From design to deployment, we handle it all.

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

CONTENTS JANUARY 2007 Issue 153

Next Month

MIGRATION

Next month we give you tips, instructions and advice on migrating from Windows to Linux and on how to make it easier for both operating systems to coexist. We'll show you how to manage and use NTFS partitions from within Linux, how to get to your Linux filesystems from within Windows and how to get off the hardware upgrade merry-go-round with Win4Lin Virtual Server.

Of course, we have much more. Our interview with Christof Wittig and Jerry Fiddler about db4o paints a picture of a unique blend of company and open source. We'll show you how STLdb4 makes C++ programming with the Berkeley DB a breeze and how to use libferris to create virtual documents.

Gemini 2U
— Ultra Dense Series

Up to 40% Less Power Consumption

Up to 40% Less Heat

- Two fully independent systems in a 2U
- Ability to run two discrete operating systems in one box
- Up to 8 CPU cores (forward compatible to 16)
- 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

Patent Pending

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 up to 40% less power consumption, 40% less heat and 30% 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.

AMD Solution Provider PLATINUM

AMD 64 Opteron

Open Source Systems™

LINUX JOURNAL

Since 1994: The Original Magazine of the Linux Community

Digital Edition
Now Available!

Read it first
Get the latest issue before it hits the newsstand

Keyword searchable
Find a topic or name in seconds

Paperless archives
Download to your computer for convenient offline reading

Same great magazine
Read each issue in high-quality PDF

Try a Sample Issue!
www.linuxjournal.com/digital

Linux on a Small Satellite

$119

Iraq — University of Baghdad

Kosovo — Pristina

Benin — Cotonou

Denmark — Djursland

Mongolia — Ulanbataar

Mozambique — Maputo

South Africa — Western Cape

Iraq — Sulaimaniyah

Italy — Apennine mountains

Czech Republic, The First MikroTik User Meeting, 154 participants from 37 countries — Prague

MikroTik around the world
no matter what the conditions, mikrotik is routing your world. www.mikrotik.com

I Love Your Opinions

Nick, I love [your columns] and all your articles. Up until you came along, I thought I was the only Linux jinx. I've had very few easy install/upgrades of Linux. My latest, Kubuntu5, installs fine, although Kubuntu6 can't seem to find my gateway to the Internet, no matter what I set with ifconfig, so I'm stuck for now with 5. My motherboard is an ASUS A8V-MX with an integrated Ethernet, sound and video (which don't work), so again, I agree with you.

I've used CP/M, DOS, OS/2 and (yuck) windoze. Then three years ago, I discovered Linux, and I've been converted—if only it wasn't so damn picky.

/var/opinion/ this month, October 2006, is very VERY interesting! I plan to use your experience installing MythTV as a guide to make my attempt to do the same upgrade. Please make this an article, including all hardware and software, with explicit steps to follow.

--
Eric Forbes

Déjà Vu All over Again

Your recent changes to the magazine layout reminded me of the time when *Dr. Dobb's Journal* dropped Forth from the programming tree.

The older *Dr. Dobb's Journal*'s taught me to use the then free language Forth; *Dr. Dobb's Journal* also felt like it was losing its innocence.

Ten pages of ads before making any sense is beautifully commercial.

I wish you all success in presenting Linux with a glossy, professional, corporate image.

--
Colin Tree

Fedora Disk Labels All over Again

I have found a reason why Fedora uses disk labels rather than device file entries. It may be because, on some machines, using device files causes a strange race condition. For example, in our lab (**www.minds.may.ie/~balor/photographs/lab**) we have machines on which GRUB correctly interprets /dev/sda1 as the first SATA device. However, on boot (before root is mounted), the Linux kernel (sometimes) assigns the /dev/sda1 device file to the first USB mass-storage device that happens to be plugged in to the machine, causing a failure to mount the root partition. Obviously, using / as the root disk label means that any USB mass-storage disks would be ignored, and the correct partition on the SATA disk would be mounted. I do agree that a label such as FC5Root is preferable to /, as one could then have FC4Root and FC5Root on the same box.

--
Aidan Delaney

I noticed this same unpredictable behavior on one of my systems too. I installed a RAID card to avoid down time due to any single disk failure (I use RAID 5), and I noticed that Linux does not respect the boot order I set in the BIOS. I haven't noticed any race conditions with any given kernel, but one kernel will make the RAID card /dev/sda, and another kernel will make it /dev/sdb. I discovered how to control which it sees first with Ubuntu by changing a script in the initrd image. Look for how I did it in our new tech tips column this month called Tech Tips with Gnull and Voyd.—Ed.

No Myth about MythTV

I have just read your /var/opinion in the October 2006 issue of *Linux Journal*. You make a couple of valid points about MythTV—for example, that it can be time-consuming to set up, but once it is running, it is like 99% of the Linux boxes I have ever built in that it just keeps working! Can I suggest for ease of setup and use that you give MythDora a try? The ISO images can be found at **www.g-ding.tv**, and the main guy there is very helpful (so much so that I actually felt like I wanted to feed data back into the system, not feeling obliged as I have with some other projects I have assisted with).

Incidentally, you are right—it is becoming apparent, from what I have read on the forums, that the Hauppauge WINTV-PVR-500 is just 2*150's bolted together (so why didn't they call it the 300? Too obvious?). I am sure you are aware, but if you have any issues getting it working, the guys over at linuxtv are very approachable and have helped me out many a time with my Compro DVB-T300s and my Dvico FusionHDTV Dual (which I must admit thought would be a bag of pants but works flawlessly).

I must admit, I do look forward to reading your magazine, and your /var/opinion page does usually give me something to think about.

Now for a shameless plug: to see how far I have gotten with my MythTV build, go to **trueentropy.linuxbloggers.com**, and all should become apparent (if I have had time/remembered to update it).

--
Phil

Digital Subscriptions

Having bought the paper version of *LJ* for more than ten years, I took out a digital subscription last week. It's a real improvement! Not only can I read back issues without having to search round the flat for them, but I also can fit dozens of issues on a memory stick, and my wife doesn't complain that I'm hoarding paper!

Rotating the view on my IBM 1600x1200 ThinkPad and holding it sideways gives a perfectly readable full-page view.

--
Peter Grant-Ross

64-Bit Laptops

The discussion in the Letters column [October 2006] about 64-bit laptops caught my eye. One reason to have 64-bit laptops is binary compatibility with the desktop. My laptop is purposely set up to mirror my desktop system. I use rsync to move things back and forth and keep them synchronized.

Right now, I don't have to worry about moving executable files back and forth or where I do a "make". That's a situation I would like to keep unchanged if and when I move to 64-bit systems.

--
Aharon (Arnold) Robbins

Less Technical?

First off, I think *Linux Journal* is a terrific asset to the Linux community.

I have to agree with Guilherme DeSouza [see Letters, November 2006], who argues that *Linux Journal* is going downhill, as it moves toward less technical content and more attempts to appeal to the masses. This is what caused *Dr. Dobb's Journal* to crash while *LJ* was taking off: *Dr. Dobb's Journal* tried to expand into cute and slick articles and lost its core audience, while *LJ* was offering solid technical content.

I certainly applaud the intent of Taylor and Gagné to make Linux more accessible to a wider audience. But, I question whether *LJ* is the forum for it. For example, if you walk to your local bookseller and watch who buys *LJ*, how many MS Windows *programmers* do you think you'll see? I've tried to interest MS programmers in Linux and gotten nothing but bored stares.

If that's the case with Windows programmers, ask yourself how many Windows *users* are going to pick up a *Linux Journal*. My guess is the number of readers of *LJ* who have no programming background and no Linux background is less than 1%, but surely you have the numbers to say. If you really want to reach the masses, maybe *LJ* should sponsor a contest for the best Linux-related article published in, say, *PC Magazine* or the *Wall Street Journal*.

In short, I hope you'll not abandon your core audience as *Dr. Dobb's Journal* did. I enjoy *LJ* and read it thoroughly, and would hate to see *LJ* lose its way.

--
Steve

A Supplement to "The Dark Age of *Linux Journal*"

I have to admit that this note was provoked by Guilherme DeSouza's letter in the November 2006 issue. Let me introduce myself: I have been a user of Linux for a long time. In fact, I tried to install the Red Hat Halloween release in 1994. I had read of Linux before 1994, and I considered it the most exciting development in computing at that time. The Red Hat disc did not have all the drivers for the peripherals of my rather advanced Compaq, and despite many helpful but frustrating phone discussions with Red Hat, I actually used Slackware, which was more up to date and I think was packaged with Matt Welsh's "Linux Installation and Getting Started". This being said, I consider myself an amateur user of Linux rather than a professional. To such as myself, there was much of interest in the early issues of *Linux Journal*, and I have had a subscription since the beginning. One thing that sticks in my memory was the introduction of ext3 and the instructions in *LJ* to use a make file to install it, which really worked to my great excite-

ment. I used Linux as my basic working system in support of my research and for connecting to my central UNIX system until my retirement. Latterly, I found SUSE most satisfactory, and I continued using that at home. I will admit that I did dual-boot with Windows, because I needed some of the programs that made up the MS Office collection.

However, I am afraid that *Linux Journal* is becoming, dare I say it, boring to someone like myself. It seems to me to be aspiring to emulate a professional journal but without the rigor of the *Journal of the ACM*. My math is up to it, and I can scan the titles of the *ACM* and read the articles that interest me, but I used to read the *Linux Journal* cover to cover, and these past few months, I realized that I no longer do so. The other US Linux magazines (are there more than one?) seem to be thin negligible broadsheets, but the European journals are different. *Linux Magazine* (not the US magazine of the same name) *is* interesting, and I do read it more or less completely.

I read the on-line *TUX* magazine with a great amount of interest, but I would prefer to see some of its content in *Linux Journal*. I don't know if your editorial staff has looked at *Linux Magazine* recently, but they should do so despite the $10 price.

--
James Silverton

Thanks to all who have written to us about the technical content of Linux Journal. *One thing that needs to be said up front is that it is unrealistic to think* Linux Journal *or any magazine will please everyone. Those who find a particular article boring should keep in mind that for every article that you may feel is beneath you, another person finds that particular article the most useful and views your favorite technical article as incomprehensible. The Linux universe does not revolve around only one type of reader.*

Some of you may have noticed that we occasionally include some less-technical end-user content, because a portion of our readership appreciates it and benefits from it. We try to pick more advanced end-user topics, however, rather than basic point-and-click tutorials. Ironically, with respect to James Silverton's suggestion that we include content from TUX *in* Linux Journal, *that is the only category we consciously try to avoid— "new desktop user" content. Our sister magazine,* TUX *(**www.tuxmagazine.com**), targets that audience very well.*

Aside from including a variety of content, the only conscious shift we are making is to focus some of our more technical content to provide readers with information they can apply in practical ways, not just for our readers' amusement, but so they are better equipped to do their jobs. Having said that, we certainly do not believe that articles for your amusement and instruction are a bad thing. We understand that purely academic articles are desirable, interesting and have long-term benefits. So, we'll always include them. We simply need to strike the right balance. You can help by keeping your letters coming.—Ed.

One
PGI Unified Binary™

Now, PGI® compilers can generate a single PGI Unified Binary executable fully optimized for both Intel EM64T and AMD64 processors, delivering all the benefits of a single x64 platform while enabling you to leverage the latest innovations from both Intel and AMD. PGI Fortran, C, and C++ compilers deliver world-class performance and a uniform development environment across Linux and Windows as part of an integrated suite of multi-core capable software development tools. Visit **www.pgroup.com** to see why the leading independent software vendors in structural analysis, computational chemistry, computational fluid dynamics and automotive crash testing choose PGI compilers and tools to build and optimize their 64-bit applications.

The Portland Group™
www.pgroup.com   ++ 01 (503) 682-2806

diff -u

WHAT'S NEW IN KERNEL DEVELOPMENT

Michael Halcrow has submitted some patches to add support for public key cryptography in **eCryptFS**. Overall, folks like **Andrew Morton** seem to be in favor of this; although Andrew points out that there already is key management support in the kernel, and that perhaps the existing code should be extended to support eCryptFS's public key features, instead of creating that support afresh. But, Michael feels he's going in the right direction, and in spite of how any particular implementation details will resolve themselves, it does seem as though public key support in eCryptFS has arrived.

Alon Bar-Lev has extended the **kernel boot-command-line length** from 255 characters to 2,048 characters to accommodate all the stuff that's been piling into the command line in recent years, such as module arguments, initramfs, suspend and resume, and more. Unfortunately, it's become clear that one cannot simply extend the kernel command line. The command-line code is written in assembler and has such poor design and odd code behaviors within it that simple changes turn out to require bigger fixes. But **Andi Kleen**, **H. Peter Anvin**, Alon and others have taken this as an opportunity to clean up the whole mess. So, that's exactly what they're doing. It may delay the migration from the 255 to 2,048 character boot command lines, but it probably will open up other doors that have not yet been considered.

Writing **user-space PCI drivers** has been an insane process, according to **Greg Kroah-Hartman** and **Thomas Gleixner**. So they decided to do something about it. Thomas wrote up some infrastructure code to rein in the whole process, and Greg added his own touches. Now they've released the code, and a bunch of folks, including Andrew Morton, have begun piling on to get it into shape for actual inclusion into the kernel. The code already seems poised to become a generic user-space driver subsystem, not only for PCI drivers. So naturally, a bunch of people are considering possible names for the subsystem—everything from User Space Driver (USD) subsystem to Framework for Userspace Drivers (FUD) subsystem. Personally, I'd like to see a subsystem called FUD. Meanwhile, folks like **Manu Abraham** already are chomping at the bit to see this thing implemented fully, as it would have made some work he did with **Andrew de Quincey** go much more smoothly.

Neil Brown has been frustrated by the sheer number of ways it is possible to feed **configuration parameters** into the kernel in recent years. Between sysctl, SysFS, module parameters, kernel parameters and (in a hushed whisper) ProcFS, he doesn't know which thing to use anymore to configure some random module he's writing. He has asked for help and guidance. The discussion that followed may not have led to a definitive answer for Neil, beyond **Horst von Brand**'s recommendation of sysctl, but it did manage to get **Oleg Verych** to talk about his new configuration interface, called **etab** (short for External Text and Binary). The etab interface stores configuration in key/value pairs, and according to Oleg, may be useful in many parts of the kernel where configuration is done.

Joerg Roedel has implemented the protocol defined in **RFC 3378** to allow Ethernet packets to be tunneled through IPV4. As **Philip Craig** pointed out, **iproute2** already exists and would be a logical place to add Joerg's features. Joerg has agreed with this, but says he did the implementation separately to gain experience. Once the code begins to stabilize, his plan is to add it to iproute2.

Intel's Arjan van de Ven has announced the first release of the Linux-ready **Firmware Developer Kit**. This open-source Intel initiative involves a set of tests to see how well a system's BIOS will interact with Linux. Hopefully, says Arjan, this will help BIOS developers ensure that their systems continue to interoperate with Linux. Intel also is hopeful that developers will hop on board and start feeding bug fixes and support for additional BIOSes to the upstream sources.

—ZACK BROWN

Google Offers Code Search— Are Koders and Krugle Feeling Lucky?

In journalism, we say, "three examples makes a trend". In business school, professors teach that three competitors make a market category. Both tropes now apply to code search, since Google has jumped into a market pioneered by Koders.com three years ago and expanded by Krugle.com in the middle of 2005.

I asked Chris DiBona, Google's top open-source guru, about differentiation. He replied, "We are more comprehensive by an order of magnitude, and I think we give a faster, smarter experience. Our dupe-detection is really cool. You can almost instantly see which routine is more popular/used in the world (search for btree or some other common algorithm)."

Koders and Krugle are hardly standing still, of course. And, they can now press their advantages around the edges of a large market presence. For Koders, those include algorithms optimized for code searching and results ranking, search filters, an API so other services can access the search index, and an Enterprise Edition that searches behind company firewalls. For Krugle, those include iterative searching, search of related non-code documentation, ties of metadata to code, and a notes function for comments on (and linkage to) code.

Those, of course, are subsets of current offerings by all three services, which are sure to evolve and change even more as competition heats up and programmers become more involved.

—DOC SEARLS

A Little Linux

It's been a year since Mirus announced Koobox, a new line of desktop PCs that come loaded with Linux. The first offerings were standard tower configurations, starting at $299 US, pre-loaded with Linspire's latest distro. Then, in summer 2006, the company added a Mac-Mini-like unit with a mouse/keyboard/speaker bundle for $399.99 US (after a mail-in rebate). Since then, Mirus has been adding other Mini models, scaled upwards with faster CPUs, bigger drives and features like DVD+RW.

Mirus is a subsidiary of Equus, a Microsoft Platinum OEM and Gold Certified Partner, yet calls itself "The Largest Whitebox System Builder to the Channel" and was named by CRN as number 1 out of the 50 system builders. It'll be interesting to see how it does.

—DOC SEARLS

Mini Koobox

Linux laptops. Supported.

Technical support by phone and email, manufacturer's warranty, user's manual

Power management, suspend, hibernate, processor control

One touch suspend, hibernate, volume, brightness, external VGA, wireless

True multiprocessing with Intel Core 2 Duo, up to 4 GB RAM

Laptops from top-tier manufacturers

X Windows at full LCD resolution, OpenGL, NVidia and ATI 3D acceleration

Pre-configured Linux installation

Internal gigabit ethernet, wireless a/b/g, Bluetooth, EVDO mobile broadband

Ports: USB, PCMCIA, VGA, FireWire & more

Internal optical drive: CDRW, DVD±RW

Media cards: Compact Flash, Secure Digital

Biometric fingerprint (GDM login with PAM)

LJ Index, January 2007

1. Number of top ten most reliable hosting providers in September 2006 that run Linux: **8**

2. Number of top ten most reliable hosting providers in September 2006 that run "unknown": **2**

3. Number of top ten most reliable hosting providers in September 2006 that run Windows: **0**

4. Number of top 50 most reliable hosting providers that run Linux: **23**

5. Number of top 50 most reliable hosting providers that run FreeBSD: **6**

6. Number of top 50 most reliable hosting providers that run Windows: **12**

7. Number of top 50 most reliable hosting providers that run "unknown": **5**

8. Number of top 50 most reliable hosting providers that run Solaris: **4**

9. Number of sites surveyed by Netcraft: **97,932,447**

10. Results in a search for "linux" at Google codesearch: **4,280,000**

11. Number of lines of code indexed by Koders.com: **424,227,372**

12. Results in a search for "linux" at Koders.com: **179,222**

13. Results in a search for "linux" at Krugle.com: **700,529**

14. Billions of dollars in sales for the Linux server submarket in Q2 2006: **1.5**

15. Percentage increase in Linux server submarket sales for Q2 2006: **6.1**

16. Linux server shipment percentage growth for Q2 2006: **9.7**

17. Blade server sales percentage increase for Q2 2006: **37.1**

18. Blade server shipment percentage increase for Q2 2006: **29.7**

19. IBM's percentage share of blade server sales for Q2 2006: **39.5**

20. HP's percentage share of blade server sales for Q2 2006: **38.9**

Sources: 1–9: Netcraft.com, October 8, 2006 | 10: codesearch.google.com | 11, 12: Koders.com | 13: Krugle.com | 14–20: International Data Corp.

—Doc Searls

They Said It

I'd much rather pay for DRM-free music than get copy-protected music for free.
—MIKE ARRINGTON,
www.techcrunch.com/2006/10/07/allofmp3-outsources-marketing-to-us-government

All the creativity, customer whims, long tails, and money are at the network's edge. That's where chipmakers find the volumes that feed their Moore's law margins. That's where you can find elastically ascending revenues and relentlessly declining costs.
—ANDY KESSLER,
www.wired.com/wired/archive/14.10/cloudware.html?pg=5&topic=cloudware&topic_set=

The supermodel couldn't find a rat to eat.
—SAID BY SOMEBODY AT THE FREEDOM TO CONNECT CONFERENCE

FREE T-SHIRT!

Wanted: 500 loyal *Linux Journal* readers to complete a comprehensive magazine survey. Upon completion and return of your survey, we will send you a FREE, limited-edition *LJ* T-shirt to thank you for helping us.

If you are interested in participating, please send your full name and mailing address to survey@linuxjournal.com. We will respond to the first 500 e-mails we receive.

USER FRIENDLY by J.D. "Illiad" Frazer

LINUX JOURNAL EDITION

ERWIN, WHAT IS THAT?

IT'S A HYPERBOLIC TRAJECTORY PAYLOAD WITH AN EMBEDDED LAUNCH TRIGGER SYSTEM.
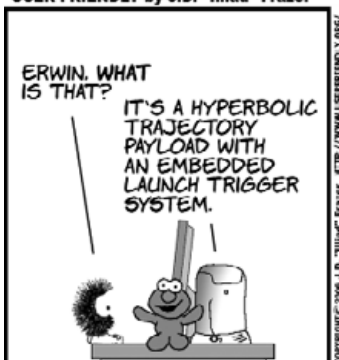
IF YOU SQUEEZE THE PAYLOAD, THE EMBEDDED SYSTEM EMITS SOUNDS THAT INCLUDE GIGGLES AND CUTESY PHRASES, THEREBY CAUSING SOMEONE TO HURL IT AWAY FROM THE EARTH.

IT'S A "TICKLE-ME-ELMO" DOLL...

CONSIDER OURSELVES LUCKY THAT IT DOESN'T FORM THE FOUNDATION OF UBIQUITOUS COMPUTING.

Tech Tips with Gnull and Voyd

CHESTER GNULL AND LAVERTA VOYD

Howdy. My husband is Chester Gnull and I'm Laverta Voyd, and I'm the lady to light a way for all you sweethearts out there who do fancy stuff with Linux. Me and my husband's gonna be bringing you tech tips just about every month now. I reckon you and yours are wondering why my husband's and me's last names don't match. Well, Chester don't like much in the way of attention, so he got the editor to change our last names so's we don't get no pesky e-mails or people messin' with us in real life.

I don't know nothing about Linux. Chester, he's the smart one, but he's not much of a talker. That's why I'm here. He don't do nothing without me, and I don't mind much cause I like talkin' and I like hosting. Chester don't understand why we gotta talk at all, but that's what the editor wants, and well, he's paying us, so we figure there ain't nothing wrong with that. So those *LJ* folks are gonna send us the tips, my Chester does the pickin' and I do the hosting. And, I say, I do love hosting, but seeing as this here's just writing stuff, we ain't gonna be serving up none of my specials like biscuits and gravy with sausage and real maple syrup, and it's all homemade but the maple syrup. But they tell me the tips are just as tasty to you Linux folk. That don't make much sense to me, but Chester says that's how it is and I believe my Chester.

Now honeys, we got some tips to start. One tip is by the editor to get things rolling. He don't get no $100 but I figure he gets enough just being editor. So, we want you to send us some of your tips. If we put your tech tip in this here column, you get $100. We know that ain't gonna get you no Fleetwood mobile home, and I'm talkin Park Models, not even them fancy Entertainer Models with two bathrooms. But $100 will get you some good eats at your local Piggly Wiggly. So send them tips in, sweethearts, and we'll appreciate it real nice. You send 'em on in to techtips@linuxjournal.com and the editors will pass 'em on to Chester for ya, and we'll do the rest.

Modify initrd to Make 3ware RAID the First Serial Device

› *This tip makes Ubuntu see a 3ware RAID controller as the first serial device on your system in Ubuntu.—Chester*
› *As you can see, Chester's real wordy, huh? That's why he's wrangled me into doing this. I mean, he's my lovin' man and I know that 'cause he shows me. But it wouldn't kill ya to say three little words now and again, would it, Chester?—Laverta*
› *Three little words. Happy?—Chester*

You can install a RAID card in your PC and configure the BIOS to make the BIOS consider the RAID card to be the first SCSI device on your system. But, Ubuntu (and probably other distributions) do not necessarily respect your BIOS settings. For example, I have an ASUS M2N32 WS Professional motherboard, which includes a PCI-X slot for the 3ware 9550SX-4LP RAID card. I can set the BIOS to make this card the first device. However, if I add a SATA drive, the Ubuntu initrd will see the onboard SATA as the first SCSI device on the system, in spite of the BIOS settings.

There may be a kernel boot parameter to override this behavior, but I haven't found one that works. Regardless, I like the following solution if for no other reason than it teaches one how to extract, modify an Ubuntu initrd and then reassemble it for use.

Here's why the Ubuntu initrd defies the BIOS settings. The initrd for Ubuntu runs the script shown in Listing 1.

The following line, which discovers storage controllers, happens to discover the NVIDIA SATA first:

```
/sbin/udevplug -s -Bpci -Iclass=0x01*
```

You can force this script to find the 3ware controller first by adding a line that explicitly loads the 3ware module before this line. Listing 2 shows how to modify the script to do that (Listing 2 is only an excerpt from the relevant part of the script).

Listing 1. The initrd scripts/local-top/udev File

```sh
#!/bin/sh -e
# initramfs local-top script for udev

PREREQ=""

# Output pre-requisites
prereqs()
{
        echo "$PREREQ"
}

case "$1" in
    prereqs)
        prereqs
        exit 0
        ;;
esac


# Each call to udevplug can take up to three minutes
if [ -x /sbin/usplash_write ]; then
    /sbin/usplash_write "TIMEOUT 540"
    trap "/sbin/usplash_write 'TIMEOUT 15'" 0
fi

# Load drivers for storage controllers found on the
# PCI bus; these show up the same for both IDE and
# SCSI so there's no point differentiating between
# the two. Do it in serial to try to provide some
# predictability for which wins each time.
/sbin/udevplug -s -Bpci -Iclass=0x01*

# We also need to load drivers for bridges (0x06),
# docking stations (0x0a), input devices (0x09),
# serial devices (0x0c) and "intelligent" devices
# (0x0e). This is both to support filesystems on the
# end and just in case there's a keyboard on the end
# and things go wrong.
/sbin/udevplug -Bpci -Iclass=0x0[69ace]*

# If we're booting from IDE, it might not be a PCI
# controller, but might be an old-fashioned ISA
# controller; in which case we need to load ide-generic.
/sbin/modprobe -Qb ide-generic
/sbin/udevplug -W
```

```
/sbin/modprobe 3w-9xxx

# Load drivers for storage controllers found on the
# PCI bus; these show up the same for both IDE and
# SCSI, so there's no point differentiating between
# the two. Do it in serial to try to provide some
# predictability for which wins each time.
/sbin/udevplug -s -Bpci -Iclass=0x01*
```

This forces the script to discover the 3ware RAID card first and assign it as /dev/sda before udevplug discovers the rest of the PCI storage controllers.

The trick here is that you need to unpack the default initrd file that comes with Ubuntu, modify this script, and then repack it and use it instead of the default initrd.

Here's one way to do that. These instructions assume you are using Ubuntu Dapper AMD64 with the kernel 2.6.15-27-amd64-generic. If you're using some other kernel, you must change the command accordingly. You can be more careful than I have been with these instructions and use sudo for all the appropriate commands. However, I jumped into a root shell with the `sudo -s -H` command to make this easier to read:

```
$ sudo -s -H
(enter password)
# cd /root
# mkdir initrd-tmp
# cd initrd-tmp
# gzip -dc /boot/2.6.15-27-amd64-generic | cpio -id
```

This unpacks your initrd so that you can manipulate its contents. Now, edit this file. (Use whichever editor suits you. I am using vi as an example.)

```
# vi scripts/local-top/udev
```

This is the file that contains the code in Listing 1. Add the modprobe command as shown in Listing 2. Save the file.

All this assumes that the module 3w-9xxx exists in your initrd. If it doesn't, or you need some other module in your initrd, you'll have to copy it to the following location (once again, this assumes you are using the 2.6.15-27-amd64-generic kernel—modify as necessary for your setup):

```
# cp <module> /root/initrd-tmp/lib/modules/
➥2.6.15-27-amd64-generic/kernel/drivers/scsi
```

Now you need to repack the initrd file. I suggest that you name this initrd something other than the original, so that if you have done something wrong, you can revert to the original easily.

Here is how to repack the file to a new initrd. This assumes your current working directory is still /root/initrd-tmp:

```
# find . | cpio --quiet --dereference -o -H newc | gzip
➥-9 > /boot/2.6.15-27-amd64-generic-3w
```

Now change your bootloader to add another boot option to use the new initrd file. You can replace the existing boot entry, but that's asking for trouble (although GRUB, for example, lets you edit a boot entry at boot time, so there's always hope if you use GRUB). If you use GRUB, specify the modified initrd as the initrd image, like this:

```
initrd    /boot/initrd.img-2.6.15-27-amd64-generic-3w
```

Reboot, and try it out.

This should work for cards other than the 3ware if you are having the same problem with another RAID card (or even some other storage card). All you have to do is change /sbin/modprobe to load the appropriate module for your card. Don't forget to check to see whether the driver module exists in the unpacked initrd before you pack it again.
—**Nicholas Petreley**

Knoppix Does More Than Show Off Linux to Windows Users

❯*Your computer won't boot because you been using one of them unofficial kernels, I bet. That'll get you in a heap of trouble. It's yer own fault. Boot a Linux live CD to fix the damage you did.—Chester*

It happens to the best of us, you sit at your computer in the morning, turn it on, and find that it won't boot properly. After an hour of troubleshooting, diagnostics and grumbling, you come to the conclusion that something about your hard drive is toast. You think of all the files you may have just lost in the process and curse the fact that you didn't back up diligently enough.

Most of the time when your OS is dead, your files are still intact on the drive; you just have to find a way to get to them. In some cases, your

problem may be that the root partition is too corrupted to mount it, but not so corrupt that you can't restore it. For example, your root partition may be formatted as XFS, and all you need to do is run a utility like xfs_repair on the partition to get things back in order.

Some distributions come with a repair disk, and some installation disks have a repair option. But, you might find it more useful to boot to a live CD to make repairs, because a live CD may put more utilities at your disposal than a repair disk. Knoppix is one of many live CD versions of Linux that runs straight from the CD and allows you access to the hard drives.

Even if you are in a worst-case scenario and have to recover individual files, all you need to recover files, or possibly the entire contents of the hard drive, is a copy of Knoppix (or your favorite live CD distro) and a portable hard drive, jump drive or some other kind of USB portable storage device. Or, if you have an unused SATA or IDE spot in your system, you always can open up the computer and plug in the extra drive (properly configured, of course). If you go portable, then how big the portable storage device is depends on how much you want to save.

Double-check the BIOS on your target computer to make sure it is set to boot from a CD. If your BIOS allows you to interrupt the boot sequence with the Esc key, F8, or some other key in order to choose which drive to boot, you may not even have to reconfigure your BIOS. Regardless, boot from CD, and Knoppix should boot up automatically into the desktop.

Once in the desktop, all that's left to do is search the computer's hard drive and find the files to salvage and transfer to your portable media device or additional internal device. Finding the files will require that you know where the file is on the hard drive, and this will be more or less difficult depending on the filesystem on the drive that was corrupted.
—*Brad Hall*

Finding Disk Space and inode Hogs

❯ *I knowed somebody was gonna get to this problem sooner or later. You get too many inodes on your system, and you're asking for another heap of trouble. This tells you how to find out and fix it.—Chester*

One of the most common tasks of a system administrator is storage management. When you're faced with a full or almost full filesystem, it's good to have a few tools at your disposal to help figure out "where" the hog is.

Searching for space hogs is very easy. With one simple command you can total up the contents of every directory in a tree (/usr in this case) and see which are the largest:

```
du -k /usr/ |sort -n |tail -30

167156  /usr/include
168960  /usr/share/icons
173972  /usr/share/texmf
244332  /usr/bin
263144  /usr/lib/openoffice.org2.0
265492  /usr/share/doc
344536  /usr/share/locale
1223992 /usr/lib
1959412 /usr/share
4159996 /usr/
```

As you can see, /usr/share/ and /usr/lib/ are pretty big, and you can drill down further by going up the list.

A somewhat rarer situation is running out of inodes in a filesystem. In this case, you will see available space, but the system will be unable to write new files because it has run out of inodes. To find inode hogs, use this quick Perl script named inodu:

```perl
#!/usr/bin/perl -w

my $start=$ARGV[0];

foreach $object (`find '$start'`){
        my @parts=split(/\//,$object);
        while(pop(@parts)  ){
                my $object = join('/',@parts);
                $object =~ s/\/+/\//g;
                $object2qty{$object}++;
        }
}

foreach $object (sort { $object2qty{$a} <=> $object2qty{$b} }
                       keys %object2qty){
        print $object2qty{$object} . "\t${object}\n";

}
```

This will total up the number of filesystem objects in each directory and supply an output much like the previous example. Use it like this:

```
cd /usr

./inodu .

10420   ./include
10973   ./share/texmf
12012   ./share/man
13207   ./share/doc
14953   ./share/icons
16481   ./src/kernels
17201   ./src
22982   ./lib
105527  ./share
174270  .
174271
```

As you can see, again share and lib are the inode hogs using more than 100,000 inodes!

If you find yourself in any of these situations, there are a number of ways to create more free space or inodes. First, look for log files that can be purged, moved or compressed. Ask users to clean up their home directories. Remove any unnecessary software. If you are using Linux LVM and ext3fs, you can expand the filesystem using lvresize and resize2fs to grow a filesystem. This creates more free space and inodes, but only if you have free space in your volume group. If you have free disk space, you can create a new partition (for, say, your /var tree), move the files to that partition and mount it as /var. As a last resort, you can move files and directories and use symlinks so the old path still works. I say "last resort" because this method can get out of hand very quickly and can make things very confusing.
—*Matthew Hoskins*

Credits
- Nicholas Petreley is Editor in Chief of *Linux Journal*.
- Brad Hall lives in Jacksonville, Florida, with his pet chickens and life-size cardboard cutout of *Star Trek: DS9*'s Dr. Bashir.
- Matthew Hoskins is Senior Information Systems Analyst at the New Jersey Institute of Technology.

Linux Journal pays $100 US for any tech tips we publish. Send your tips with your contact information to techtips@linuxjournal.com.∎

Prototype

Prototype eases the burden of using JavaScript in Ajax.

REUVEN M. LERNER

During the last few months, we have looked at ways to use JavaScript, a version of which is included in nearly every modern Web browser. For most of its life, JavaScript has been used to create simple client-side effects and actions on Web pages. But during the past year or two, JavaScript has taken center stage as part of the Ajax (Asynchronous JavaScript and XML) paradigm. It is no longer enough to create Web applications that reside on the server. Modern Web applications must include Ajax-style behavior, which probably means integrating JavaScript into the mix of server-side programs, HTML and relational databases.

As we have seen in the last few installments of this column, however, using JavaScript requires a fair amount of repeated code. How many times must I invoke document.getElementById(), just to grab nodes that I want to modify? Why must I create a library that handles the basic Ajax calls that I will be making on a regular basis? Must I create all of my own widgets and graphic effects?

Fortunately for Web developers everywhere, the explosive interest in Ajax has led to equally productive work on libraries to answer these questions and needs. Many of these libraries have been released under open-source licenses and are thus available for Web developers to include in a variety of different types of sites.

This month, we look at one of the best-known JavaScript libraries, known as Prototype. Prototype, developed by Sam Stephenson (a member of the Ruby on Rails core team), has been included in all copies of Ruby on Rails for some time. Prototype aims to make it easier to work with JavaScript, offering a number of shortcuts for some of the most common uses.

> Prototype aims to make it easier to work with JavaScript, offering a number of shortcuts for some of the most common uses.

Getting and Using Prototype

If you are using Ruby on Rails for your Web applications, Prototype is already included. You can begin to use it in your applications by adding the following inside a Rails view template:

```
<%= javascript_include_tag 'prototype' %>
```

Listing 1. simpletext.html

```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>Title</title>

    <script type="text/javascript">
        function removeText(node) {
            if (node != null)
            {
                if (node.childNodes)
                {
                    for (var i=0 ; i < node.childNodes.length ; i++)
                    {
                        var oldTextNode = node.childNodes[i];
                        if (oldTextNode.nodeValue != null)
                        {
                            node.removeChild(oldTextNode);
                        }
                    }
                }
            }
        }

        function appendText(node, text) {
            var newTextNode = document.createTextNode(text);
            node.appendChild(newTextNode);
        }


        function setText(node, text) {
            removeText(node);
            appendText(node, text);
        }

        function setHeadline () {
            var headline = document.getElementById("headline");
            var fieldContents = document.forms[0].field1.value;
            setText(headline, fieldContents);
        }
    </script>
  </head>
  <body>
    <h2 id="headline">Simple form</h2>
    <form id="the-form" action="/cgi-bin/foo.pl" method="post">
        <p>Field1: <input type="text" id="field1" name="field1" /></p>
        <p><input type="button" value="Change headline"
            onclick="setHeadline()"/></p>
    </form>
  </body>
</html>
```

If you are not using Rails, you still can use Prototype. Simply download it from its site (see the on-line Resources). Then use:

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

The above assumes, of course, that you have put prototype.js in the /javascript URL on your Web server. You might have to adjust that URL to reflect the configuration of your system.

Once you have included Prototype, you can start to take advantage of its functionality right away. For example, Listing 1 shows simpletext.html. This file contains some simple JavaScript that changes the headline to the contents of the text field when you click on the submit button.

We do this by defining a function (setHeadline) and then by setting that function to be invoked when we click on the button:

```
<p><input type="button" value="Change headline"
    onclick="setHeadline()"/></p>
```

Now, what happens inside setHeadline? First, we grab the node containing the headline:

```
var headline = document.getElementById("headline");
```

Then, we get the contents of the text field, which we have called field1:

```
var fieldContents = document.forms[0].field1.value;
```

Notice how we must grab the value by going through the document hierarchy. First, we get the array of forms from the document (document.forms), then we grab the first form (forms[0]), then we grab the text field (field1), and then we finally get the value.

Now we can set the value of the headline by attaching a text node to the h2 node. We do this with a function called setText, which I have included in simpletext.html; setText depends in turn on removeText and appendText, two other helper functions that make it easy to work with text nodes in JavaScript.

All of this is very nice and is typical of the type of JavaScript coding I often do. How can Prototype help us? By simplifying our code using two built-in functions. The first, $(), looks a bit strange but is legitimate—its full name is $ (dollar sign), and it performs much the same task as document.getElementById, returning the node whose ID matches its parameter. The second, $F, returns the value from the form element whose ID matches the parameter.

In other words, we can rewrite our function as:

```
function setHeadline() {
var headline = $("headline");
var fieldContents = $F("field1");
setText(headline, fieldContents);
}
```

Sure enough, this works just as well as the previous version. However, it's a bit easier to read (in my opinion), and it allows us to avoid traversing the document hierarchy until we

Listing 2. simpletext-prototype.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>Title</title>

    <script type="text/javascript" src="prototype.js"></script>
    <script type="text/javascript">
        function setHeadline() {
            Element.update($("headline"), $F("field1"));
        }
    </script>
  </head>
  <body>
    <h2 id="headline">Simple form</h2>
    <form id="the-form" action="/cgi-bin/foo.pl" method="post">
        <p>Field1: <input type="text" id="field1" name="field1" /></p>
        <p><input type="button" value="Change headline"
            onclick="setHeadline()"/></p>
    </form>
  </body>
</html>
```

reach the form element.

We can improve our code even further by removing our setText, updateText and removeText functions, all of which were included simply because JavaScript doesn't provide any easy way to manipulate the text of a node. But Prototype does through its Element class, allowing us to rewrite setHeadline as:

```
function setHeadline() {
    Element.update($("headline"), $F("field1"));
}
```

The code invokes Element.update, handing it two parameters: the node whose text we want to modify and the text we want to insert in place of the current text. We have just replaced 30 lines of our code with one line, thanks to Prototype. You can see the result in Listing 2.

The $() function is more than merely a terse replacement for document.getElementById(). If we hand it multiple IDs, it returns an array of nodes with those IDs. For example, we can add a second headline and then set them both with the following code:

```
function setHeadline() {
    var headlines = $("headline", "empty-headline");

        for (i=0; i<headlines.length; i++)
        {
            Element.update(headlines[i], $F("field1"));
        }
}
```

Whereas there is only text in the headline node when the page is loaded, pressing the button results in setting both headline and empty-headline to the contents of the field1 field.

```
Listing 3. simpletext-each.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>Title</title>

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

    function setHeadline() {
        var headlines = $("headline", "empty-headline");

            headlines.each(
                function(headline, index) {
                    Element.update(headline, index + " " + $F("field1"));
                }
            );
    }
    </script>
  </head>
  <body>
    <h2 id="headline">Simple form</h2>
    <h2 id="empty-headline"></h2>
    <form id="the-form" action="/cgi-bin/foo.pl" method="post">
        <p>Field1: <input type="text" id="field1" name="field1" /></p>
        <p><input type="button" value="Change headline"
            onclick="setHeadline()"/></p>
    </form>
  </body>
</html>
```

Doing More with Prototype

Prototype brings much more to the table than $(), $F() and a few convenience classes. You can think of it as a grab-bag of different utility functions and objects that make JavaScript coding easier.

For example, in our above definition of setHeadline, we had the following loop:

```
for (i=0; i<headlines.length; i++)
{
    Element.update(headlines[i], $F("field1"));
}
```

This should look familiar to anyone who has programmed in C, Java or Perl. However, modern programming languages (including Java) often support enumerators or iterators, for more expressive and compact loops without an index variable (i, in the above loop). For example, this is how we can loop over an array in Ruby:

```
array_of_names = ['Atara', 'Shikma', 'Amotz']
array_of_names.each do |name|
    print name, "\n"
end
```

Prototype brings Ruby-style loops to JavaScript, by defining the Enumerator class and then providing its functionality to the built-in Array object. We thus could rewrite our setHeadline function as:

```
function setHeadline() {
    var headlines = $("headline", "empty-headline");

        headlines.each(
            function(headline) {
                Element.update(headline, $F("field1"));
            }
        );
}
```

This code might look a bit odd, half like Ruby and half like JavaScript. In addition, it might seem strange for us to be defining a function inside of a loop, which is itself executing inside of a function. However, one of the nice features of JavaScript, like many other modern high-level languages, is that functions are first-class objects, which we can create and pass around exactly like any other type of object. Just as you wouldn't be nervous about creating an array inside of a loop, you shouldn't be nervous about defining a function inside of a loop.

I should also note that the each method provided by Prototype's Enumerated object takes an optional index argument, which counts the iterations. So, we can say:

```
function setHeadline() {
    var headlines = $("headline", "empty-headline");

        headlines.each(
            function(headline, index) {
                Element.update(headline, index + " " + $F("field1"));
            }
        );
}
```

Now, each headline will appear as before, but with a number prepended to the text. Listing 3 shows the resulting page.

Prototype provides additional methods for Enumerable objects, such as all find (to locate an object for which a function returns true); inject (to combine the items using a function, useful for summing numbers); min/max (to find the minimum or maximum value in a collection); and map (to apply a function to each member of a collection). These methods are available not only for arrays, but also for Hash and ObjectRangle, two classes that come with Prototype.

Ajax

One of the most common reasons for the recent interest in JavaScript is the growing interest in Web applications that incorporate Ajax techniques. As we have seen in the last few installments of this column, Ajax is nothing more than 1) creating an XmlHttpRequest object, 2) writing a function that sends the HTTP request with that object, 3) setting the event handler to invoke that function, and 4) writing a function that is invoked when the HTTP response returns. It isn't particularly difficult to deal with all of these things in code, but why should you be creating XmlHttpRequest objects at all, when you could be concentrating on higher-level concerns?

Fortunately, Prototype includes objects and functionality that make Ajax programming quite easy. For example, last month's column showed how we could use Ajax to check whether a user name was already taken when an individual registers for a Web site, which I show in Listing 4. The idea is that when someone enters a user name, we immediately fire

Listing 4. post-ajax-register.html

```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>Register</title>

    <script type="text/javascript">
    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 removeText(node) {
        if (node != null)
        {
        if (node.childNodes)
        {
            for (var i=0 ; i < node.childNodes.length ; i++)
            {
            var oldTextNode = node.childNodes[i];
            if (oldTextNode.nodeValue != null)
            {
                node.removeChild(oldTextNode);
            }
            }
        }
        }
    }

        function appendText(node, text) {
            var newTextNode = document.createTextNode(text);
            node.appendChild(newTextNode);
        }


        function setText(node, text) {
            removeText(node);
            appendText(node, text);
        }

        var xhr = getXMLHttpRequest();

        function parseResponse() {

            // Get variables ready
            var response = "";
            var new_username = document.forms[0].username.value;
            var warning = document.getElementById("warning");
            var submit_button = document.getElementById("submit-button");

            // Wait for the HTTP response
        if (xhr.readyState == 4) {
        if (xhr.status == 200) {
            response = xhr.responseText;

                switch (response)
                {
                case "yes":
                    setText(warning,
                        "Warning: username '" +
                        new_username +"' was taken!");
                    submit_button.disabled = true;
                    break;

                case "no":
                    removeText(warning);
                    submit_button.disabled = false;
                            break;

                case "":
                    break;

                default:
                    alert("Unexpected response '" + response + "'");
                        }

        }
        else
        {
            alert("problem: xhr.status = " + xhr.status);
        }
        }
        }

        function checkUsername() {
            // Send the HTTP request
        xhr.open("POST", "/cgi-bin/check-name-exists.pl", true);
        xhr.onreadystatechange = parseResponse;

        var username = document.forms[0].username.value;
            xhr.send("username=" + escape(username));
        }

    </script>
  </head>
  <body>
    <h2>Register</h2>
    <p id="warning"></p>
    <form action="/cgi-bin/register.pl" method="post"
enctype="application/x-www-form-urlencoded">
        <p>Username: <input type="text" name="username"
onchange="checkUsername()" /></p>
        <p>Password: <input type="password" name="password" /></p>
        <p>E-mail address: <input type="text" name="email_address" /></p>
        <p><input type="submit" value="Register" id="submit-button" /></p>
    </form>
  </body>
</html>
```

off a request to the server. The server's response will tell us whether the user name has been taken. We invoke our Ajax request by setting the username field's onchange event handler to invoke checkUsername:

```
function checkUsername() {
    // Send the HTTP request
xhr.open("POST", "/cgi-bin/check-name-exists.pl", true);
xhr.onreadystatechange = parseResponse;

var username = document.forms[0].username.value;
    xhr.send("username=" + escape(username));
}
```

Unfortunately, getting to this point requires that we have already defined xhr to be an instance of our XmlHttpRequest object, which we do as follows:

```
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;
}

    var xhr = getXMLHttpRequest();
```

Prototype can remove much of the previous code, making it possible not only to reduce the clutter in our Web pages, but also to think at a higher level of abstraction. Just as text processing becomes easier when we think about strings rather than bits and characters, Ajax development becomes easier when we no longer need to worry about instantiating various objects correctly or keep track of their values.

We can rewrite checkUsername to take advantage of Prototype as follows:

```
function checkUsername()
{
    var url =
"http://www.lerner.co.il/cgi-bin/check-name-exists.pl";

var myAjax = new Ajax.Request(
    url,
    {
        method: 'post',
        parameters: $F("username"),
        onComplete: parseResponse
    });
}
```

In the above function, we define two variables. One of them, url, contains the URL of the server-side program to which our Ajax

---

Listing 5: ajax-register-prototype.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>Register</title>

    <script type="text/javascript" src="prototype.js"></script>
    <script type="text/javascript">
        function parseResponse(originalRequest) {

            var warning = $("warning");
            var submit_button = $("submit-button");

        switch (originalRequest.responseText)
        {
        case "yes":
            Element.update(warning,
                        "Username '" + $F("username") +"' is taken!");
            submit_button.disabled = true;
            break;

        case "no":
            Element.update(warning, "");
            submit_button.disabled = false;
            break;

        case "":
            break;

        default:
            alert("Unexpected response '" +
                            originalRequest.responseText + "'");
        }
    }

        function checkUsername()
        {
        var url =
"http://maps.lerner.co.il/cgi-bin/check-name-exists.pl";

            var myAjax = new Ajax.Request(
            url,
            {
            method: 'get',
            parameters: "username=" + $F("username"),
            onComplete: parseResponse
            }
            );
        }

    </script>
  </head>
  <body>
    <h2>Register</h2>
    <p id="warning"></p>
    <form action="/cgi-bin/register.pl" method="post"
enctype="application/x-www-form-urlencoded">
        <p>Username: <input type="text" id="username" name="username"
onchange="checkUsername()" /></p>
        <p>Password: <input type="password" name="password" /></p>
        <p>E-mail address: <input type="text" name="email_address" /></p>
        <p><input type="submit" value="Register" id="submit-button" /></p>
    </form>
  </body>
</html>
```

request will be submitted. The second variable is myAjax, which is an instance of Ajax.Request. When we create this object, we pass it our url variable, as well as an object in JSON (JavaScript Object Notation) format. This second parameter tells the new Ajax.Request object what request method and parameters to pass, as well as what function to invoke upon a successful return.

It might seem as though we have simply rewritten the original version of checkUsername. But, when you consider the changes we now can make to parseResponse, you'll see how much simpler Prototype makes our lives:

```
function parseResponse(originalRequest) {

    var response = originalRequest.responseText;
    var new_username = $F("username");
    var warning = $("warning");
    var submit_button = $("submit-button");

switch (response)
{
case "yes":
    setText(warning,
        "Warning: username '" +
        new_username +"' was taken!");
    submit_button.disabled = true;
    break;

case "no":
    removeText(warning);
    submit_button.disabled = false;
    break;

case "":
    break;

default:
    alert("Unexpected response '" + response + "'");
}
}
```

The resulting rewrite of our program, post-ajax-register.html, is shown in Listing 5, ajax-register-prototype.html. It uses a number of features of Prototype, from simple ones, such as $(), to the Ajax request. We no longer need to wait for the response to arrive in its complete form; now we can let Prototype do the heavy lifting.

Conclusion

Several months ago, I remarked in this column that I don't very much like JavaScript. Although there still are elements of the language that I dislike, Prototype has done wonders to change my attitude toward the language. I no longer feel as bogged down in verbose syntax. Prototype has provided me with a feeling of liberation, and I'm able to concentrate on higher-level functionality rather than iterating through hierarchies of nodes or worrying about cross-browser compatibility. With a bit of practice, you also might find Prototype to be the antidote for anti-JavaScript feelings.

What's more, Prototype now sits at the base of a stack of different JavaScript libraries, such as Scriptaculous and Rico. In the coming months, we will look at what these libraries can do for your Web development, including Ajax development. We will then look at some alternatives to Prototype, which also have a great deal to offer the aspiring Ajax programmer.∎

Resources for this article: www.linuxjournal.com/article/9455.

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.

It's About Time!

To answer the classic question, "Does anybody really know what time it is?", our Chef's answer seems to be, "Yes, but you need lots and lots of clocks."

MARCEL GAGNÉ

TIP

Chef Marcel also recommends that you visit **www.marcelgagne.com/clocks.html** for links to even more great Linux clocks.

Time, François...it's all about time. Yes, I'll explain in a bit, but for now, time is running out and our guests will be here shortly. Just make sure the main server is updated against the reference time server. Forget that, *mon ami*. I'm sure it's accurate, and besides our guests have arrived! To the cellar, *immédiatement*! Head to the South wing and bring back the 2001 Châteauneuf du Pape, the Guigal we were sampling earlier this evening. I will show our guests to their tables.

Welcome, *mes amis* to *Restaurant Chez Marcel*, home of great wine, exquisite Linux and open-source fare and, of course, world-class guests. Please, sit down and make yourselves comfortable. I've sent François to the cellar to bring back tonight's wine selection, and he should be back shortly. You may notice an overabundance of timepieces on your respective desktops, so I'll start with a little movie history, by way of explanation.

I'm going to pretend that some of you are old enough to remember the 1960 George Pal movie version of H.G. Wells' *The Time Machine*. In the movie, George, the H.G. character, has a room filled with ticking clocks of every kind: cuckoo clocks, grandfather clocks—you name it—no

digital clocks though. If you don't remember that, perhaps you remember Robert Zemeckis' 1985's *Back to the Future*, starring Michael J. Fox. Doc Brown, played by Christopher Lloyd, has his own lab full of ticking clocks. Guess which one takes its inspiration from the other? Take a look at Figure 1, *mes amis*, and you'll see a Linux desktop version of George's Victorian home or Doc Brown's lab—depending on your video memory.

Some of you may be asking yourself why people would possibly want another clock on their system. After all, both KDE and GNOME have a clock embedded in their panels. Click on the clock, and a nice little calendar pops up, as in my desktop screenshot (Figure 1). Clocks are cool though, and some are more cool than others. On today's menu, I have several clocks for your enjoyment. From the super-stylish to the decidedly strange, you are bound to find something you like. Here is something you will definitely like and perhaps even love. My faithful waiter has just returned with the wine. Please pour for our guests, François.

While François pours the wine, I want you to take one more look at the clock in the lower right-hand corner of my KDE kicker panel. That's not the default KDE clock, but Fred Schättgen's StyleClock, a themeable replacement that includes an alarm clock and a countdown timer (your chef has used it to take little naps at his chair).

From the menu, you can set an alarm or a countdown timer. Both modes come with some one-click presets, but both the alarm and timer allow for a custom setting. Of course, we also can select themes for that special visual
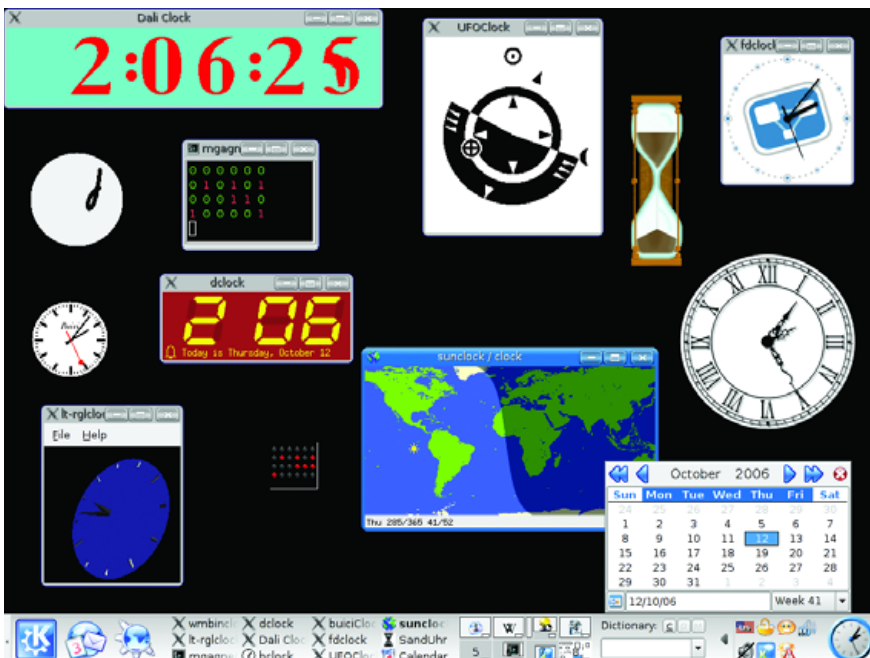
Figure 1. Does Marcel really know what time it is with all these clocks?
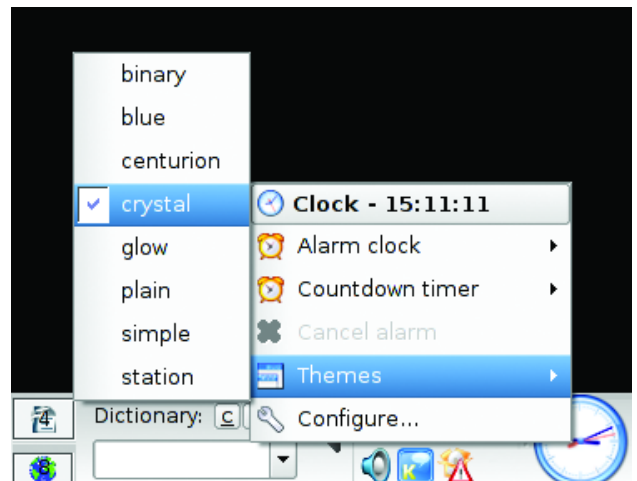
Figure 2. To configure the StyleClock, simply right-click on the clock.

Although binary clocks may be geeky, there's something cool about a nice, retro, analog clock running on your desktop.

touch. I happen to like the analog styles, but StyleClock comes with both analog and digital themes. There's also the mandatory, super-geeky, binary clock.

Speaking of binary clocks, if you've been reading this column for a long time, you know that although I tend to run a KDE desktop, I still have an enduring fondness for Window Maker and its trademark dock apps. It is for this reason that I now direct your attention to Thomas "Engerim" Kuiper and Sune Fjod's wmBinClock. This slick little Window Maker dock app can display the time vertically or horizontally (horizontal is the default). You read the time by doing binary translation of LEDs that are either on (1) or off (0). When reading the time horizontally, the seconds are the two vertical rows of LEDs on the right. The two middle LEDs are minutes and so on. This is a great little application, and no, you don't need Window Maker to run it. It works just as well under KDE or GNOME.

Maybe you aren't running a graphical display, or you have a fondness for running things in a terminal window, but you would still like a binary clock. Nico Golde's BinClock displays the time in a terminal window. By default, the time is displayed similarly to wmBinClock (Figure 3), but this is a one-time display. To run the clock continuously, you must use the -l option to loop:

```
binclock -l
```
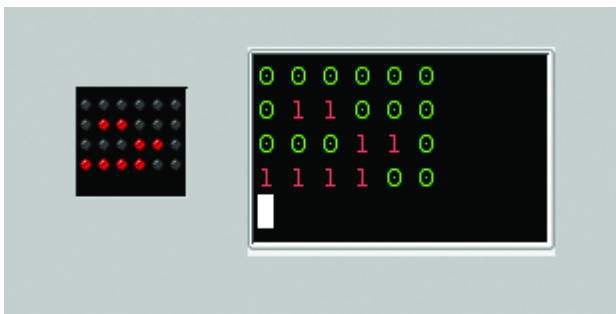

Figure 3. Two binary clocks, wmBinClock and dclock, seem to be keeping good time with each other.

Use the -h option to see a number of command-line options that let you run the clock in a single-line or traditional mode or change the color of the ones and zeros. Of course, if you want to do straight text, you simply could type the date command in your terminal window. If you want a calendar for the current month, type cal. I do, however, want to focus on the desktop.

Although binary clocks may be geeky, there's something cool about a nice, retro, analog clock running on your desktop. To avoid looking for and downloading anything, try the venerable Xclock that comes with your system's X software. This baby was originally written by Tony Della Fera, Dave Mankins and Ed Moy. To run the Xclock, simply type xclock (use your Alt-F2 program launcher or the command line). By default, it doesn't
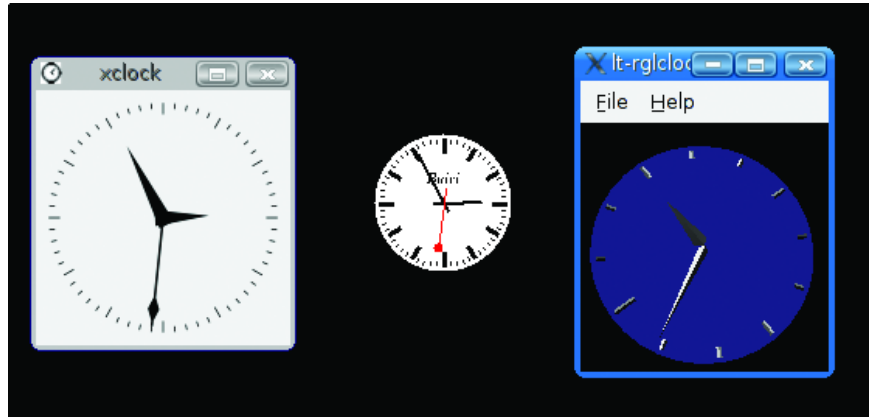
show a second hand. To activate that, type xclock -update 1. This adds a second hand that updates every second.

The Xclock hasn't changed much over the years (why mess with success?), but that lack of change got Marc Singer writing his Buici clock, a simple, yet classy clock that does nothing other than show you the time with a nice, red, sweep second hand. For those who like a little more animation than just a sweep second hand, I recommend Kaz Sasayama's rglclock. This is a rotating 3-D Mesa/OpenGL clock that you can drag with the mouse to spin in whatever direction and at whatever speed you like. All three of these are shown in Figure 4.

Taking classy to a higher plane was surely Mirco Mueller's plan when he wrote Cairo Clock. Seriously, this is a gorgeous-looking clock with several different faces, 12- and 24-hour formats and more. To change a running Cairo Clock, right-click and a menu appears letting you change not only the look of the clock, but several other attributes as well (Figure 5). You

Figure 4. Analog clocks come in many styles, from the classic Xclock on the left, followed by the simple but classy Buici clock in the center, and the spin-happy rglclock to the right.
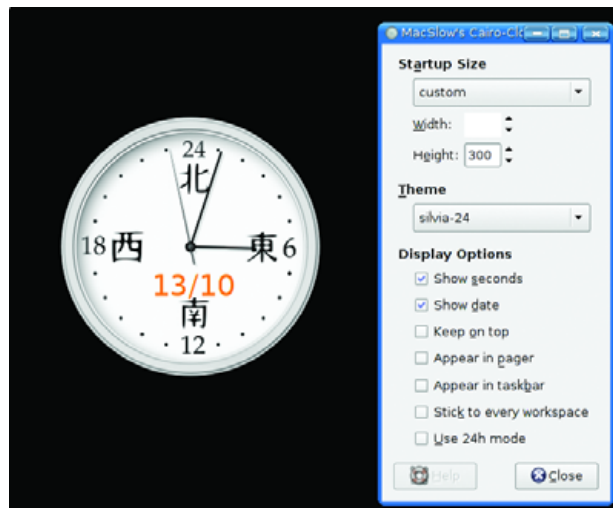
Figure 5. The highly configurable and beautiful Cairo Clock can be changed while it is running.

From the super-stylish to the decidedly strange, you are bound to find something you like. Here is something you will definitely like and perhaps even love.

even can change the size to whatever you like.

Although I've spent some time talking about analog clocks, there are some pretty cool digital clocks out there as well. One of my favorites is Jamie Zawinski's XDaliClock (Figure 6), a wonderfully strange digital clock where the numbers don't so much change, as morph. Second by second, and minute by minute, digits melt from one to the

Figure 6. The very cool XDaliClock melts away the seconds, while dclock keeps somewhat more solid time.
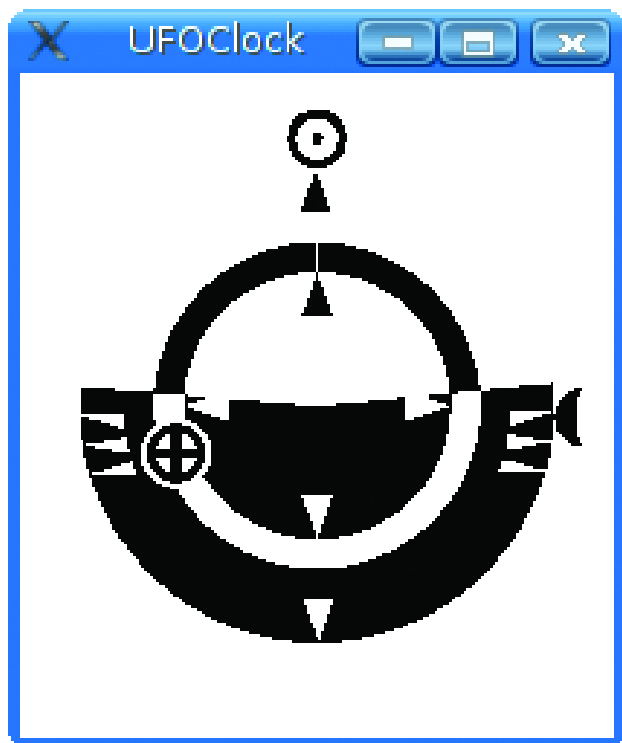
Figure 7. The UFOClock—timepiece or artifact left behind by an alien race?

other. You'll be watching this one just to see the hours change as 59 minutes and 59 seconds approaches. Use the command `xdaliclock -cycle`, and you'll not only see the numbers morph, but the background color as well.

Tim Edwards' dclock, a modification of Dan Heller's original code, is a great digital clock that looks like the old seven-segment LED display clocks. dclock has a number of command-line arguments that let you set the date format, the color of the LED segments (both on and off) and more. For instance, typing `dclock -date Today is %A, %B %d -fg yellow -bg brown -led_off brown4` generates the clock in the lower part of Figure 6. Furthermore, while the clock is running and your mouse pointer is inside the active window, you can change various settings with single keystrokes. For example, pressing the S key toggles the seconds display, R reverses the video colors and / increases the angle of the digits. Check the documentation for other one-key changes.

All this talk of clocks just makes it more apparent that closing time is fast approaching. While François refills your glasses a final time, I'll leave you with perhaps the strangest clock of all, the aptly named UFOClock by Matt Wronkiewi (Figure 7), which is also very cool and worthy of some desktop space.

The UFOClock displays the time of day, the phase of the moon, ratio of day to night, time to the beginning (or end) of twilight, and the time until the solstice or equinox. If you are asking, yes, I'm still trying to figure it all out. The distribution bundle comes with an example configuration file, so you can set the latitude and longitude of your home location (so you can tell the time of day).

And now, *that* time is officially upon us. With all these clocks, there is no way to escape the reality of closing time, and there are still so many clocks to explore. Please, raise your glasses and let us all drink to one another's health. *A votre santé! Bon appétit!* ∎

Resources for this article: www.linuxjournal.com/article/9456.

Marcel Gagné is an award-winning writer living in Waterloo, Ontario. He is the author of *Moving to Ubuntu Linux*, his fifth 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.

CLOCKYWOCK

On a related note, check out Thomas S. Glascock's Clockywock at **www.soomka.com**. This is an ncurses-based analog clock that runs in a terminal window. It's high technology meets low.

How Do People Find You on Google?

DAVE TAYLOR

Getting back to Apache log analysis by ending with a cliff-hanger.

I admit it. I got sidetracked last month talking about how you can use a simple shell script function to convert big scary numbers into more readable values that are understandable. Sidetracked because we were in the middle of looking at how shell scripts can help you dig through your Apache Web server logs and extract useful and interesting information.

This time, I show how you can ascertain the most common search terms that people are using to find your site—with a few invocations of grep and maybe a few lines of awk for good measure.

Understanding Google
For this to work, your log has to be saving referrer information, which Apache does by default. You'll know if you peek at your access_log and see lines like this:

```
195.110.84.91 - - [11/Oct/2006:04:04:19 -0600] "GET
➥/blog/images/rdf.png HTTP/1.0" 304 -
➥"http://www.askdavetaylor.com/date_math_in_linux_shell_script.html"
➥"Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)"
```

It's a bit hard to read, but this is a log entry for someone requesting the file /blog/images/rdf.png, and the referrer, the page that produced the request, is also shown as being date_math_in_linux_shell_script.html from my askdavetaylor.com site.

If we look at a log file entry for an HTML hit, we see a more interesting referrer:

```
81.208.53.251 - - [11/Oct/2006:07:32:32 -0600]
➥"GET /wicked/wicked-cool-shell-script-library.shtml
➥HTTP/1.1" 200 15656 "http://www.google.com/
➥search?q=Shell+Scripting+&hl=it&lr=&start=10&sa=N"
➥"Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0;
➥.NET CLR 1.0.3705)"
```

Let me unwrap that just a bit too. The request here is for wicked-cool-shell-script-library.html on my site (intuitive.com), based on a Google search (the referrer is google.com/search). Dig into the arguments on the Google referrer entry, and you can see that the search was "Shell+Scripting". Recall that + represents a space in a URL, so the search was actually for "Shell Scripting".

(Bonus tip: because we're at start=10, this means they're on the second page of results. So, we know the match that led this person to my site is somewhere between #11 and #20.)

Okay, so now the question is, can we extract only these searches and somehow disassemble them so we can identify the search terms quickly? Of course we can!

Extracting Google Searches
For now, let's focus only on Google's search results, but it's easy to extend this to other search engines too. They all use the same basic URL structure, fortunately:

```
$ grep 'google.com/search' access_log | head -1
168.230.2.30 - - [11/Oct/2006:04:08:05 -0600]
➥"GET /coolweb/chap14.html HTTP/1.1" 200 31508
➥"http://www.google.com/search?q=%22important+Style+Sheet+
➥Attribute.%22&hl=en&lr=" "Mozilla/4.0 (compatible;
➥MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322;
➥.NET CLR 2.0.50727; InfoPath.1)"
```

Okay, that was simple. Now, extracting only the referrer field is easily done with a quick call to awk:

```
$ grep 'google.com/search' access_log | head -1 | awk '{print $11}'
"http://www.google.com/search?q=%22important+Style+Sheet
➥+Attribute.%22&hl=en&lr="
```

Okay, closer. The next step is to chop off the value at the ? and then at the & afterward. There are a bunch of ways to do this, but I use only two calls to cut, because, well, it's easy:

```
$ grep 'google.com/search' access_log | head -1 | awk
➥'{print $11}' | cut -d\? -f2 | cut -d\& -f1
q=%22important+Style+Sheet+Attribute.%22
```

Nice! Now, we need to strip out the q= artifact from the HTML form used on Google itself, replace all occurrences of + with a space, and (a little bonus task) convert %22 into a double quote so the search makes sense. This can be done with sed:

```
$ grep 'google.com/search' access_log | head -1 |
➥awk '{print $11}' | cut -d\? -f2 | cut
➥-d\& -f1 | sed 's/+/ /g;s/%22/"/g;s/q=//'
"important Style Sheet Attribute."
```

Let me unwrap this a bit so it's easier to see what's going on:

```
grep 'google.com/search' access_log | \
  head -1 | \
  awk '{print $11}' | \
  cut -d\? -f2 | cut -d\& -f1 | \
  sed 's/+/ /g;s/%22/"/g;s/q=//'
```

Obviously, the head -1 is only there as we debug it, so when we pour this into an actual shell script, we'll lose that line. Further, let's create a variable for the name of the access log to simplify things too:

```
#!/bin/sh

ACCESSLOG="/var/logs/httpd.logs/access_log"

grep 'google.com/search' $ACCESSLOG | \
   awk '{print $11}' | \
   cut -d\? -f2 | cut -d\& -f1 | \
   sed 's/+/ /g;s/%22/"/g;s/q=//'
```

We're getting there....

Sorting and Collating

One of my favorite sequences in Linux is `sort | uniq -c | sort -rn`, and that's going to come into play again here. What does it do? It sorts the input alphabetically, then compresses duplicate lines with a preface count of how many matches are found. Then, it sorts that result from greatest matches to least. In other words, it takes raw input and converts it into a numerically sorted summary.

This sequence can be used for lots and lots of tasks, including figuring out the dozen most common words in a document, the least frequently used filename in a filesystem, the largest file in a directory and much more. For our task, however, we simply want to pore through the log files and figure out the most frequent searches that led people to our Web site:

```
#!/bin/sh

ACCESSLOG="/var/logs/httpd.logs/access_log"

grep 'google.com/search' $ACCESSLOG | \
   awk '{print $11}' | \
   cut -d\? -f2 | cut -d\& -f1 | \
   sed 's/+/ /g;s/%22/"/g;s/q=//" | \
   sort | \
   uniq -c | \
   sort -rn | \
   head -5
```

And the result:

```
$ sh google-searches.sh
 154 hl=en
  42 sourceid=navclient
  13 client=safari
   9 client=firefox-a
   3 sourceid=navclient-ff
```

Hmmm...looks like there's a problem in this script, doesn't there?

I'm going to wrap up here, keeping you in suspense until next month. Why don't you take a stab at trying to figure out what might be wrong and how it can be fixed, and next month we'll return to this script and figure out how to make it do what we want, not what we're saying it should do!■

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.

Running Network Services under User-Mode Linux, Part III

MICK BAUER

Fine-tune and firewall your UML guest systems.

In the last two Paranoid Penguin columns, I walked you through the process of building a virtual network server using User-Mode Linux. We built both host and guest kernels, obtained a prebuilt root filesystem image, configured networking on the host, and when we left off last month, we finally had booted our guest kernel with bridged networking, ready for configuration, patching and server software installation.

This month, I tie up some loose ends in our example guest system's startup and configuration, show you the uml_moo command, demonstrate how to write firewall rules on your UML host system, offer some miscellaneous security tips and give some pointers on creating your own root filesystem image. And, can you believe we will have scratched only the surface of User-Mode Linux, even after three articles? Hopefully, we'll have scratched deeply enough for you to be off to a good start!

Guest System Configuration

You may recall that last time we set up bridged networking on our host, creating a local tunnel interface called uml-conn0 that we bridged to the host system's "real" eth0 interface. If you don't have last month's column, my procedure was based on the one by David Cannings (see the on-line Resources). When we then started up our host (User-Mode) kernel, we mapped a virtual eth0 on the guest to uml-conn0 via a kernel parameter, like so:

```
umluser@host$ ./debkern ubd0=debcow,debroot
➥root=/dev/ubda eth0=tuntap,uml-conn0
```

The last parameter, obviously, contains the networking magic: eth0=tuntap,uml-conn0. It can be translated to "the guest kernel's eth0 interface is the host system's tunnel/tap interface uml-conn0". This is important to understand; to the host (real) system, the guest's Ethernet interface is called uml-conn0, but to the guest system itself, its Ethernet interface is plain-old eth0.

Therefore, if you run an iptables (firewall) rule set on either host or guest (I strongly recommend you do so at least on the host), any rules that use interface names as sources or targets must take this difference in nomenclature into account. We'll discuss some example host firewall rules shortly, but we're not quite done with guest-kernel startup parameters yet.

Going back to that startup line, we've got definitions of our virtual hard drive (ubd0, synonymous with ubda), our path to virtual root and, of course, our virtual Ethernet interface. But what about memory?

On my OpenSUSE 10.1 host system, running a UML Debian guest with the above startup line resulted in a default memory size of about 29MB—pretty puny by modern standards, especially if I want that guest system to run real-world, Internet-fac-

ing network services. Furthermore, I've got an entire gigabyte of physical RAM on my host system to allocate; I easily can spare 256MB of RAM for my guest system.

To do so, all I have to do is pass the parameter mem=256M to the guest kernel, like so:

```
umluser@host$ ./debkern mem=256M ubd0=debcow,debroot
➥root=/dev/ubda eth0=tuntap,uml-conn0
```

Obviously enough, you can specify however much more or less than that as you like, and you can allocate different amounts of RAM for multiple guests running on a single host (perhaps 128M for your virtual DNS server, but 512M for your virtual Web server, for example). Just be sure to leave enough non-guest-allocated RAM for your host system to do what *it* needs to do.

Speaking of which, you'll save a lot of RAM on your host system by not running the X Window System, which I've always recommended against running on hardened servers anyhow. The X server on my test host uses around 100MB, with actual desktop managers requiring more. On top of this, the X Window System has a history of security vulnerabilities with varying degrees of exploitability by remote attackers (remember, a "local" vulnerability ceases being local the moment any non-local user starts a shell).

Managing COW Files

If, as I recommended last month, you run your UML guest with a Copy on Write (COW) file, you may be wondering whether your UML guest-kernel startup line is the only place you can manage COW files. (A COW file is created automatically when you specify a filename for one in your ubd0=... parameter.)

Actually, the uml-utilities package includes two standalone commands for managing COW files: uml_moo and uml_mkcow. Of the two, uml_moo is the most likely to be useful to you. You can use uml_moo to merge all the filesystem changes contained in a COW file into its parent root filesystem image.

For example, if I run the example UML guest kernel startup command described earlier, and from within that UML guest session I configure networking, apply all the latest security patches, install BIND v9 and configure it and finally achieve a "production-ready" state, I may decide that it's time to take a snapshot of the UML guest by merging all those changes (written, so far, only into the file debcow) into the actual filesystem image (debroot). To do so, I'd use this command:

```
umluser@host$ uml_moo ./debcow newdebroot
```

The first argument you specify to uml_moo is the COW file you want to merge. Because a COW file contains the name of

> You can use uml_moo to merge all the filesystem changes contained in a COW file into its parent root filesystem image.

the filesystem image to which it corresponds, you don't have to specify this. Normally, however, you should specify the name of the new filesystem image you want to create.

My example uml_moo command, therefore, will leave the old root filesystem image debroot intact (maybe it's also being used by other UML guests, or maybe I simply want to preserve a clean image), creating a new filesystem named newdebroot that contains my fully configured and updated root filesystem.

If I want to do a hard merge, however, which replaces the old filesystem image with the merged one (with the same filename as before), perhaps because my hard disk is too full for extra image files, I'd instead use `uml_moo -d ./debcow` (the -d stands for destructive merge).

iptables and UML

Whether you chroot your User-Mode guests, and whether you use SELinux, depends on how deep you want your layers of security to go and how much time and effort you're able to expend. However, I strongly recommend that on any Internet-facing, bridged User-Mode Linux system, you use iptables on your UML host to restrict your guest systems' network behavior.

On the one hand, if your UML system already resides outside a firewall in a DMZ network (as should any Internet server), you're already protecting your internal network from the possibility of a network server compromise. However, there's really no good reason not to take the opportunity also to use UML-host iptables rules to reduce the ability of an attacker to use one compromised UML guest to attack other UML guests, the UML host itself or other systems in your DMZ network.

There are two categories of rules I strongly recommend you consider. First, anti-IP-spoofing rules can help ensure that every packet sent by each guest bears the source IP address you actually assigned to that guest, and not a forged (spoofed) source IP. These are low-maintenance rules that you'll have to think about only at setup time, unless for some reason you change a guest system's IP address.

Suppose you have a UML system whose IP address is 10.1.1.10 and whose tun/tap interface is (from the host's perspective) uml-conn0. The anti-spoofing rules you install on the UML host might therefore look like that shown in Listing 1.

The first rule logs the spoofed packets; the second one actually drops them. As you may know, the LOG target doesn't cause packets to cease being evaluated against subsequent iptables rules, but the DROP target does, so the LOG rule must come before the DROP rule.

Due to space constraints, I can't launch into a primer on how to write iptables rules or how they're managed on your Linux distribution of choice. But, I can talk about the bridge-specific magic in Listing 1: the physdev iptables module and the --physdev-in parameter.

Usually, we use iptables' -i and -o flags to denote which network interface packets are received and sent from, respectively. However, when writing iptables rules on a system doing bridged networking, we need to be a bit more precise, especially when we're also using tun/tap interfaces, as eth0 then takes on a different role than in normal Layer 3 (routed) networking.

Therefore, where we might normally use `-i uml-conn0` in a rule, on a bridging host, we should instead use `-m physdev --physdev-in uml-conn0`. Similarly, instead of `-o uml-conn0`, we'd use `-m physdev --physdev-out uml-conn0`. As with other module invocations, you need only one instance

---

Listing 1. Anti-IP-Spoofing Rules

```
iptables -A FORWARD -m physdev --physdev-in uml-conn0
 ➥-s ! 10.1.1.10 -j LOG --log-prefix "Spoof from uml-conn0"

iptables -A FORWARD -m physdev --physdev-in uml-conn0
 ➥-s ! 10.1.1.10 -j DROP
```

Listing 2. Service Rules for the UML Guest

```
iptablee -A FORWARD -m state --state
 ➥RELATED,ESTABLISHED -j ACCEPT

iptables -A FORWARD -m physdev --physdev-out uml-conn0
 ➥-p udp --dport 53 -m state --state NEW -j ACCEPT

iptables -A FORWARD -m physdev --physdev-out uml-conn0
 ➥-p tcp --dport 53 -m state --state NEW -j ACCEPT

iptables -A FORWARD -m physdev --physdev-in uml-conn0
 ➥-p udp --dport 53 -d ! 10.1.1.0/24 -m state --state NEW -j ACCEPT

iptables -A FORWARD -m physdev --physdev-in uml-conn0
 ➥-p tcp --dport 53 -d ! 10.1.1.0/24 -m state --state NEW -j ACCEPT

iptables -A FORWARD -m physdev --physdev-in uml-conn0
 ➥-p udp --dport 514 -d logserver -m state --state NEW -j ACCEPT

iptables -A FORWARD -j LOG --log-prefix
 ➥"Forward Dropped by default"

iptables -A FORWARD -j DROP

iptables -A OUTPUT -d 10.1.1.10 -p tcp --dport 22 -m
 ➥state --state NEW -j ACCEPT
```

---

of `-m physdev` if a given iptables rule uses both the --physdev-in and --physdev-out rules.

After setting up a pair of anti-IP-spoofing rules, you also should create a set of "service-specific" rules that actually govern how your guest system may interact with the rest of the world, including other guest systems and the host itself.

Remember that in our example scenario the guest system is a DNS server. Therefore, I'm going to enforce this logical firewall policy:

1. The UML guest may accept DNS queries (both TCP and UDP).

2. The UML guest may recurse DNS queries against upstream (external) servers.

3. The UML guest may send its log messages to a log server (called logserver).

4. The UML host may initiate SSH sessions on the UML guest.

However, I strongly recommend that on any Internet-facing, bridged User-Mode Linux system, you use iptables on your UML host to restrict your guest systems' network behavior.

Listing 2 shows iptables commands that could enforce this policy.

Listing 2 has two parts: a complete set of FORWARD rules and a single OUTPUT rule. Because, logically speaking, UML guest systems are "external" to the UML host's kernel, interactions between UML guests and each other, and also interactions between UML guests and the rest of the world, are handled via FORWARD rules. Interactions between UML guests and the underlying host system, however, are handled by INPUT and OUTPUT rules (just like any other interactions between external systems and the host system).

Because all of my logical rules except #4 are enforced by iptables FORWARD rules, Listing 2 shows my UML host's complete FORWARD table, including an initial rule allowing packets associated with already-approved sessions, and a final pair of "default log & drop" rules. Note my use of the physdev module; I like to use interface-specific rather than IP-specific rules wherever possible, as that tends to make it harder for attackers to play games with IP headers.

The last rule in Listing 2 should, in actual practice, appear somewhere in the middle of a similar block of OUTPUT rules (beginning with an allow-established rule and ending with a default log/drop rule pair), but I wanted to illustrate that where the source or destination of a rule involves the UML host system, you can write an ordinary OUTPUT or INPUT rule (respectively) rather than a FORWARD rule.

Because your UML host is acting as an Ethernet bridge, you can write still-more-granular and low-level firewall rules—even filtering by MAC addresses, the ARP protocol and so forth. But for that level of filtering, you'll need to install the ebtables command. iptables rules of the type I've just described should, however, suffice for most bastion-host situations.

Miscellaneous Security Notes
If you patched your UML host's kernel with the SKAS patch, you've already got reasonably good assurance that an attacker who compromises a UML guest won't be able to do much, if anything, on the host system. However, I'm not one to argue against paranoia, so I also recommend you chroot your UML guest system. This is described in detail on the UML Wiki (see Resources). And, what about shell access to your UML guests? There are various ways to access "local consoles". You get one automatically when you start your UML guest from a UML host shell manually—after your UML kernel loads, you'll be presented with a login prompt.

That doesn't do you much good if you start your UML guest automatically from a script, however. The "Device

Inputs" page on the User-Mode Linux home page (see Resources) describes how to map UML guest virtual serial lines to UML host consoles. For me, however, it's easiest simply to install SSH on my UML guest system, configure and start its SSH dæmon, and create a firewall rule that allows connections to it only from my UML host.

Generally speaking, you want to use the same security controls and tools on your UML guest (tripwire, chrooted applications, SELinux, tcpwrappers and so on) as you would on any other bastion server.

Building Your Own Root Filesystem Image
Describing in detail the process of building your own root filesystem image from scratch would require its own article (one which I may yet write). Suffice it to say, the process is all but identical to that of creating your own bootable Linux CD or DVD, without the final step of burning your image file to some portable medium. There are three major steps:

1. Create an empty filesystem image file with dd.

2. Format the image file.

3. Mount it to a directory via loopback.

4. Install Linux into it.

The first three steps are the easiest. To create a 1GB ext3 image file, I'd run the commands shown in Listing 3 as root. Installing Linux into this directory gets a bit more involved, but if you've got a SUSE host system, the Software module in YaST includes a wizard called "Installation into Directory". Like other YaST modules, this is an easy-to-use GUI.

Similarly, if you run Debian, you can use the command debootstrap. See Michael McCabe and Demetrios Dimatos' handy article "Installing User Mode Linux" for detailed instructions on using debootstrap to populate your root filesystem image.

See the UML Wiki for some pointers to similar utilities in other distributions. The Linux Bootdisk HOWTO (see Resources), although not specific to UML, is also useful.

Conclusion
I hope you're well on your way to building your own virtual network servers using User-Mode Linux! The two most important sources of UML information are the UML home page and the UML Wiki (see Resources). Those and the other Web sites mentioned in this piece should help you go much further with User-Mode Linux than I can take you in an introductory series of articles like this. Have fun, and be safe!∎

Resources for this article: www.linuxjournal.com/article/9457.

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

Listing 3. Making and Mounting an Empty Filesystem Image

```
dd if=/dev/zero of=./mydebroot bs=1024K count=1000
mkfs.ext3 ./mydebroot
mkdir /mnt/debian
mount -o loop ./mydebroot /mnt/debian
```

Ode to Joy

This month, maddog gives us a little music history and some words on patents.

JON "MADDOG" HALL

I am not a musician. Although I did play the clarinet in grade school, that was a long, long time ago, and I have forgotten what I learned. Therefore, when I want to hear music, I tend to go down to the *Alideia dos Piratas*, my favorite restaurant, and listen to the musicians who play there.

One evening as I sat in my favorite seat looking out over the ocean, I must have drifted away to the music because a young friend of mine, Bryant, asked me why I was smiling.

"I remembered a story about a musical instrument that came very close to non-existence", I said.

The year was approximately 1703, and the place was Florence, Italy. An instrument maker named Bartolomeo Cristofori had a great idea for a new instrument. He was certain that the new instrument would take the market by storm, and he wanted to patent his idea. Patents as we know them today already existed in Florence, Italy, because they were introduced to Italy in 1474, more than 200 years previous. Getting a patent was a natural thing for people to do if they had invented a totally new instrument.

There was a problem, however. The new instrument was expensive to make, and there was no music for it. If there was no music, there would be no demand for the instrument. No demand for the instrument meant no sales of the instrument. And, if there were no sales of the instrument, there would be no music created. A vicious circle.

So instead of taking out a patent, the instrument maker decided to publish far and wide how to make the instrument. It took him several years to find a writer who would write about this instrument, and eventually, he found a magazine in Germany that would publish the articles. When German instrument makers saw the article, they agreed it would make a great instrument, and so they started making copies of it and giving it to the great songwriters of the day—people like Handel, Bach and (later) Mozart.

For the instrument that we are discussing, the instrument that replaced the harpsichord (which could play only one loudness of music due to having the strings plucked rather than hit), was the pianoforte (soft and loud), which we simply call the piano today.

And yet, even with all those instrument makers free to make the piano, it still took almost 100 years for the piano to replace the harpsichord as the "standard keyboard instrument".

"Wow", Bryant responded when I finished telling the story, "imagine if Cristofori had tried to patent it so that only he could make it. It might have taken another 100 years for the piano to make it in the marketplace, if it ever did. Were there ever patents associated with pianos?"

"Yes", I said, "but not always to good effect."

If you look in the back of a piano, you normally see a list of patents that are covered in that piano. Sometimes the list is 50 or more patents long. Each one of them stands for some small improvement that some person made, and often these patents were licensed out to other piano makers for a small sum. Sometimes, however, as you take apart a piano to fix it, you find some strange mechanism or technique in making the piano, and you wonder why someone did that particular thing so strangely—until you realize it was to get around a particular patent. The piano maker actually made an inferior instrument, trying hard not to pay a royalty for the patent, or because the competitor would not license the rights to that patent.

"But patents must have some good use", said Bryant. "Why would the government have created them otherwise?"

In some cases, it can be argued that patents are useful. Particularly when the concepts they are protecting are ones that took large amounts of time and money to create— medicines, for example.

There are people who graduate from college and spend their entire lives looking for an effective treatment for a disease. The search for the treatment typically requires lots of expensive equipment, lots of staff and many expensive chemicals. Once the treatment is found, there needs to be testing of the treatment, and the results have to be studied and approved. All of the expense for this research is expected to be repaid in selling the treatment. Without a patent on the medicine, other companies could duplicate what those researchers had done, and undercut the profits on the production of the medicine.

There are arguments about whether patents on medicine are too long or whether the costs of patented medicines are too high, but a lot of this could be handled by governmental programs to get medicines into the hands of the people who need them. The issue is whether the law will give an effective monopoly to the creator of the medicine long enough for them to recover the large costs they have incurred.

If some competitor wants to make the same medicine, the patent can be licensed out. There are relatively few numbers of medicine developers, and relatively few numbers of people who can produce a drug safely and effectively.

Compare this to the normal way a software patent is created today. Software developers have a problem. They study the problem a day or two, then they write a nice piece of code to solve the problem. A lawyer looks over their shoulder as they submit the code and asks the fatal question: "Do you think we can patent that?" And, before the developers can deny that it is patentable, the patent is well on its way to the patent office. I have simplified this scenario somewhat, and surely there are issues in computer science that take more effort than what I have just described, but not the orders of magnitude greater effort and cost that medical patents represent—and particularly not in today's society.

When I started programming, computers cost millions of dollars (and that was when a million US dollars was a lot of money). The programming community was relatively small. There were a couple journals on programming and algorithms. If you wanted software written for you, either you did it yourself, or you went to a relatively small number of people who could write software for you. Software was not everywhere, as it is today. And, even without the Patent Office allowing software to be patented

(software patents really got underway in 1981), concepts like microcode, compilers, databases, subroutines and so forth were created and built upon by the software community.

Then, just as the cost of hardware began to drop appreciably, and as software started to become intertwined with our lives, more and more software patents started being applied to software. Unlike the issue of medicine, or the making of steel or automobiles, everyone uses the concepts of software, and most people can create software, both for profit and nonprofit.

For those of you who are not programmers, imagine Michelangelo painting the Sistine Chapel—lying on his back, year after year, painting. Just as he finishes, his arch-enemy Leonardo da Vinci walks in and tells him that he has to start all over again, because last week he had patented a particular brush stroke that Michelangelo used a lot.

Or, imagine that as Beethoven finishes his *Symphony no. 9 in D Minor*, which included "Ode to Joy", one of the most beautiful works of music, in walks one of his greatest critics, Johann Nepomuk Hummel and signs (because Beethoven is deaf) that he has to rewrite the entire symphony because last week Hummel had patented the triplet.

Unlike a law of nature, such as gravity, patents are laws of humankind and just as we can make those laws, we can dismantle them when they have served their purposes. It is particularly necessary to dismantle the laws if the laws are now hurting the type of innovation the laws were supposed to foster.

With hundreds of programmers joining the ranks of programming every day, and thousands more people using computers every day, it is unreasonable for programmers to have to memorize the thousands of software patents that have been created. It is also unreasonable for people writing software, either as a hobby or offering at no cost to society, to have to pay legal fees (either for lawyers or for patent royalties) to someone to distribute software that they wrote.

I have nothing against copyright law. Copyright violation is relatively easy to avoid. But I believe that in the modern world software patents are hindering innovation, not helping it.

And I invited Bryant to my house to listen to my pianola, which is yet another story....■

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.
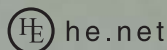
Embedded at the Edge

Visiting the grass-roots Net growing out of Copenhagen's basements.

DOC SEARLS

As I write this, I'm embedded in a hotel in Copenhagen. Besides being jet-lagged (it's past 3am and I'm still wide awake), I'm also cursing the hotel's firewall, which blocks Web sites it thinks are violent (Howard Stern), sexual (one in five results for "brassiere" on Google) or worth forbidding on other grounds (most display ads are blocked). Bandwidth (banned width?) is slow as mud. And both SMTP and SSH are blocked, so even my normal route-arounds of blocked outbound e-mail are thwarted. Naturally, I posted a gripe on my blog under the irresistible headline "Something's censored in Denmark", and instantly got back an e-mail from my Danish friend Thomas Madsen-Mygdal. He wrote, "Blame the Americans" and pointed to the Web site of WatchGuard (**www.watchguard.com**), a US-based firewall hardware company. WatchGuard runs on Linux (**www.watchguard.com/help/lss/50/handbook/need_f27.htm**).

So, I'm faced with the ironic task of writing about embedded Linux while my access to the whole Internet is throttled by an embedded Linux product. Fortunately, I spent much of yesterday morning watching embedded (and other forms of) Linux being put to good use in the grass roots of applied tech in Copenhagen's urban habitat.

The object of my interest was Indienet.dk, a local pure-Internet infrastructure provider. Although they support VoIP phones, they don't sell VoIP as a service. Nor do they sell TV. (Those last two combine with Net service to form the familiar "triple play" of offerings that telcos, cablecos and many of their new muni competitors are pushing in the

US.) "We're bottom level", says Jakob Frederiksen, our host and the company's sales and service chief. "We just provide the base-level connectivity."

Indienet is better known for its vans than its offices. The vans are bright orange and carry the gear required to install fiber optic and Ethernet cabling in old urban neighborhoods, like Vesterbro, where the company is based. Headquarters is a practical and nondescript upstairs space. Ethernet cabling forming artful shapes is stapled to the walls. This is where Jakob handles business and monitors performance, waiting for trouble that rarely happens. When asked to describe the company's tech, he replies, "It's all Linux." On the same screen, he shows us a video of the company trenching a street on a Sunday (when vehicle traffic is minimal) to lay conduit for fiber optic cabling. Indienet does most of the grunt work themselves, including re-assembly of the old cobblestone street surface when the trenches are patched back up.

Down at street level, Jakob shows us Indienet's garage workshop, where wooden spools of green and orange fiber optic cabling sit on floors and shelves. Piled near the door are many smaller spools of CAT6 wiring. Wire channels, conduit, connecting hardware, drills, saws and other tools are neatly arranged in the space behind a collection of foosball tables in shipping boxes. Nearest the door are two pallets of Applied Telecom switches. "Those are the best", Jakob says—though he saves his highest admiration for Soekris, the Santa Cruz-based maker of small workhorse embedded Linux boxes. "They're simple and extremely reliable." (I just noticed, while surfing around the Soekris site, that the company's founder and CEO is Soren Kristensen. I'm guessing he's Danish. Coincidence?)

We walk down the street to a curbside where one of the Indienet vans is parked. Next to it Jakob's partner Preben Conrad is rolling a smoke before heading down into the basement of an apartment building, where he and another guy are wiring up the place. Preben is the lead technical guy for the company and its chief installer. In the basement, the guys show me the empty rack where the routers, switches and other gear will go, and the metal wiring channels they've been assembling and are now attaching to the ceiling. Next to the channels, several generations of telephone wiring and cable TV co-ax are tacked to floor joists and disappear through holes drilled in the floors above. The difference between installations is notable. Indienet's is built to service and to improve when the need comes. The building, typical for the neighborhood, is seven floors high and about 100 years old. The basements make it easy to run wiring to appropriate points below the stacks of flats above. And, the co-op nature of most apartments makes it easy to deal with one entity.

In most cases, the co-ops themselves are the customers. And, once the installation is complete, the relationship with Indienet is service-based. The company just makes sure the

Figure 1. Indienet headquarters—the art on the walls is made from Ethernet cabling.
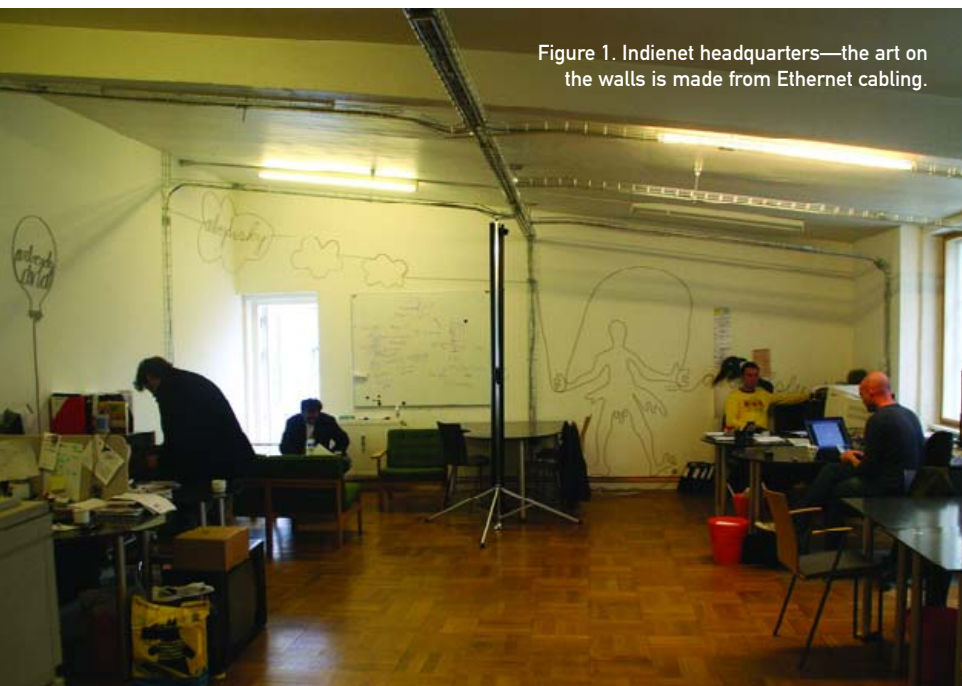
Internet is up and running. In some cases, that involves monitoring usage and doing traffic shaping. This is more likely to be necessary where the shared connection is a single ADSL line. ADSL is the familiar standard Internet offering in Denmark, though Indienet and similar grass-roots outfits would rather change that, especially as more fiber-based connections are deployed. (Fiber deployment is inherently wide open and symmetrical.)

Thomas, who set up the visit and is tagging along, explains how, as Indienet continues wiring and daisy-chaining apartment buildings in districts like Vesterbro, the company and its customers increase their independent buying power, which they bring to the country's backbone transit providers and ISPs.

In the midst of all this is a convergence of different approaches, deals, partnerships and deployment approaches. Thomas gives Bryggenet, Tirkontnet and Parknet as three examples of ideal grass-roots efforts. These typically include 20 to 30 buildings, or a total of 3–4,000 households. Parknet's deployment, he says, is currently a 200-megabit one. By contrast, Indienet, which owns the infrastructure it deploys, is a customer service provider to groups of people who would rather not have to deploy and keep up the infrastructure themselves. Both types of entities typically

Figure 2. Jakob Frederiksen of Indienet.dk, outside its offices in Copenhagen

hook into GlobalConnect, which is a country-wide fiber interconnect provider. GlobalConnect in turn hooks into a variety of wholesale transit providers. At this "back end", competition also increases, and costs gradually come down across a very complicated board.

In Thomas' own case, his co-op apartment did its own Ethernet deployment, hooked up with one of the transit

Figure 3. Indienet Van

Figure 4. Spools
of CAT6 Cabling

does. "The Net goes down maybe once a year", Thomas says. And, there are plenty of geeks among the residents anyway. Theirs is a totally DIY-IT approach.

Among wholesale transit providers are TDC, which for many decades was Denmark's national PTT (which stood for Postal Telegraph and Telephone, and generically still stands for the old state-owned communications companies). Later, Thomas and I spoke at length with TDC's Per Rasmussen, who made it clear just how complex and competitive the market for Internet deployment and service has become.

For example, in suburban and rural areas, the old cable television head ends are being put to new use by the communities themselves. It's easy to forget today that what we now just call "cable" began as CATV, or Community Antenna TV. In Denmark as well as in the US and elsewhere, cable got its start when communities—on their own or with the help of specialized contractors—put up towers and antennas at some central point (usually where the antennas could see the originating transmitters of TV signals), and amplified those signals down coaxial cabling to the houses of customers. Today, Per explains, these old cables (and newer ones too) are being repurposed for Internet service, with telephony and television riding on top of the Net as services. Meanwhile, a large number of private electric power companies, leveraging war chests of cash gained from selling their power plants, are getting into the phone/Internet/TV triple-play business, driving fiber down many last miles to many homes. With their smaller cash reserves, companies like TDC have to be more careful about how and what they deploy, along with whom and what they connect to.

What gets Thomas most excited is users owning their own infrastructure, and the opportunities for small grass-roots companies like Indienet to grow the Net "from the outside in"—from customers at the edges toward the backbones. All the market fragmentation and competition, he says, serves to force vertical integrators to unbundle their offerings. Hearing this made me envious. Back in the US, most residents have a choice between two completely integrated vertical silos—one from the phone company and one from the cable company. Neither of which seems terribly interested in deploying fiber, much less raw Internet infrastructure. Although TDC and Denmark's power companies might envy that kind of exclusivity and success, Thomas says those are bad models. "The cost of the Net itself is headed toward zero", he says. "Along the way, it's just a matter of taking on a small infrastructure cost of $300–2,000 dollars for once, instead of paying off a service provider's investment for 50 years with a high premium."

As I wrap this up, the Net came back up in the hotel, without all the blocked sites and ads (though SMTP and SSH are still blocked). The front desk tells me the problem was a "software mistake". Seems their service provider installed the wrong firewall. What we had was one intended for a school or a business, rather than for a hotel. "I wish it was simpler", the guy says.

If the hotel listened to some of the enterprising geeks in the neighborhood, it would be.∎

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.

companies, and distributes an ADSL service to residents. The cost is so low (around $3/month) that it falls under the standard co-op fees to households. Although there is no Indienet to call if something goes wrong, usually nothing

THE PENGUIN

Another server down. Another night at the office. Whether it's deploying a hundred new servers or installing the latest security patch, it doesn't matter. You're sleeping with the servers again. Penguin Computing® introduces Scyld Enterprise *WebMaster*™. Its centrally-managed, highly available architecture makes large pools of Linux servers act like a single, consistent virtual system, significantly reducing complexity and time to deploy servers on the fly. Its highly secure environment makes your server farm a 'virtual fortress' and its simplified manageability cuts your maintenance windows and administration by up to 80%, dramatically improving TCO. So go on home, catch some zzzzs and know that Penguin is standing guard.

MAY CAUSE DROWSINESS

Highly SCYLD

AMD 64 Opteron™

PENGUIN HIGH DENSITY CLUSTER. The highest density, modular blade server architecture on the market. With powerful Scyld Enterprise *WebMaster*™ for single point command and control, and AMD Dual Core Opteron™ for a highly productive user experience.

www.penguincomputing.com

PENGUIN COMPUTING

REALLIFE**LINUX**

Steven Goodwin's *Game Developer's Open Source Handbook* (Charles River Media)

Charles River Media's recent book titled *Sex in Video Games* came out too early for me to include it here (curses!), but then a new title with much appeal caught my eye: *The Game Developer's Open Source Handbook* by Steven Goodwin. The book is targeted at "all game developers, especially the 'Indies', who want to use the wealth of free software in their own games to help increase the scope of the technology available and reduce the financial burden". Charles River also calls it "required reading for the producers and systems analysts of game studios who want to see the big picture". The book's main purpose is to help the game developer find and utilize the plethora of open-source software tools and libraries—such as graphic editors, IDEs, MIDI sequencers, 3-D editors, movie playback code and so on—for use in every aspect of the development process. The author, Steven Goodwin, has been responsible for developing five different game titles, including *Die Hard: Vendetta* on the three big console platforms.

www.charlesriver.com

EMAC Inc.'s SoM-NE64M

The EMAC folks have let us know about their new 16-bit, System on Module Internet-appliance engine, which they have übercreatively named SoM-NE64M. The SoM-NE64M module is based on the Freescale ColdFire MC9S12NE64, 16-bit, 68HC12-compatible processor with built-in Ethernet MAC and PHY and two serial ports. It also features 64KB of Flash, 32KB of EEPROM and 8KB of RAM, with room for up to 512KB. The aforementioned functionality is integrated into a diminutive board—smaller than a business card and using less than a Watt of power—and is designed to plug in to a custom carrier board. Applications for the SoM-NE64 can be programmed using GNU tools within an Eclipse IDE or with CodeWarrior. One of the product's advantages, says EMAC, is "more functionality built in than many other SoM designs", making the carrier board easier to design and produce and thus lowering cost and time to market. Target applications are Web/network data acquisition and control.

www.emacinc.com

SafeNet's Sentinel Hardware Keys

SafeNet has introduced its Sentinel Hardware Keys to the world of Linux. The product is a rights management token with military-grade security that is intended to allow "software developers in the Linux community to protect 32-bit software applications from piracy and implement flexible licensing models", sayeth SafeNet. When attached to a computer or network, the keys monitor and enforce the licensing of products that have been protected using SafeNet's solution. The Java-based Sentinel Hardware Keys Software Development Kit is supported on Red Hat Enterprise Linux, Fedora Core and SUSE and includes "a device driver to access keys, a network server dæmon to manage licenses, a Web-browser-based monitoring tool to track licenses on site and a set of Business Layer APIs for high-level licensing implementation."

www.safenet-inc.com
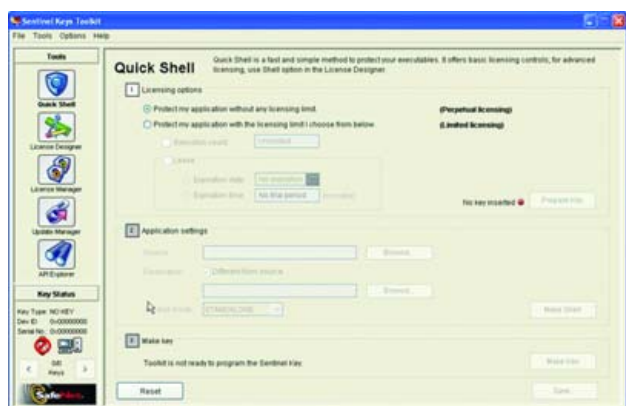
Steve Atwal's *Building Websites with XOOPS* (Packt Publishing)

Packt Publishing is a relatively new yet prolific IT publisher that focuses heavily on Linux and open-source titles. Its tagline reads "Community Experience Distilled", with the firm contributing a royalty back to the open-source projects it writes about. A case in point is Packt's new title, called *Building Websites with XOOPS: A step-by-step tutorial* by Steve Atwal. XOOPS is a popular open-source, object-oriented, PHP-based Web content management application. The book introduces readers to XOOPS and shows how to use it to create "small to large dynamic community Websites, intracompany portals, corporate portals, Weblogs and much more". Some topics covered include configuration of XOOPS, working with news stories and managing diverse elements, such as blocks, modules, users, themes and more.

www.packtpub.com
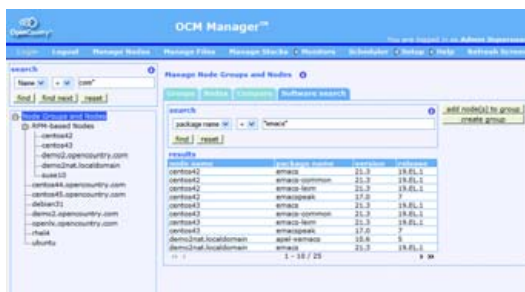
Open Country's OCM Universal Linux System Management Suite

Open Country hops on the 64-bit bandwagon with release of the OCM Universal Linux System Management Suite, Version 3.1. This systems management application now supports Intel's Itanium 2 processor line. OCM's raison d'être is to "help companies with widely distributed Linux investments to easily discover their entire inventory of hardware/software investments, then track installations and updates, deploy security patches, simplify repetitive management tasks, and respond effectively to changing computing needs". Open Country further credits its Web-based architecture with optimizing expertise and reducing labor costs over traditional client-server architectures. In addition, besides the mainline Linux distributions, OCM supports many distributions less common to North America, such as Asianux, CS2C, Red Flag, Turbolinux, Haansoft and several others.

www.opencountry.com

Interact-TV's ProTelly Home Entertainment Servers

Okay media packrats, this one's for you. Interact-TV has just released a line of home entertainment servers, called ProTelly, which will permit you to stash your DVDs and audio CDs in the basement for good. The products range from the the baseline ProTelly Media Server that can hold up to 150 DVDs to the ProRAID, which, with 3TB of protected storage, can hold up to 600 DVDs. All ProTelly products include features such as a subscription-free PVR, video library with a "save DVD" function, as well as music and photo libraries. In addition, it has features that Interact-TV says people in the home networking and home automation fields are looking for, namely component video out with 720p and 1080i, Gigabit Ethernet and MPEG-2 video encoding. Naturally, Linux is inside, making all of the enjoyment possible.

www.interact-tv.com

Arkeia Software's Arkeia Network Backup

Arkeia Software recently brought forth the release of Arkeia Network Backup Version 6, the firm's flagship data-protection solution for medium- to large-sized networks. Arkeia says that the main intent of Version 6 is to "improve backup performance and increase flexibility for distributed infrastructures such as organizations with Storage Area Networks". Some of the specific new features include a media server for SAN option that enables LAN-free backup for SAN environments, remote drive management for LANs and WANs to centralize the management of remote servers and networks and to consolidate and share drives across the LAN, an integrated virtual tape library option to leverage the performance and flexibility of disk technology, and a disk-to-disk-to-tape option to shorten backup/restore times and to create granular tiered storage policies. A trial version is available at Arkeia's Web site.

www.arkeia.com

The OpenVZ Project's OpenVZ

The OpenVZ Project recently announced that its OpenVZ OS-level server virtualization solution, which is built on Linux, is now available for systems using Power 64-bit processors. Like other virtualization solutions, OpenVZ allows one to create isolated, independent, secure virtual environments on a single physical server in order to achieve better server utilization and ensure that applications do not conflict. However, the OpenVZ Project asserts that its advantages lie not only in its single rather than its multiple kernels but also in its "portability across different architectures since 95% of the code is platform-independent". The OpenVZ Project is an Open Source community project supported by the firm SWsoft, which utilizes OpenVZ as the heart of its commercial virtualization product, dubbed Virtuozzo. The OpenVZ software, complete with Power support, can be downloaded from the project's Web site.

www.openvz.org

```
[kir@ovz-x64 ~]$ sudo /usr/sbin/vzlist
    VPSID      NPROC STATUS    IP_ADDR         HOSTNAME
    1001          15 running   10.1.1.1        vps1001
    1002          15 running   10.1.1.2        vps1002
    1003          15 running   10.1.1.3        vps1003
    1004          15 running   10.1.1.4        vps1004
    1005          15 running   10.1.1.5        vps1005
    1006          15 running   10.1.1.6        vps1006
    1007          15 running   10.1.1.7        vps1007
    1008          15 running   10.1.1.8        vps1008
    1009          15 running   10.1.1.9        vps1009
    1010          15 running   10.1.1.10       vps1010
    1011          15 running   10.1.1.11       vps1011
    1012          15 running   10.1.1.12       vps1012
    1013          15 running   10.1.1.13       vps1013
    1014          15 running   10.1.1.14       vps1014
    1015          15 running   10.1.1.15       vps1015
```

Have Laptop, Will Travel—the LS1250 Laptop from R Cubed Technologies

R Cubed Technologies makes a Linux dream machine out of an ASUS notebook. JAMES GRAY

Although you certainly can pick up a laptop from number of mainline PC makers and install Linux yourself, this remains a risky proposition. Whether it's fun or frustrating depends on the distro, the machine and, of course, your skills. The graphics adapters, chipsets, power-saving features and other elements make laptops inherently more complex than your standard desktop. Many of us look forward to the challenge of calling on our ingenuity and resources, such as the Linux on Laptops site (**www.linux-on-laptops.com**), to make the thing work. But what if you absolutely positively need it to work out of the box?

Your desire for more standard hardware might direct you to the mainline companies; however, there you'll be barking up the wrong tree. HP, for instance, once had a pre-installed Linux laptop. My conspiracy theory on why it disappeared? One of their VPs freaked when the 425 area code popped up on her caller ID; hence the kibosh. Regardless of the reason, your better bet is to call on one of the myriad scrappy, garage-and-basement-founded hardware companies that flourish in our community. If you look around, you'll find a wide array of options, with many of the machines produced by mainline companies but customized by Linux specialists.

A fine example of this innovative breed of Linux company is R Cubed Technologies, whose LS1250 laptop is the focus of this review. *Linux Journal* Editor in Chief, Nick Petreley, had had his eye on this sweet little machine for some time and asked me to review it, not knowing I had actually just bought one. Thus, I have had the machine for a few months and am in the perfect position to rate it after much day-in-day-out usage.

Exhibit A
My old laptop was a beast. I bought it as a desktop replacement with a nice, big display

for doing GIS. Unfortunately, I couldn't get a cheap copy of ArcGIS, so I do GIS at my university's computer lab instead. Then, I started traveling more, which left me lugging the beast around the world on my chronically sore shoulder. "Wouldn't it be nice to travel in comfort?", I thought.

Beyond portability, I wanted a laptop that would fit my mobile editor/student lifestyle. I was looking for solid performance at a fair price and dual-boot functionality, as well as excellent keyboard, display and Wi-Fi support. See the sidebar for information and specs on the LS1250.

As you can see from its specs (for example, the older processor), although the LS1250 is by no means cutting edge, it packs a solid punch into a small, easy-to-tote package. Note also that the LS1250 is actually built by Taiwan's ASUS Computer. R Cubed's role is to ship you the LS1250 packed with Linux goodies, as well as other OSes if you so desire. Thus, in order to give credit where due, let's take a closer look at both the LS1250's physical aspects (ASUS' responsibility) and the functional aspects (R Cubed's responsibility) and see how this machine stacks up.

Let's Get Physical
Being the geek that I am, I approach the practical with confidence and the style factor with apprehension. Though style is secondary to me, I admit that ASUS has made a sleek and attractive laptop. I like the LS1250's matte silver-grey color with black trim. The nagging doubts I had earlier about the "cool factor" have been gradually vanquished with each woman (now four and counting!) who raves about my cool laptop. Admittedly, a Mac or Sony VAIO will generate more net saliva, but the LS1250 may prove a better value on a "conversation starters per dollar" basis.

Not only is the LS1250 handsome, but also it feels well built. The carbon-fiber alloy materi-

- ▶ Vendor: R Cubed Technologies.
- ▶ URL: **www.shoprcubed.com**.
- ▶ Model: LS1250, based on ASUS Z33Ae platform (**usa.asus.com**).
- ▶ CPU: Pentium-M 760 (2.0GHz).
- ▶ Chipset: Intel 915GM.
- ▶ RAM (maximum): 768MB DDR2 integrated (1GB).
- ▶ OS options, as tested: SUSE 10.1, Windows XP Professional (dual-boot). Also available: Fedora Core, Ubuntu, Red Hat Enterprise Linux WS, Windows XP (Home, Media Center or 2003 Server).
- ▶ Display size/type: 12.1" XGA TFT LCD.
- ▶ Resolution: 1024 x 768.
- ▶ Video: integrated with 128MB shared memory.
- ▶ Hard disk: 80GB, 7200 RPM.
- ▶ CD/DVD: DVD-ROM/CD-RW (fixed).
- ▶ Ethernet: built-in 10/100Mbps LAN.
- ▶ WLAN: built-in wireless 802.11b/g.
- ▶ Bluetooth: yes.
- ▶ Modem: yes (56K), but not supported in Linux.
- ▶ USB 2.0: four.
- ▶ FireWire: one.
- ▶ PCMCIA Type II: one.
- ▶ Card reader: SD/MMC/MS.
- ▶ Monitor: VGA.
- ▶ Sound: earphones and microphone.
- ▶ Serial/Parallel/PS/2: none.
- ▶ Battery type: 3-cell Lithium-Ion (2.5 hours, 1.75 hours actual).
- ▶ Weight: 3.4 lbs./1.5 kg.
- ▶ Dimensions (LxWxH): 10.8 x 9.3 x 1.3 in./27.4 x 23.6 x 3.3 cm.
- ▶ Support/warranty: one year included with purchase.
- ▶ Price as tested: $1,654 US (including two-year extended warranty).

al gives the chassis a nice, solid feel—neither bulky, creaky nor "plasticky" but rather more rigid, almost metallic. Both ASUS and R Cubed claim that the carbon fiber improves portability and is "120% stronger than conventional material". Dropping the laptop to prove the latter point was fortunately not part of the test.

Regarding portability, this is an area where the LS1250 performs well. ASUS rightfully classifies the LS1250 (that is the Z33Ae in its catalog) as an ultraportable. Weighing in at a thrifty 3.4 lbs. (1.5 kg), the LS1250 slips easily into my backpack or laptop case with the same burden as a mid-sized, softcover book.

Once I've transported my LS1250 to its destination, I am generally pleased with its physical performance. The aspect I like most is the crisp, responsive keyboard. Despite this enjoyment, however, I find fault with the combination cursor block and scroll keys that were stuffed into the congested lower-right corner. I

continually reach over and press the wrong undersized key or play Twister with my fingers. Getting rid of one of those special (er, stupid) Windows keys would free up plenty of room for a better layout.

The 12.1" XGA TFT LCD display is bright, crisp and consistent with no dead pixels and works decently in direct sunlight.

As mentioned above, the ergonomic design of the LS1250 is strong. For example, ASUS won a German industrial-design award for the rimless design of its touch pad, which sits flush with the palm rest. This design feels smooth to the touch and eliminates dust accumulation.

I chose the standard 3-cell Li-Ion battery, which worked solidly but less than promised. While in a "word processing" power-management mode, I got around 1.75 rather than 2.5 hours worth of word processing and wired Web surfing.

Finally, heat management is solid, though

the underside and palm rest can get quite warm under everyday working conditions. The fan located on the right side runs continuously but acceptably quietly.

R Cubed Inside

Of course, I could have bought the LS1250 directly from ASUS and installed Linux myself. Instead, I chose to have R Cubed do my dirty work. It was a good decision, because R Cubed invests a great deal of effort to make nearly everything work smoothly out of the box. When I placed my order, R Cubed was offering only Fedora Core, but when I asked for SUSE 10.1, R Cubed obliged and sent me a fully functional machine with its own customized kernel. Now R Cubed says that it is an official Novell partner because of this, so maybe I should be asking for royalties?

I was very pleased with R Cubed's Linux installation for three main reasons. First, the LC1250 came preconfigured with a majority of the applications you'll find in a SUSE distro. Second, all of the function buttons worked appropriately on both Linux and Windows XP. And third, I was surfing wirelessly within seconds of starting the machine. Let's take a closer look at each of these.

Everything You Need

The folks at R Cubed shipped me the LS1250 partitioned to my specification, which was dual-boot SUSE Linux 10.1 (60GB) and Windows XP (20GB). The GRUB bootloader was already configured as well. After booting the Linux side, I fired up KDE and found a vast majority of applications that come with a SUSE distribution, all conveniently categorized in the menus and with key icons on the desktop and below on the Kicker (taskbar). Not only do I have all of the standard applications—OpenOffice.org, Acrobat Reader, Firefox and so forth—but nearly every application type has at least two options from which to choose. It has barely been necessary to install additional programs on the machine. My most pleasant surprise occurred when I plugged my Canon PowerShot digital camera in to the USB port, which instantly was recognized by digiKam. I was asked if I wanted to download the photos on the camera, which I did, and I was managing them without a hitch.

I also am enamored with the Wi-Fi capabilities out of the box. The installed KNetworkManager is smart, performs automatic link-ups for you and makes managing wireless networks a breeze. I discuss actual Wi-Fi performance below.

Through no fault of R Cubed, I was unable to play most video formats out of the box, despite having the various media players installed, such as Kaffeine, Totem and RealPlayer. This left me to find and install

7th Annual

2007 Web Services/SOA on Wall Street

Show and Conference

February 14, 2007, Wednesday

Roosevelt Hotel, NYC
Madison Ave and 45th St, next to Grand Central Station

The Largest New York Meeting for Web Services and Service-Oriented Architecture.

The 7th Annual Web Services on Wall Street will assemble Wall Street and global financial services IT decision-makers for a focused show and conference for SOA and Web Services. Java, XML, AJAX and other Web Services technologies will also be featured in this conference program and technology showcase.

SOA is hot: Wall Street IT managers, project managers and Web Services decision-makers want to see the latest tools and techniques to lower costs and reduce total cost of ownership. The show will provide face-to-face contact with exhibitors and 800 attendees at the convenient Roosevelt Hotel at Grand Central Station.

WEB SERVICES/SOA ON WALL STREET IS YOUR FIRM'S OPPORTUNITY TO MEET AND SELL.

Media Coverage of 2006 Web Services/SOA on Wall Street

eWeek - "Web Services, SOA, Open Source Converge"
eWeek - "SOA Fosters Ecosystem, Partnerships"
eWeek - "Tangosol Delivers New Data Grid System"
Security Industry News - "Goldman-Backed Developer Touts Java Breakthrough"
LinuxPlanet - "Wall Street: Linux Gets a Lift From Web Services, Tech Support"
Byte and Switch - "Wall Street Eyes SOAs"

2006 Gold sponsors
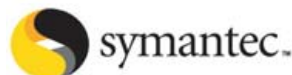
A Focused, No-Waste, Intimate Market:

Make your decision now. Past 2006 exhibitors will have priority in exhibit space assignments. This is your opportunity to get into the show now and benefit from the advanced promotion for the 2007 show. Past 2006 sponsors will have priority on their sponsorship programs to achieve high visibility, logo branding, and reinforce your presence on Wall Street. Please call us. We'll be happy to hold space for you subject to your approval. You can also email us at: www.webservicesonwallstreet.com or phone: 212-286-0333.

Web Services on Wall Street

For more information on the event, to be a sponsor or exhibitor, contact: Flagg Management Inc
353 Lexington Ave, New York, NY 10016
212-286-0333 Fax: 212-286-0086
flaggmgmt@msn.com

Visit: **www.webservicesonwallstreet.com**

codecs on my own. I learned that R Cubed is now providing multimedia installation icons that allow one to acquire instantly all of the codecs one needs to play back all common video formats.

Buttons That Work

On my old laptop, I never even thought about the function keys at the top of the keyboard, because none of them worked on Linux. However, all of them work on the LS1250 on both Linux and Windows—hibernate, wireless on/off, brightness control, display, browser launch, volume control and mute, and so on. R Cubed also provides users with a directory called .asus_acpi in their home Linux directory where users can customize what occurs when each button is pressed.

A button worthy of special mention puts the computer into five different modes of operation. These performance modes range from Turbo at the top to Word Processing in the middle to Maximum Battery Savings at the bottom. Each step down not only dims the display but reduces processor speed and hard drive spin while expanding the read and look-ahead caching in order to avoid powering up the hard drive.

Not Just Good Looks

Because this is a review of only one laptop, our performance assessments will be subjective. The performance matches my expectations considering the processor and memory (768MB RAM) onboard. I run all of the applications I want to, including audio CDs on Amarok, never feeling like the system is overtaxed or sluggish.

PROS:

► Excellent, thorough Linux configuration.

► Solid construction, stylish design.

► Light and portable yet usable.

► Strong performance, including Wi-Fi.

► Innovative touch pad.

► Accessible technical support.

CONS:

► Cramped cursor block.

► Poor battery performance (standard 3-cell).

► Limited phone-support hours.

The Wi-Fi performance exceeds my expectations. Besides the excellent network management mentioned above, the signal reception excels under even challenging conditions. My ultimate test is whether I can sit outside on my porch under a metal roof about 30 feet from my router. All of my previous laptops with wireless PC cards struggled to maintain a connection. The LS1250, with its integrated wireless, maintains a strong connection. Furthermore, from an unobstructed distance of about 50 feet the LS1250 was able to maintain wireless performance of about 13Mbps.

I tried experimenting with the 3-D acceleration by playing the game *Chromium B.S.U.*, but I had difficulty. Despite enabling acceleration with YaST, the game told me that it was unavailable, and it played sluggishly. R Cubed informed me that such a problem should not have occurred because the chipset supports acceleration. Unfortunately, I was unable to ameliorate this problem before deadline.

A Few Words about Service and Support

One of the most pleasant aspects of working with R Cubed is that the company is big enough to put out professional products yet small enough to know who you are. When I called to inquire about my order, I simply mentioned my name and the person at R Cubed knew what I ordered and its status off the top of his head. Furthermore, R Cubed has an order-tracking system that shows the order's status. My only complaint is that R Cubed does not enter any information in the system between the order placement and shipment. Thus, I was waiting for what I felt was a long time and was forced to call to discover the ETA of my machine.

Despite this complaint, the post-order support was as friendly, accessible and as personal as my earlier inquiry. The technician (probably the same person), who picked up after a few rings, knew my machine and troubleshot my problem (which was no audio on the Windows side) in just a few minutes. Phone-support hours are 8:00am to 5:00pm CT on weekdays only.

Recently I spoke with R Cubed's CTO, who told me that the firm is gradually expanding its support and features. One of these new offerings is remote support, whereby a technician can remotely access and troubleshoot a customer's machine via VNC. Another offering is a set of custom self-installations of applications, such as Internet Explorer on Wine, Google Earth and VMware Server. Finally, you can now ship your machine back to R Cubed and, for $50 US, the company will upgrade your OS with its custom kernel to maintain full functionality.■

James Gray is Products Editor for *Linux Journal*.

SMP
and
EMBEDDED REAL TIME

As embedded real-time applications start to run on SMP systems, kernel issues emerge.

Paul E. McKenney

With the advent of multithreaded/multicore CPUs, even embedded real-time applications are starting to run on SMP systems—for example, both the Xbox 360 and PS/3 are multithreaded, and there even have been SMP ARM processors! As this trend continues, there will be an increasing need for real-time response from SMP systems. Because not all embedded systems vendors will be willing or able to create or purchase SMP real-time operating systems, we can expect that a number of them will make use of Linux.

Because of this change, a number of real-time tenets have now become myths. This article exposes these myths and then discusses some of the challenges that Linux is surmounting in order to meet the needs of this new SMP-real-time-embedded world.

Real-Time Myths
New technologies often have a corrosive effect on the wisdom of the ages. The advent of commodity multicore and multithreaded hardware is no different, making myths of the following pearls of wisdom:

1. Embedded systems are always uniprocessor systems.

2. Parallel programming is mind crushingly difficult.

3. Real time must be either hard or soft.

4. Parallel real-time programming is impossibly difficult.

5. There is no connection between real-time and enterprise systems.

Each of these myths is exposed in the following sections, and Ingo Molnar's -rt real-time patchset (also known as the CONFIG_PREEMPT_RT patchset after the configuration variable used to enable real-time behavior) plays a key role in exposing the last two myths.
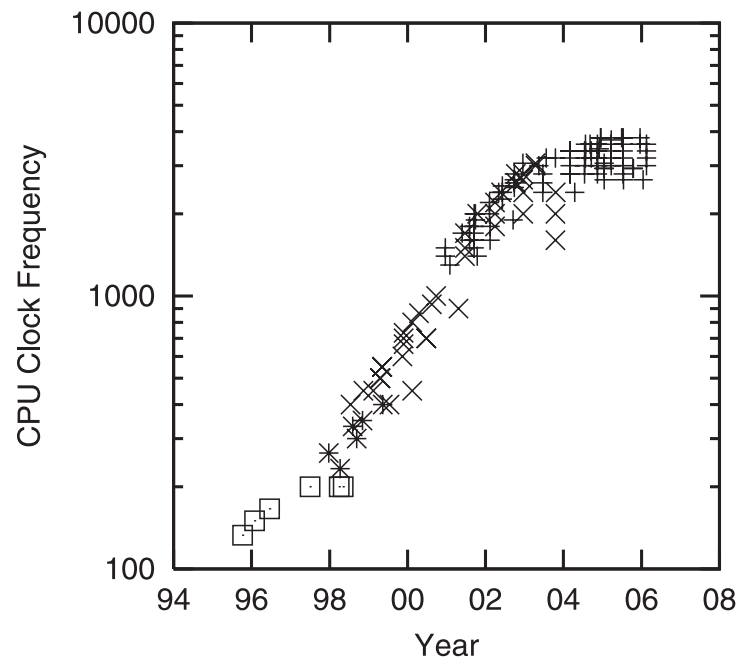
Figure 1. Clock-Frequency Trend for Intel CPUs

MYTH 1

Embedded Systems Are Always Uniprocessor Systems
Past embedded systems almost always were uniprocessors, especially given that single-chip multiprocessors are a very recent phenomenon.

The PS/3, the Xbox 360 and the SMP ARM are recent exceptions to this rule. But what does the future hold?

Figure 1 shows how clock frequencies have leveled off since 2003. Now, Moore's Law is still in full force, as transistor densities are still increasing. However, these increasing densities are no longer providing the side benefit of increased clock frequency that they once did.

Some say that parallel processing, hardware multithreading and multicore CPU chips will be needed to make good use of the ever-increasing numbers of transistors. Others say that embedded systems need increasing levels of integration and reduced power consumption more than they do ever-increasing performance. Embedded systems vendors might therefore choose more on-chip I/O or memory over increased parallelism.

This debate will not be resolved soon, although we have all seen examples of multithreaded and multicore CPUs in embedded systems. That said, as multithreaded/multicore systems become cheaper and more prevalent, we will see more rather than fewer of them.

But these multithreaded/multicore systems require parallel software. Given the forbidding reputation of parallel programming, how are we going to program these systems successfully?

MYTH 2

Parallel Programming Is Mind Crushingly Difficult

Why is parallel programming hard? Answers include deadlocks, race conditions and testing coverage, but the real answer is that it is not really all that hard. After all, if parallel programming was really so difficult, why are there so many parallel open-source projects, including Apache, MySQL and the Linux kernel?

A better question would be "Why is parallel programming perceived to be so difficult?" Let's go back to the year 1991. I was walking across the parking lot to Sequent's benchmarking center carrying six dual-80486 CPU boards, when I suddenly realized that I was carrying several times the price of my house. (Yes, I did walk more carefully. Why do you ask?) These horribly expensive systems were limited to a privileged few, who were the only ones with the opportunity to learn parallel programming.

In contrast, in 2006, I am typing on a dual-core x86 laptop that is orders of magnitude cheaper than even one of Sequent's CPU boards. Because almost everyone now can gain access to parallel hardware, almost everyone can learn to program it and also learn that although it can be nontrivial, it is really not all that hard.

Even so, many multithreaded/multicore embedded systems have real-time constraints. But what exactly is real time?

Figure 2. Hard Real Time: But I have a hammer.
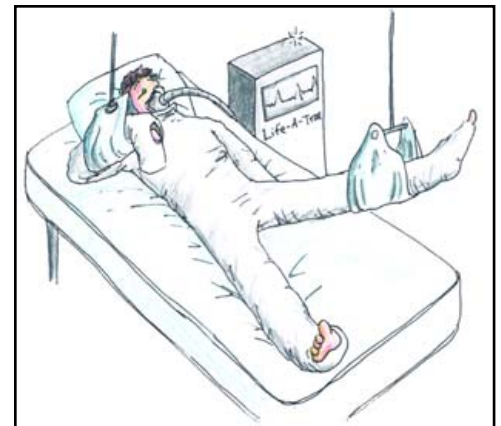
Figure 3. Hard real time: sometimes system failure is not an option!

Figure 4. Hard real time: at least I let you know!

MYTH 3

Real Time Must Be Either Hard or Soft

There is hard real time, which offers unconditional guarantees, and there is soft real time, which does not. What else do you need to know?

As it turns out, quite a bit. There are at least four different definitions of hard real time. Needless to say, it is important to understand which one your users have in mind.

In one definition of hard real time, the system always must meet its deadlines. However, if you show me a hard real-time system, I will show you the hammer that will cause it to miss its deadlines, as shown in Figure 2.

Of course, this is unfair. After all, we cannot blame software for hardware failures that it did not cause. Therefore, in another definition of hard real time, the system always must meet its deadlines, but only in absence of hardware failure. This divide-and-conquer approach can simplify things, but, as shown in Figure 3, it is not sufficient at the system level. Nonetheless, this definition can be useful given restrictions on the environment, including:

1. Interrupt rates.

2. Cache misses.

3. Memory-system overhead due to DMA.

4. Memory-error rate in ECC-protected systems.

5. Packet-loss rate in systems requiring networking.

If these restrictions are violated, the system is permitted to miss its deadlines. For example, if a hyperactive interrupt system delivered an interrupt after each instruction, the appropriate action might be to replace the broken hardware rather than to code around it. After all, if this degenerate situation must be accounted for, the latencies will likely become uselessly long. Alternatively, "diamond hard" real-time operating systems and applications might run with interrupts disabled, giving up compatibility with off-the-shelf software in order to gain additional robustness in face of hardware failure.

In yet another definition of hard real time, the system is allowed to miss its deadline, but only if it announces its failure within the deadline specified. This sort of definition can be useful in data-fusion applications. For example, a system might have a high-precision location sensor with unpredictable processing overhead as well as a rough-and-ready location sensor with deterministic processing overhead. A reasonable hard real-time strategy would be to give the high-precision sensor a fixed amount of time to get its job done, and if it fails to do so, abort its calculation, relying instead on the rough-and-

ready sensor. However, one (useless) way to meet the letter of this definition would be to announce failure unconditionally, as illustrated by Figure 4. Clearly, a useful system almost always would complete its work in time (and this observation applies to soft real-time systems as well).

Finally, some define hard real time with a test suite: a system passing the test is labeled hard real time. Purists might object, demanding instead a mathematical proof. However, given that proofs can be subject to error, especially for today's complex systems, a test suite can be an excellent additional proof point. I certainly do not wish to put my life at the mercy of untested software!

This is not to say that hard real time is undefined or useless. Instead, "hard real time" is the start of a conversation rather than a complete requirement. You should ask the following questions:

1. Which operations must provide hard real-time response? (For example, I have yet to run across a requirement for real-time filesystem unmounting.)

2. What is the deadline? A ten-millisecond deadline is one thing; a one-microsecond deadline is quite another.

3. What is to happen in case of hardware failure?

4. What is the required probability of meeting that deadline? (For hard real time, this will be 100%.)

5. What degradation of non-real-time performance, throughput and scalability can be tolerated?

One piece of good news is that real-time deadlines that once required extreme measures are now easily met with off-the-shelf hardware and open-source software, courtesy of Moore's Law.

But, what if your real-time application is to run on an embedded multicore/multithreaded system? How can you deal with both real-time deadlines and parallel programming?

MYTH 4

Parallel Real-Time Programming Is Impossibly Difficult

Parallel programming might not be mind crushingly hard, but it is certainly harder than single-threaded programming. Real-time programming is also hard. So, why would anyone be crazy enough to take on both at the same time?

It is true that real-time parallel programming poses special challenges, including interactions with lock-induced delays, interrupt handlers and priority inversion. However, Ingo Molnar's -rt

patchset provides both kernel and application developers with tools to deal with these challenges. These tools are described in the following sections.

Locking and Real-Time Latency

Much ink has been spilled on locking and real-time latency, but we will stick to the following simple points:

1. Reducing lock contention improves SMP scalability and reduces real-time latency.

2. When lock contention is low, there are a finite number of tasks, critical-section execution time is bounded, and locks act in a first-come-first-served manner to the highest-priority tasks, then lock wait times for those tasks will be bounded.

3. An SMP Linux kernel by its very nature requires very few modifications in order to support the aggressive preemption required by real time.

The first point should be obvious, because spinning on locks is bad for both scalability and latency. For the second point, consider a queue at a bank where each person spends a bounded time T with a solitary teller, there are a bounded number of other people N, and the queue is first-come-first-served. Because there can be at most N people ahead of you, and each can take at most time T, you will wait for at most time NT. Therefore, FIFO priority-based locking really can provide hard real-time latencies.

For the third point, see Figure 5. The left-hand side of the diagram shows three functions A(), B() and C() executing on a pair of CPUs. If functions A() and B() must exclude function C(), some sort of locking scheme must be used. However, that same locking provides the protection needed by the -rt patchset's preemption, as shown on the right-hand side of this diagram. If function B() is preempted, function C() blocks as soon as it tries to acquire the lock, which permits B() to run. After B() completes, C() may acquire the lock and resume running.

This approach requires that kernel spinlocks block, and this change is fundamental to the -rt patchset. In addition, per-CPU variables must be protected more rigorously. Interestingly enough, the -rt patchset also located a number of SMP bugs that had gone undetected.

However, in the standard Linux kernel, interrupt handlers cannot block. But interrupt handlers must acquire locks, which can block in -rt. What can be done?

Interrupt Handlers

Not only are blocking locks a problem for interrupt handlers, but they also can seriously degrade real-time latency, as shown in Figure 6.

This degradation can be avoided by running the interrupt handler in process context, as shown in Figure 7, which also allows them to
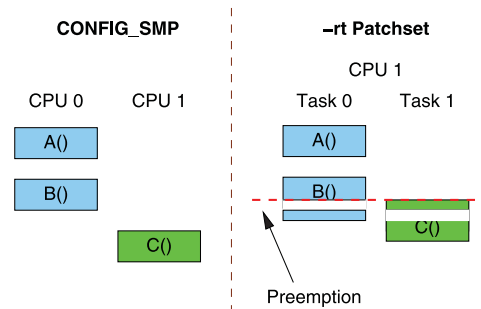
Figure 5. SMP Locking and Preemption
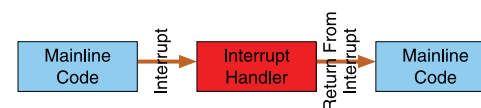
Figure 6. Interrupts Degrade Latency
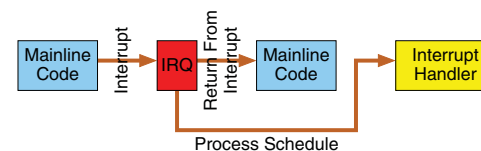
Figure 7. Move Interrupt Handlers to Process Context
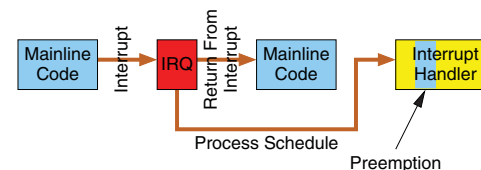
Figure 8. Preempting Interrupt Handlers

acquire blocking locks.

Even better, these process-based interrupt handlers can actually be preempted by user-level real-time threads, as shown in Figure 8, where the blue rectangle within the interrupt handler represents a high-priority real-time user process preempting the interrupt handler.

Of course, "with great power comes great responsibility." For example, a high-priority real-time user process could starve interrupts entirely, shutting down all I/O. One way to handle this situation is to provide a low-priority "canary" process. If the "canary" is blocked for longer than a predetermined time, one might kill the offending thread.

Running interrupts in process context permits interrupt handlers to acquire blocking locks, which in turn allows critical sections to be preempted, which permits extremely fast real-time scheduling latencies. In addition, the -rt patchset permits real-time application developers to select the real-time priority at which interrupt handlers run. By running only the most critical portions of the real-time application at higher priority than the interrupt handlers, the developers can minimize the amount of code for which "great respon-

sibility" must be shouldered.

However, preempting critical sections can lead to priority inversion, as described in the next section.

Priority Inversion

Priority inversion is illustrated by Figure 9. A low-priority process P2 holds a lock, but is pre-empted by medium-priority processes. When high-priority process P1 tries to acquire the lock, it must wait, because P2 holds it. But P2 cannot release it until it runs again, which will not happen while the medium-priority processes are running. So, in effect, the medium-priority processes have blocked a high-priority process: in short, priority inversion.

One way to prevent priority inversion is to disable preemption during critical sections, as is done in CONFIG_PREEMPT builds of the Linux kernel. However, this preemption disabling can result in excessive latencies.

The -rt patchset therefore uses priority inheritance instead, so that P1 donates its priority to P2, but only for as long as P2
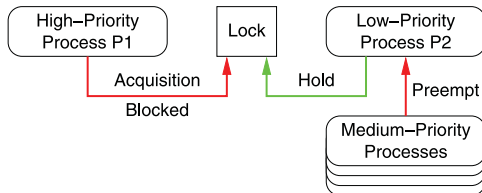
Figure 9. Priority Inversion
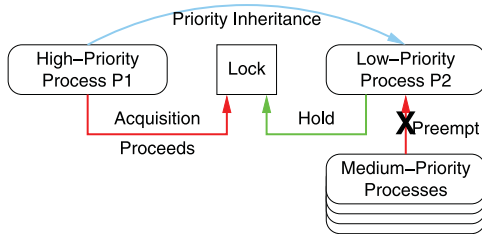
Figure 10. Priority Inheritance
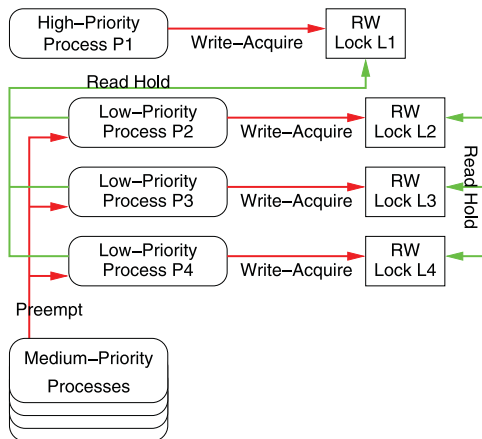
Figure 11. Reader-Writer Lock Priority Inversion

continues to hold the lock, as shown in Figure 10. Because P2 is now running at high priority, it preempts the medium-priority processes, completing its critical section quickly and then handing the lock off to P1.

So priority inheritance works well for exclusive locks, where only one thread can hold the lock at a given time. But there are also reader-writer locks, which can be held by one writer on the one hand or by an unlimited number of readers on the other. The fact that a reader-writer lock can be held by an unlimited number of readers can be a real problem for priority inheritance, as illustrated in Figure 11. Here, several low-priority processes are read-holding lock L1, but are preempted by medium-priority processes. Each low-priority process might also be blocked write-acquiring other locks, which might be read-held by even more low-priority processes, all of which are also preempted by the medium-priority processes.

Priority inheritance can solve this problem, but the cure is worse than the disease. For example, the arbitrarily bushy tree of preempted processes requires complex and slow bookkeeping. But even worse, before the high-priority writer can proceed, all of the low-priority processes must complete their critical sections, which will result in arbitrarily long delays.

Such delays are not what we want in a real-time system. This situation resulted in numerous "spirited" discussions on the Linux-kernel mailing list, which Ingo Molnar closed down with the following proposal:

1. Only one task at a time may read-acquire a given reader-writer lock.

2. If #1 results in performance or scalability problems, the problematic lock will be replaced with RCU (read-copy update).

RCU can be thought of as a reader-writer lock where readers never block; in fact, readers execute a deterministic number of instructions. Writers have much higher overhead, as they must retain old versions of the data structure that readers might still be referencing. RCU provides special primitives to allow writers to determine when all readers have completed, so that the writer can determine when it is safe to free old versions. RCU works best for read-mostly data structures, or for data structures with hard real-time readers. (More detail may be found at **en.wikipedia.org/wiki/RCU**, and even more detail may be found at **www.rdrop.com/users/paulmck/RCU**. Although user-level RCU implementations have been produced for experimental purposes, for example, **www.cs.toronto.edu/~tomhart/perflab/ipdps06.tgz**, production-quality RCU implementations are currently found only in kernels. Fixing this is on my to-do list.)

A key property of RCU is that writers never block readers, and, conversely, readers do not block writers from modifying a data structure.
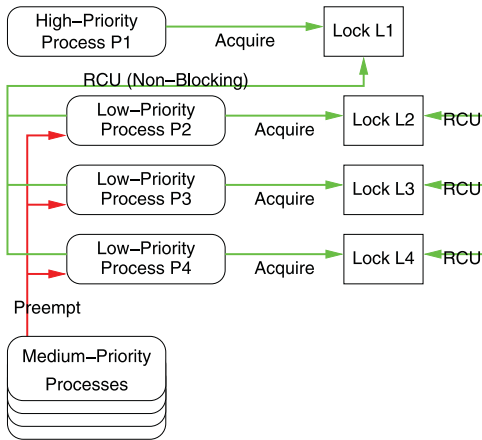
Figure 12. RCU Prevents Priority Inversion

Figure 13. Priority Boosting for Users?

Therefore, RCU cannot cause priority inversion. This is illustrated in Figure 12. Here, the low-priority processes are in RCU read-side critical sections and are preempted by medium-priority processes, but because the locks are used only to coordinate updates, the high-priority process P1 immediately can acquire the lock and carry out the update by creating a new version. Freeing the old version does have to wait for readers to complete, but this freeing can be deferred to avoid degrading real-time latencies.

This combination of priority inheritance and RCU permits the -rt patchset to provide real-time latencies on mid-range multiprocessors. But priority inheritance is not a panacea. For example, one could imagine applying some form of priority inheritance to real-live users who might be blocking high-priority processes, as shown in Figure 13. However, I would rather we did not.

Parallel Real-Time Programming Summary

I hope I have convinced you that the -rt patchset greatly advances Linux's parallel real-time capabilities, and that Linux is quickly becoming capable of supporting the parallel real-time applications that are appearing in embedded environments. Parallel real-time programming is decidedly nontrivial. In fact, many exciting challenges lie ahead in this field, but it is far from impossible.

But there are a number of real-time operating systems, and a few even provide some SMP support. What is special about real-time Linux?

MYTH 5

There Is No Connection between Real-Time and Enterprise Systems

To test the fifth and final myth, and to show just what is special about real-time Linux, let's first outline the -rt patchset's place in the real-time pantheon.

The -rt patchset turns Linux into an extremely capable real-time system. Is Linux suited to all

purposes? The answer is clearly no, as can be seen from Figure 14. With the -rt patchset, Linux can achieve scheduling latencies down to a few tens of microseconds—an impressive feat, to be sure, but some applications need even more. Systems with very tight hand-coded assembly-language loops might achieve sub-microsecond response times, at which point memory and I/O-system latencies loom large. Below this point comes the realm of special-purpose digital hardware, and below that the realm of analog microwave and photonics devices.

However, Linux's emerging real-time capabilities are sufficient for the vast majority of real-time applications. Furthermore, Linux brings other strengths to the real-time table, including full POSIX semantics, a complete set of both open-source and proprietary applications, a high degree of configurability, and a vibrant and productive community.

In addition, real-time Linux forges a bond between the real-time and enterprise communities. This bond will become tighter as enterprise applications face increasing real-time requirements. These requirements are already upon us—for example, Web retailers find that they lose customers when response times extend beyond a few seconds. A few seconds might
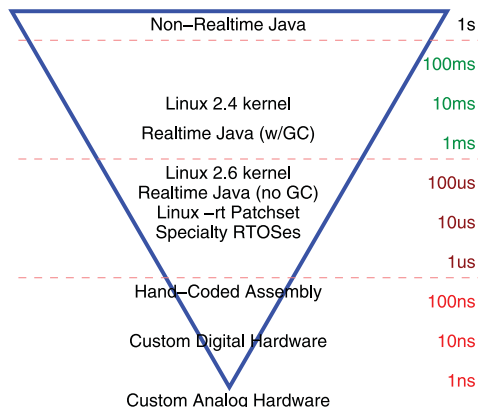
seem like a long time, but not when you 1) subtract off typical Internet round-trip times and 2) divide by an increasingly large numbers of layers, including firewalls, IP load levelers, Web servers, Web-application servers, XML accelerators and database servers—across multiple organizations. The required per-machine response times fall firmly into real-time territory.

Web 2.0 mashups will only increase the pressure on per-machine latencies, because such mashups must gather information from multiple Web sites, so that the slowest site controls the overall response time. This pressure will be most severe in cases when information gathered from one site is used to query other sites, thus serializing the latencies.

We are witnessing nothing less than the birth of a new kind of real time: enterprise real time. What exactly is enterprise real time? Enterprise real time is defined by developer and user requirements, as might be obtained from the real-time questions listed in the discussion of Myth 3. Some of these requirements would specify latencies and guarantees (hard or soft) for various operations, while others would surround the ecosystem, where real-time Linux's rich array of capabilities, environments, applications and supported hardware really shine.

Of course, even the rich real-time-Linux ecosystem cannot completely remove the need for special-purpose hardware and software. However, the birth of enterprise real time will provide a new-found ability to share software between embedded and enterprise systems. Such sharing will greatly enrich both environments.

Future Prospects

Impressive as it is, real-time Linux with the -rt patchset focuses primarily on user-process scheduling and interprocess communication. Perhaps the future holds real-time protocol stacks or filesystems, and perhaps also greater non-real-time performance and scalability while still maintaining real-time response, allowing electrical power to be conserved by consolidating real-time and non-real-time

Figure 14. Real-Time Capability Triangle

workloads onto a single system.

However, real-time applications and environments are just starting to appear on Linux, both from proprietary vendors and F/OSS communities. For example, existing real-time Java environments require that real-time programs avoid the garbage collector, making it impossible to use Java's standard runtime libraries. IBM recently announced a Java JVM that meets real-time deadlines even when the garbage collector is running, allowing real-time code to use standard libraries. This JVM is expected to ease coding of real-time systems greatly and to ease conversion of older real-time applications using special-purpose languages, such as ADA.

In addition, there are real-time audio systems, SIP servers and object brokers, but much work remains to provide a full set of real-time Web servers, Web application servers, database kernels and so on. Real-time applications and environments are still few and far between.

I very much look forward to participating in—and making use of—the increasing SMP-real-time capability supported by everyday computing devices!

Acknowledgements

No article mentioning the -rt patchset would be complete without a note of thanks to Ingo Molnar, Thomas Gleixner, Sven Deitrich, K. R. Foley, Gene Heskett, Bill Huey, Esben Neilsen, Nick Piggin, Steven Rostedt, Michal Schmidt, Daniel Walker and Karsten Wiese. I also owe a debt of gratitude to Ted Ts'o, Darren Hart, Dinakar Guniguntala, John Stultz, Vernon Mauery, Jennifer Monk, Sripathi Kodi, Tim Chavez, Vivek Pallantla and Hugh Miller for many valuable real-time-Linux words and deeds. I am likewise grateful to David Bacon and his real-time-GC-research team and to Boas Betzler for many productive conversations. We all owe Bruce Jones, John Kacur and Mark Brown many thanks for their invaluable service rendering this article human-readable. Finally, many thanks go to Daniel Frye for his unstinting support of this effort.

Legal Statement

This work represents the view of the author and does not necessarily represent the view of IBM.

Intel is a registered trademark of Intel Corporation or its subsidiaries in the United States and other countries.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc., in the United States, other countries or both.

Linux is a registered trademark of Linus Torvalds in the United States, other countries or both.

Other company, product or service names may be trademarks or service marks of others.∎

Paul E. McKenney is a Distinguished Engineer with IBM's Linux Technology Center. He has worked on NUMA, SMP and real-time algorithms and, in particular, RCU for longer than he cares to admit. In his spare time, he jogs and supports the usual house-wife-and-kids habit.

Choosing a GUI Library for Your Embedded Device

EVALUATING TWO VERY DIFFERENT GUI LIBRARIES, QTOPIA AND NANO-X.

MARTIN HANSEN
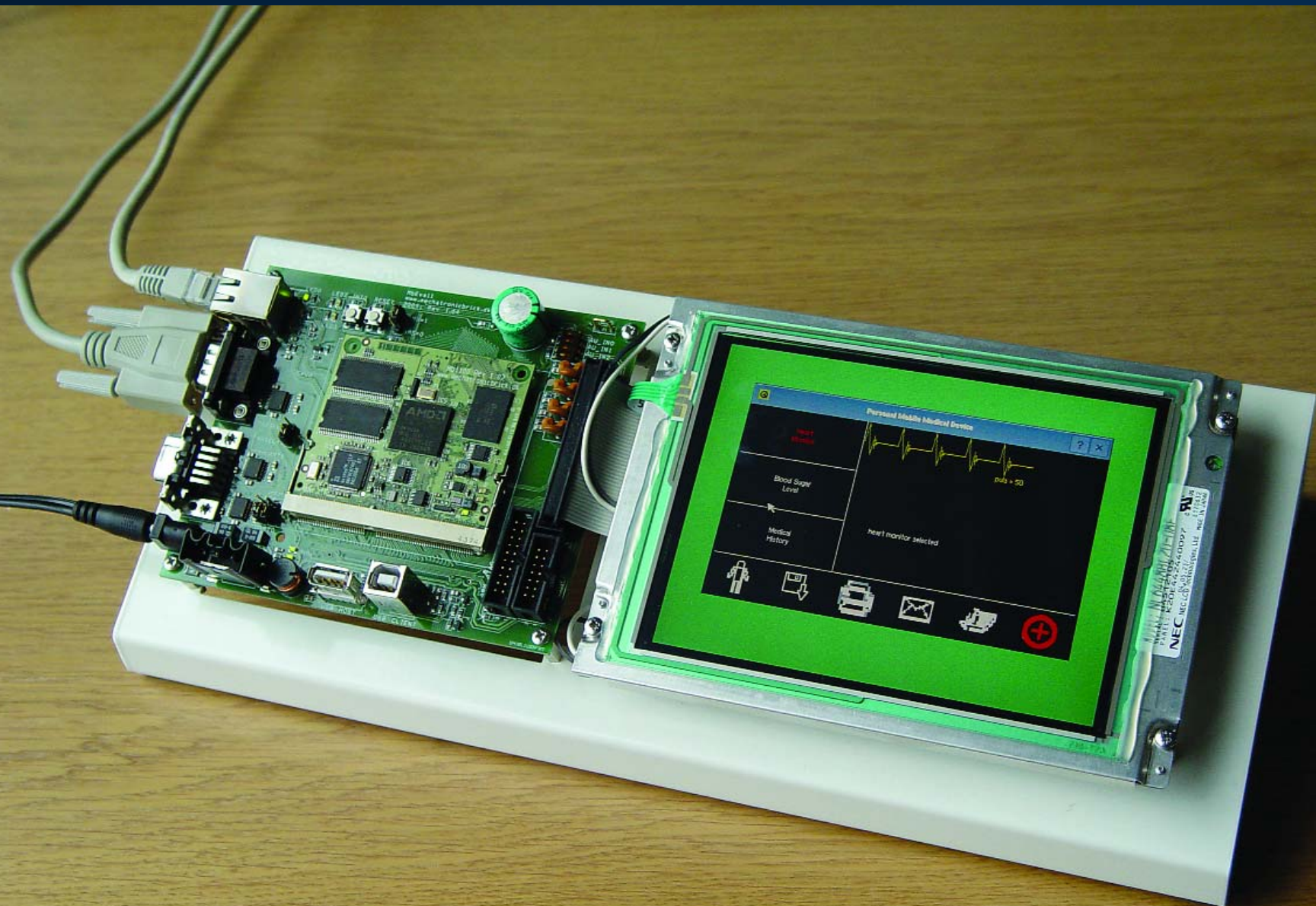
In this article, we look at two GUI libraries, examine the differences and give some advice on when to choose each.

The company I work for is dedicated to helping customers make the right decision about the technology they want to use in their embedded software development, and afterward it supports them in using the chosen technology. My specialty is embedded Linux.

Talking with customers, I see that more and more products need some sort of graphical display. So, I decided it was time to gain more knowledge about GUI development on embedded Linux.

The path I chose was the practical one. I did some research and found that the most common libraries are Qtopia (also known as Qt embedded) and Nano-X (formerly known as Microwindows).

How to Test

The first solution is simply to implement some test app to demonstrate the two GUI libs. Such "test apps" only seldom resemble a real-world application, but I do this mostly because I am an engineer, and engineers are more interested in the technology beneath than in the appearance.

"Then why have an engineer test the libraries?", you might ask. Well, think of GUI libraries as the technology to make an appearance. Therefore, you need both the technology view and the appearance view.

Another aspect of me doing this test is that I am not involved in any of the projects, and therefore I come with the knowledge that most other programmers have when they start out using the libraries. Some of my choices regarding the implementation probably are not optimal. They are made from the information available to the common user of the library—such as the problem with the scrolling graph, discussed below.

So, before ranting at me about how I could have done things differently, please look in the docs. Are they clear about the matter? If not, maybe it is better to change the docs instead.

I decided to get some external inspiration and went to the nearby university, where they have a department educating in User Centered Design (UCD). I asked one of the students, Esti Utami Handarini Povlsen (who was an old friend of mine), to come up with a GUI specification that I would then implement, using the two libraries. After having calibrated our technical language so we could communicate, we found a suitable design that I took to my home to implement.

The design that I got was for a Personal Mobile Medical Device (PMMD). The design consists of a single window with some static buttons and a changing display area showing text and/or graphs.

It turns out that the most challenging part is the heartbeat monitor graph, which is a varying line scrolling across the screen.

The Target Device

The platform I used for the evaluation is the MIPS32-based mb1100 from Mechatronic Brick. The mb1100 development kit is equipped with an AMD Alchemy au1100 CPU, a 6.5" TFT screen, an ADS7846E four-wire touch screen, 32MB of RAM and 32MB of Flash.

Qtopia

I started out with the Qtopia library. The creator of Qtopia is the Norwegian company Trolltech. Trolltech is mostly known for its Qt library on desktop computers; Qt is the base GUI of KDE. Qtopia is the embedded version of the Qt library.

Both Qt and Qtopia are dual-licensed, under the GPL and a commercial license. You can download the GPL version from the Trolltech Web site and use it as any other GPL library. This forces your Qt/Qtopia applications to go under the GPL too. You also can choose to buy the commercial license, which allows you to make closed source applications. The differences between the two versions are minor, if any, except of course for the licenses.

Getting It Up and Running

Qtopia can run directly on the framebuffer device, so make sure that the kernel is compiled with framebuffer support and that it is working.

That is the easy part. The difficult part is making the touch screen work. After having corrected a few glitches in the driver, I had a lot of

trouble calibrating the device in Qtopia.

I am using Qtopia with tslib for the touch screen, and after having corrected the driver, tslib was working, and the little calibration program included in the tslib package calibrated well. Drawing lines with the pen in the same program worked fine. After starting a Qtopia program, the calibration was gone, and I tried the calibration program from the Qtopia package with no luck.

I found the error when looking in the sources of Qtopia and tslib. When tslib starts up, it looks into a file in /etc. This file tells tslib what modules to load, and those modules usually include the linearization module and different noise filters.

The linear module is the one that does the calibration. When looking in the sources of Qtopia, I found that the programmer wanted to make sure that the linear module was loaded, so after parsing the tslib config file, Qtopia loads the linear module, regardless of what is written in the config file. This means that if the linear module is defined in the config file, it is loaded twice, and this breaks the functionality. Having figured this out, I removed the linear module from the config file. (I know the correct solution would be to correct the Qtopia sources, but I took the shortcut.) Now the calibration worked in Qtopia.

Programming in Qtopia

I will not go into detail about the implementation of my application, as it is not within the scope of this article. However, to summarize, Qtopia is C++-based, and I think the Qtopia designers have a good grasp of the idea of C++. As is no surprise, all widgets are objects, and to have standard functions (methods) in your own widgets (defined in your own class), you inherit from base classes.

The different objects (widgets) need to communicate. For example, if I click on a button, the button object might want to tell the text field object to update the text. In Qt, and thereby Qtopia, this is done using signals and slots. They are simply standard methods with an attribute. This interface makes it possible to make the objects independent of each other. The button just sends a signal, "clicked", the text object has a slot "update", and they compile and work fine without each other. Then, when I put them together in my app and give the connect (obj1, clicked, obj2, update) command in the initialization to connect signal clicked with slot update, the magic happens. The text is updated when I click the button.

Those connections even can be made automatically, simply by giving them the right name. If I have a widget named cancelBtn, with the signal clicked, and I make a slot named on_cancelBtn_clicked, the clicked signal from the cancelBtn is automagically connected to this slot. This signal/slot design makes the code easy to read and maintain. On the other hand, if you are not familiar with signals and slots, and you look at someone else's code, you can go on a wild goose chase looking for the calling of the slot (method) for a long time.

Documentation

So far, the documentation has been a great help. They have done a great job writing the documentation of the API. However, the API documentation does not help you if you don't know what API call you should use for a task. I spent a lot of time making the drawing object work correctly,
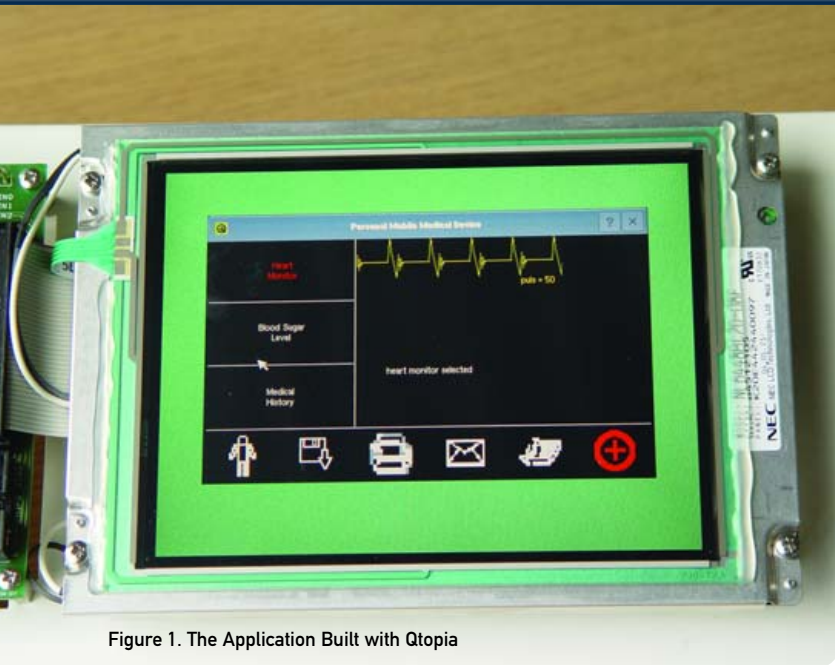
Figure 1. The Application Built with Qtopia

```
for (;;) {
  GrGetNextEvent(&event);
  switch (event.type) {
    case GR_EVENT_TYPE_EXPOSURE:
      GrText(w, gc, 10, 30, text, -1, GR_TFASCII);
      break;
    case GR_EVENT_TYPE_BUTTON_DOWN:
      text="hej verden";
      GrText(w, gc, 10, 30, text, -1, GR_TFASCII);
      break;
    case GR_EVENT_TYPE_CLOSE_REQ:
      GrClose();
      exit(0);
  }
}
```

because I had to collect the information from different places in the documentation. I never found an efficient way to make my scrolling graph. I did not find any bitmap manipulation that would scroll my heartbeat graph, so I chose to repaint the whole thing for every scroll step. There might be an easier way, but I did not find it.

Therefore, if you want to do more advanced programming in Qtopia, you need to find a good book or guide to complement the API documentation.

Nano-X

Nano-X was formerly known as Microwindows. Why the change of name? Take a wild guess. If your guess includes a lawyer, you are probably on the right track.

Nano-X is an open-source project at Nano-X.org, started and still headed by Greg Haerr. Nano-X is licensed under the MPL license. The MPL license allows you to create closed source drivers and applications. But, the Nano-X source itself must stay open. There is an option to use the GPL license, if desired.

Getting It Up and Running

The Linux package from Mechatronic Brick includes the Nano-X library, but this version did not include support for PNG pictures. I needed PNG support, so I had to recompile. This was quite easy after I found out what config file is used when building in the Mechatronic Brick setup. I noticed that Nano-X comes with a config file that set up Nano-X to be built with TCC, a small and very fast C-only compiler. I decided to use this too, and then the library was compiling in no time.

Programming in Nano-X

Starting to program in Nano-X is quite a change, especially when coming from the nice and polished C++ classes of Qtopia. Nano-X is so much simpler, which leaves a lot more work for the application programmer.

Nano-X does not have widgets, or buttons or combo boxes—only windows. There are libraries to put on top of Nano-X that will give you more features, such as Nano-X's own reimplementation of the win32 library and the Fast Light Toolkit (FLTK). In this article, we delve into the basic part of Nano-X.

Basically, when programming for Nano-X, you do four things:

1. Create windows.

2. Paint in the windows.

3. Select event types for each window.

4. Wait for an event (the event loop).

A typical standard application window is made of a base window with the frames and the small x close button (of course, there are options to customize this look). Subwindows act as buttons and display fields. Yes, in Nano-X, a button is declared like a subwindow with the mouse-click event selected.

In Qtopia, I simply made a class, connected some signals and slots, and puff, the magic happened. In Nano-X, I had to take care of things myself. A central part of a Nano-X application is the event loop, typically a for (ever) loop containing the get event function and a case structure to handle the event (see Listing 1 for an example). When I get a mouse-click event, I ask which window captured this event and act from that. This means that the single button is not isolated in its own piece of code, but weaves into the app. The basic function of the button or the display field should of course be in a function by itself, but the event loop must be aware of which events are selected in the button and what to do with the events.

Documentation

The documentation for Nano-X is a bit lacking. There are some great documents out there; however, the links from the Web page are not
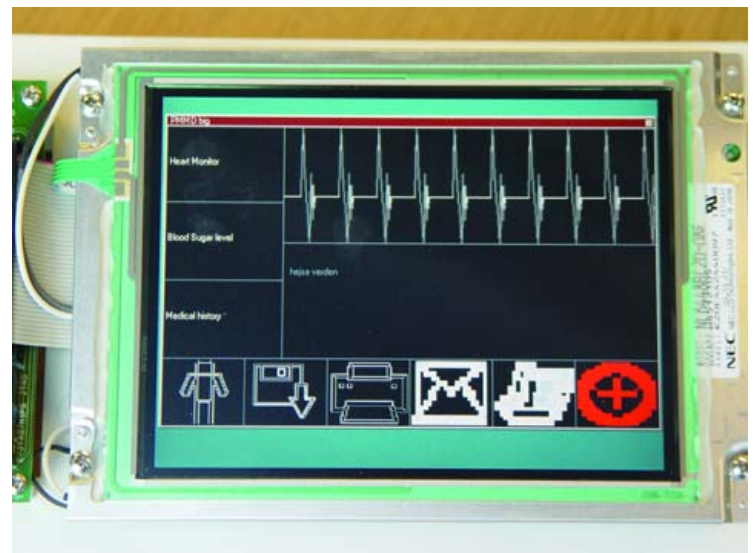
Figure 2. The Application Built with Nano-X

Table 1. Summary Table

	PMMD-QT	PMMD-NX
GUI	Qtopia from Trolltech (GPL version)	Nano-X
Programming Language	C++	C
Time spent learning to use the library	Approx. one week (three days for the installation and two days to learn the API)	Approx. one week (three days for the installation and two days to learn the API)
Development time for GUI and heartbeat monitor graph	Approx. two to four days	Approx. five to seven days
Code size of library	Compressed: 1.1–3.2MB	<100K
Documentation	API: really good; installation: needs work	API: usable; installation: needs work
License	GPL license and commercial license. The GPL version is free to download; the commercial version must be purchased.	MPL license with possibility for closed source drivers and applications. Nano-X is free to download.

updated, and many of them are dead. I used Google to find the most useful documentation. One also can use the Nano-X source and the mailing lists. The mailing list is very active, and Greg Haerr is right there, giving quick responses to questions.

A make doc in the sources will make some doc on the API using Doxygen, but not all functions are documented. I had to look directly in the source a few times.

Conclusion

Nano-X does win by miles when it comes to size. However, Qtopia is far ahead when it comes to polished graphics and nice, well-structured programming. Don't get me wrong, this is not entirely a C vs. C++ issue. You can do nice programming using C and Nano-X, but it does require more skill and discipline from the programmer. Hard-core C programmers will often crank out muddy C++ code with Qtopia, so C++ doesn't always translate into good practices—it all depends on your existing skills, time and willingness to learn.

Regarding speed, I did not see much difference, except in my scrolling graph. Using Qtopia, the graph was jittery, because I did not find a way actually to scroll the bitmap, so I had to redraw the complete graph for each step. The graph turned out nicely in Nano-X, using a bitmap copy to make the scrolling, and then just drawing the new part of the graph. Given more time and trial and error, it is likely that you could scroll more efficiently in Qtopia too—probably by sub-classing the right object. But given the current documentation, I did not find a way to do it.

Table 1 is a summary table for the two versions of the PMMD that I made, PMMD-QT and PMMD-NX. Installation includes compiling of the libraries. Code size is taken from the documentation.∎

Resources for this article: www.linuxjournal.com/ article/9460.

Martin Hansen works at the Danish company Center for Software Innovation (CSI, www.cfsi.dk). CSI provides knowledge in embedded development to companies, both through advisory and by giving "Technology Injections". Martin is the company expert on embedded Linux. He has been using Linux for more than ten years and has worked with embedded Linux for the last two years. He has a practical education in electronics and a Bachelor's degree in computer science.

The HAL Project

Discovering local artists through Zeroconf.

PASCAL CHAREST, MICHAEL LENCZNER AND GUILLAUME MARCEAU

Figure 1. The HAL Team: Pascal Charest, Francis Daigneault, Michael Lenczner and Richard Lussier (missing François Proulx)

Ah, the joys of hacking Linux on inexpensive commodity hardware. We are the Montréal community wireless group Ile Sans Fil, which was covered in this magazine in October 2005. During the last three years, we have deployed embedded systems that run Linux in public spaces across our city in an effort to encourage local communities. Our all-volunteer group now has more than 100 hotspots located in cafés, libraries and parks around the city, and more than 26,000 users. To accomplish this, we used the Linksys WRT54G, a favorite of hackers, and developed the captive portal suite WifiDog.

Our latest project is HAL, the Local Artist Hub (the acronym works in French). HAL boxes are small NSLU network storage devices that we install locally at certain of our Wi-Fi hotspots and then remotely fill with music and movies by local creators. Because the box is directly on the local area network, the content can be streamed at HDTV resolution without stalls or buffering and without bandwidth charges. Plus, because we use Zeroconf, the user's media player discovers the content automatically. Besides promoting serendipitous discovery, the user gets to interact with the content using a familiar interface that is specifically designed for rich media. We hope to make HAL servers a cultural meeting spot—an easy way for passers-by to engage with works by artists from that community.

The technologies we have plugged together also can be used in many ways, either as single installations or deployed in networks across multiple sites. In this article, we describe our setup so that you can get started on your own projects.

What about the Hardware?

HAL uses the NSLU2 network device from Linksys. It's a small board with a 266MHz XScale CPU (ARM architecture, by Intel), two USB 2.0 ports and one 10/100Mbps network interface. The NSLU2 is another favorite among hackers. There are two alternative firmwares available for it, Unslung and and OpenSlug, both of which are supported by an active community. We've chosen OpenSlug for this project.

As we cannot vouch for the electrical system at the venue, we physically wire the boards with an auto-on circuit. If you want instructions on how to do that, you should visit the Web site and read through the appropriate disclaimers about voiding your warranty and burning down your house.

Because the NSLU doesn't have any built-in storage, we connect a small Seagate 5GB hard drive. The hard drive we use has the form factor of a small hockey puck. Richard Lussier, our local hardware maven, was able to package both the hard drive and the NSLU board tightly in a new enclosure, while maintaining the access to the other unused port. We suggest you do the same, if you can find your own Richard.

Figure 2. HAL Version 1.0

What about the Software?

HAL uses the open-source media distribution software Firefly Media Server (formerly known as mt-daapd), developed by Ron Pedde. Firefly servers stream media with Apple's daap protocol, making the HAL box accessible for anyone running iTunes or any other daap-enabled media player. And, Firefly does not have the five connections per day restriction of iTunes servers, which is a plus.

To install Firefly, you need to have Linux on the NSLU2. Because the NSLU2 is an ARM architecture, you need Linux binaries that have been cross-compiled for the NSLU2. If you want to try the system before flashing anything, you can install the x86 binary packages for Windows and Linux on your computer.

The OpenSlug distro contains most of the needed tools and libraries, already cross-compiled and ready to go. Whatever was missing we cross-compiled ourselves, and we put the resulting binaries on the Web for you to use. Near the end of the installation instructions below, you will launch a script that will download and install them.

To simplify the daap stream discovery process, we use multicast dns (m-dns) technology as defined by the IETF's Zeroconf Working Group. This

Besides promoting serendipitous discovery, the user gets to interact with the content using a familiar interface that is specifically designed for rich media.

is the same technology that printer manufacturers employ to make installation and configuration seamless for Mac users. We use the m-dns dæmon included in Firefly, which does not implement any of the extra functionality available in the protocol beside daap. This is okay; daap is all we need.

Finally, we push the content to the HAL boxes from a central server via rsync and a series of small bash scripts.

Time to Install

Let's get ready to hack the box. For this article, we skip the Mac OS X instructions. It is a special case that gets complicated; visit our Web site for more information. Otherwise, it's a four-step procedure: flash the device, move the operating system to the hard drive, install Firefly Media Server and customize your configuration.

First, you need flashing software. Under Microsoft Windows, use Sercomm's utility, and under Linux use upslug2. You can find both of these via our Web site at **www.halproject.net/wiki/Hal-LinuxJournal**.

Then, download "OpenSlug firmware for NSLU2, binaries version" for the distribution page, which you also can reach via our Web site.

Be careful—this next step is the one that you do not want to mess up. Hold down the reset button and power-on your NSLU. Release the reset button when the yellow light turns red (about ten seconds). If everything worked, NSLU's LED should blink green and red. This indicates that the NSLU is in upgrade mode. Now, follow your software's instructions to upload the firmware. Within about three minutes of initiating the transfer, the software should indicate that the flash procedure was successful.

Restart the NSLU. At this point your hard drive is still sitting on your desk, unplugged. At the end of the boot sequence, once the light on the NSLU stops blinking, connect your hard drive to the first USB port (the one near the power source).

Log in to the box via SSH. Depending on the device's version, past settings and the stellar alignment, the IP could be 192.168.1.77 (Linksys' default), a static address you configured before, or it could have been obtained via DHCP. The user name is root, and the password is opeNSLUg.

Once logged in, use fdisk to create partitions on the sda device. We use the following schema:

```
/dev/sda1 : 500 megs, type 82 (linux)
/dev/sda2 : 258 megs, type 83 (swap)
/dev/sda3 : "the rest", type 82 (linux)
```

The first partition is for the operating system (mounted on /). The second is the Linux swap. The third is going to be mounted on /home/musique by the installation script.

With the partitions in place, create the filesystem (nslu> is the prompt):

```
nslu> mkreiserfs -q /dev/sda1 ; mkreiserfs -q /dev/sda3
nslu> halt
```

The NSLU will turn itself off. Unplug the hard drive, and restart the NSLU. Once it is booted, ssh in, replug the USB hard drive in the same port (the one near the power, remember?), and launch the following three commands:

```
nslu> turnup init
nslu> turnup disk -i /dev/sda1 -t reiserfs
nslu> reboot
```

The first command returns all kinds of questions (new root password,

hostname, network information); the second copies the OpenSlug operating system to the hard drive, and the third reboots the NSLU. From then on, there is no need to remove the hard drive again.

If everything went well, you now have OpenSlug installed, with your own hostname and your own custom network settings. This gives you a great little Linux box with which you can run all kinds of software. The package system of OpenSlug is ipkg. Get going!

Media Server
Installing HAL is really easy—really. All you need to do is get the admin.sh script from the HAL_Project server:

```
nslu> wget http://files.halproject.net/lj/admin.sh
nslu> sh admin.sh
```

This script installs all the other required parts (such as mt-daapd, OpenSSH, rsync, libraries and so forth).

Fine-Tuning
You will want to change the default configuration. Check the `HAL-Help` command for more information. You also should run `HAL-SetName` to change the name advertised to iTunes clients.

That's it. That's all the knowledge you need to build a HAL box from scratch. Plug your HAL box in to your network to see your now-empty share automatically appear. You can add media sources with the `HAL-AddSources` command at the OpenSlug prompt.

Future Development
We would like to switch to an mdns server that is more powerful than the one distributed with Firefly Media Server. In particular, we would like to advertise other services in addition to daap shares. Imagine locative bookmarks that would automatically, and only temporarily, be added to users' browsers (Safari already supports this feature), or collaborative tools like SubEthaEdit.

Another feature high up on the to-do list is completing the central server. Media synchronization is easy with two or three HAL boxes, but in larger HAL deployments, central management tools become a necessity.

We also are investigating other hardware platforms. This article focuses on the Linksys NSLU2, but many other fun pieces of hardware exist. The ASUS WL-HDD2.5 pairs a 2.5" hard drive enclosure with a Wi-Fi radio, which would be ideal for HAL. But, its CPU is a lot slower than that of the NSLU, and its memory is almost non-existent, so it is not clear whether our software fits or whether it can be made to fit. The device is on our order-and-test list, along with many others.

Another aspect of the project open for further development is copy protection. Content providers (in our case, student-run radio stations and artist groups) are more ready to contribute media for the project when they are confident that it won't end up on a P2P network the next day. We know techniques exist for ripping content from a daap feed, but we will be working hard to limit those possibilities (knowing we won't be able to eliminate them all).

While keeping this technology suitable to individual HAL use, we're excited to bring this project to a larger scale. We have 12 boxes currently deployed, and we plan to expand to 25 by the end of 2006. Also, we hope to assist the community wireless group WirelessToronto set up its own network of HAL boxes in the near future. The goal of creating a richer, more diverse, more accessed local culture is a lofty one, but hopefully this project will have an impact.

The HAL Project has much work ahead of it. We look forward to hearing from people who feel like rolling up their sleeves and joining in.∎

Resources for this article: www.linuxjournal.com/article/9459.

Pascal Charest is a network consultant. He's the technical coordinator of the HAL Project as well as a board member of Ile Sans Fil. He spends too many nights hacking hardware.

Michael Lenczner helps develop free information infrastructures through his organization CivicSense.ca. He is the cofounder of Ile Sans Fil and the nontechnical coordinator of the Montréal HAL Project.

Guillaume Marceau is a computer science graduate student at Brown University and a new volunteer with Ile Sans Fil.

Growing a World of Linux Professionals

LPI-Japan

LPI-China

LPI-North America

LPI-South Asia

LPI-Deutsch

LPI-Latin America
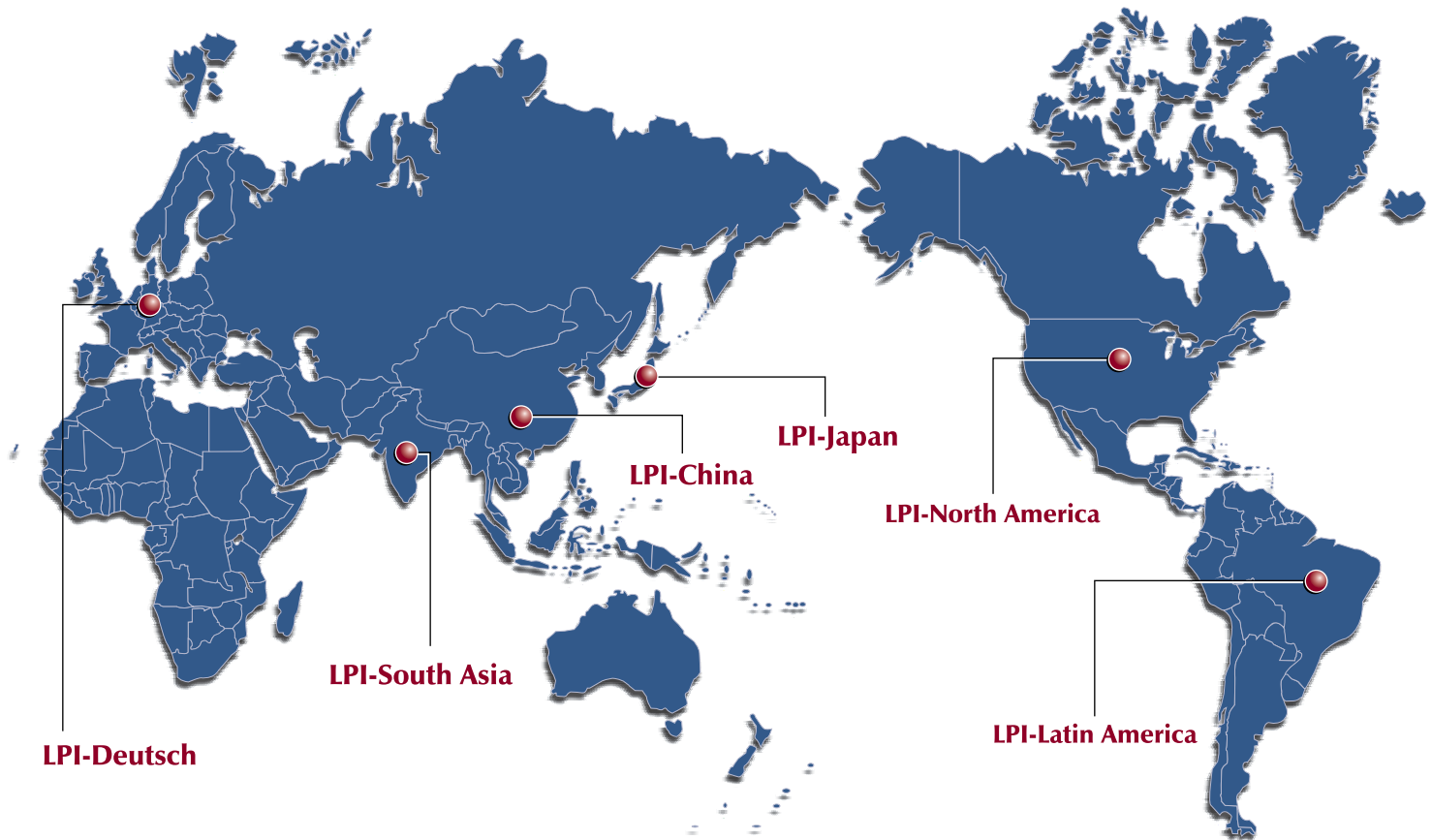
We at the Linux Professional Institute believe the best way to spread the adoption of Linux and Open Source software is to grow a world wide supply of talented, qualified and accredited IT professionals.

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.

Linux
Professional
Institute

Stable. Innovative. Growing.

HOW TO
Port Linux
When the Hardware
TURNS SOFT

Porting Linux to run on the Pico E12 and beyond.

DAVID LYNCH

In software development, possibly the most mystical and prestigious effort is taking dead hardware and breathing life into it—porting an operating system to a new platform—the mythical land of wizards and gurus, the software side of *The Soul of a New Machine*. I had performed almost every other software development task, and I wanted a chance to conquer this one.

I had been working with Linux and open-source software for many years. I am a fairly competent software developer (with hardware experience), but prior to starting the E12 port, I had done little more than tweak a Linux driver and build custom configured kernels. I was fortunate to have a friend building a new company

that was developing one of the smallest embedded systems available, the Pico E12. I practically begged for the opportunity to put Linux on the E12. "A man's reach should exceed his grasp, or what's a heaven for?"

The E12 used a Xilinx Virtex 4 FX20 FPGA (Field Programmable Gate Array) that included a 300MHz PowerPC 405 processor, 128MB of memory and 64MB of Flash ROM. I bought a Macintosh Lombard PowerBook Laptop on eBay, as a sort of simulator for the E12. It also provided a way to write for the E12 without a cross compiler. While waiting for the E12 to progress far enough to start working with it, I scoured the Web for information about Linux porting

and developed competence in PowerPC assembly language. Linux kernel programming is primarily in C, but small parts of the Linux kernel— parts critical to putting Linux on a new system are in assembler. I have programmed in many assemblers—once writing the standard C library in x86 assembler, but PPC assembler was new and took a day or two to learn. Linux had been ported to PowerPCs, even a different Xilinx FPGA, long ago.

I have a reference library of software books that fills a three-car garage. With few exceptions, they gather dust. My primary research tool today is a broadband Internet connection and a search engine. There are vast resources
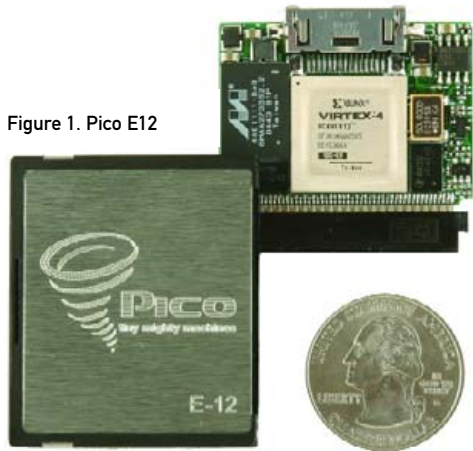
Figure 1. Pico E12

Figure 2. E12 in PCMCIA Adapter

available on the Web for Linux developers. The Linux Device Driver guide—the Linux bible for device drivers—and numerous mailing lists target all aspects of Linux systems development. Kernel-Newbies is a great place to start (see the on-line Resources). There are mailing lists for every Linux subsystem. And, there are several Linux PowerPC mailing lists—one specific to embedded PowerPC Linux. At the root of this tree is LKML, the Linux Kernel Mailing List. LKML is Mount Olympus—the home of Linus, and the other Linux gods and titans. There are Web pages documenting the experience of others porting Linux to specific boards. Finally, the ultimate reference—the Linux kernel source—is available on kernel.org.

Finally, the E12 was far enough along to start work, and I received one via FedEx. I had documents and specifications, but actually holding one made it real and answered questions that could not be read from the specifications.

Pico provided tools for hosted development. The standard E12 BIT file provided a CF interface with a simulated LPT3/JTAG port, a 512-word bidirectional communications FIFO called the keyhole, and host access to the Flash ROM. Pico also provided host-side Windows and Linux drivers that allowed reading and writing the Flash ROM. The normal FPGA BIT image contains a very small PPC monitor program that can perform a small number of tasks—most of which rely heavily on support from the host. One of those functions is the ability to load two types of files into the E12. It can load a new BIT image or load and execute binary ELF files—a simple bootloader. This saved me the difficulty of porting a bootloader, such as U-Boot. The Linux kernel was the most complex ELF file that the E12 monitor program had loaded to this point, and a few tweaks were needed to the loader.

Getting a Pulse

My first objective was to write the proverbial "Hello World" program for the E12. I spent a few days and wrote two different "Hello World" programs: one for the keyhole FIFO and one for Xilinx uartlite port.

Now, I was ready to attack Linux. I decided to start with Linux 2.6. There were numerous issues—good reasons, as well as respected and conflicting opinions favoring both 2.4 and 2.6. I elected to use Linux 2.6, because I eventually was going to have to move to 2.6 anyway. Initially, I used the PowerBook to configure and build my Linux kernel for the Pico E12. This allowed me to start without cross compilers. Eventually, I switched to building inside of a coLinux virtual machine on Windows hosting the E12. Most Pico clients are doing Windows-hosted development. It was critical that everything work in that environment. Besides, building a PowerPC Linux kernel in a Linux virtual

machine running Windows and loading it into a PowerPC, means that Linux outnumbers Windows 2 to 1 inside my laptop.

I used the Xilinx ML300 as a template to create a new Linux BSP (Broad Support Package). I grepped the kernel source for all references to the Xilinx ML300. I copied and renamed all ML300-related files to new files for the Pico E12. There were four completely unique files for the E12:

► arch/ppc/platforms/4xx/pic0_e1x.c: board-specific setup code.

► arch/ppc/platforms/4xx/pic0_e1x.h: headers and data structures for the board-specific setup code.

► arch/ppc/platforms/4xx/xparameters/xparameters_pic0_e1x.h: a set of hardware definitions created by the Xilinx software that created the bit image for the FPGA.

► arch/ppc/configs/defconfig_pic0_e1x: a default Linux configuration file for the E12.

There were major similarities between the Xilinx ML300, but there were a few specific differences. The E12 deliberately implements a lot less hardware. The E12's purpose is to provide a very minimal base platform, with the largest percentage of FPGA left for the client. The minimal useful Linux configuration must have either Ethernet, a serial port or the keyhole port. The default E12 does not have an interrupt controller—the PPC405 provides a timer interrupt that does not require a PIC. The E12 also uses the Xilinx uartlite uart, not the much larger and more common 16550 uart. There were no Linux drivers for the uartlite. Two other ML300 files, generic support for Virtex FPGAs, required minor modifications.

The next major issue was learning the Linux configuration system. I was not able to find much documentation. With Linux kernel programming, the two primary resources are the Linux source itself and the mailing lists.

The E12

The E12 is a Compact Flash card—exactly like those in many digital cameras. It has only two connectors: a CF bus connection on one end and a 15-pin miniature connector on the other. There are no other external connections. The E12 is based on an FPGA. There are a few additional components, and a few fixed elements, such as the PPC405 CPU on the FPGA. A large part of the hardware is programmable. Most external connections are through the FPGA. Almost none of the "hardware" has form or meaning until the FPGA is loaded. Changing the bit file on the fly drops in a completely new hardware design. Welcome to a new era—even the hardware is software. The BIT image—in essence the program for the FPGA—is created by an FPGA developer, programmed into the Flash ROM and automagically loaded into the FPGA on power-up. Once this BIT image "boots", hardware is created in the FPGA. Now, the pins on the connectors have meaning. The 15-pin connector provides three external connections for internal devices. It supports Ethernet, serial and JTAG connections through custom cables. The CF connector provides a bidirectional interface to a host—in most instances using a CardBus or PCMCIA adapter. Most of the pins on either connector can be whatever the FPGA programmer chooses to make them. Fielded systems may be plugged in to a CF connector solely to get power. E12's are used in daughter cards in typical embedded applications, on bus boards in high-performance computers in clusters and for applications, such as image processing or code cracking. They also are being used in applications with no operating system or extremely minimal operating systems.

Sometimes, there is excellent documentation for a system; sometimes there is nothing. Sometimes I found documentation in some obscure corner of the Web—after I had figured things out on my own. I had to develop enough understanding of the kernel build system to add a new board, some new configuration options and a few new drivers to the build system.

The first element is the Kconfig files in most of the Linux source directories. Kconfig is a cross between a very, very simple scripting language and a menu construction language. The entries in Kconfig files determine the menu structure and choices that you get when you execute any of the Linux menu configuration build options—make oldconfig, make menuconfig, make xconfig.

I had to create a new menu item under the ppc 4xx menu for the Pico E12, menu items in the drivers/serial/Kconfig file for the uartlite and keyhole serial ports, and a small collection of menu items for other options. The syntax for the Kconfig items I needed to create could be easily worked out by inspection and a small amount of trial and error. I copied blocks for similar objects, made name changes as needed, and without too much effort, it worked. Inside the .config file, source code and Makefiles, the configuration items defined in Kconfig files are prefixed with CONFIG_. After the Kconfig entries were created, entries needed to be added to the matching Makefiles. This mostly involved copying similar objects and making name changes, and except for a few very complex choices, was pretty easy.

So far, I had done very little actual coding. Most of what I had done was remove ML300-specific code from the new Pico E12 copy. I also copied the Xilinx PIC driver and created a stripped-out dummy PIC driver.

I was now able to build a Linux kernel for the Pico E12, without serial or Ethernet drivers. I still needed to write two serial device drivers: uartlite.c and keyhole.c. I deliberately chose to use the 8250 driver as a template—8250s and their numerous successors are ubiquitous, probably making up more serial devices than all others combined. I assumed that the 8250 driver would be, by far, the most stable and well-debugged serial driver. Also, many 8250-based systems are known to have problems with interrupts, so I knew that the Linux 8250 driver had to work without interrupts. This turned out to be a bad choice. The Linux 8250 driver is probably, by far, the most complex Linux serial driver.

Eventually, I remodeled my drivers based on the m32r_sio driver. I did not know much about the m32r_sio, but the driver was clean and simple, had all the features I needed and none that I did not. I also had to create a set of serial port headers for the keyhole and the uartlite, defining the uart registers and the bits within the registers. I also modeled these directly on the 8250, which was a much better decision. I have been writing uart code, including software uarts, for a long time. Writing the device-specific code for the keyhole and uartlite was simple. Fewer than a dozen lines of code were needed

to send and receive a character. The uartlite and keyhole, like most Linux serial devices, do not have modem control and operate at a single speed. The few lines of code needed to send a character were also useful elsewhere for debugging. The keyhole is not a real serial device, but it can be made to look like one to Linux and then used as a console when the E12 is hosted. This was very important.

Connecting a rat's nest of cables to the host computer and to the tiny external connector on the E12 for Ethernet and the uartlite serial port created problems. The time testing every cable connection to assure that one had not come loose prior to trying a new kernel was greater than the time writing and testing code. I wore out or damaged several external connectors before I was done. When using the keyhole, all the connections between the E12 and the host are internal. It was also useful to send debugging through one device using the other as the console. The keyhole had one other attribute that came in extremely handy—I could write 16- or 32-bit values to one register as a single output instruction and see the data on the host side. This was critical when debugging PowerPC assembly code. Inserting code to display a value or trace execution needed to be done using few instructions, minimal side effects and assuming very little was working. Outputting values directly to the keyhole port became my equivalent to flashing an LED connected to an I/O port. It was equally simple and slightly more powerful.

To some extent, all software development is working in the dark, but embedded board bring-up is particularly so. Output is a flashlight letting you see a little bit of what is going on. The E12 has provisions for JTAG debugging, either through the emulated parallel port or the 15-pin connector. The Linux kernel provides kgdb and xmon support. These presume support on the host side and working hardware and drivers on the target. Linux also provides options for outputting progress and debugging prior to loading the console driver. These were limited primarily to 8250-compatible uarts. I added uartlite and keyhole ports to the early text debugging devices. Aside from persuading Linux to use it, this primarily involved supplying a few lines of code to output a character. I have the skills needed to use debugging tools from logic analyzers to gdb. Most of the time, I find that sophisticated tools provide massive amounts of additional information, obfuscating the problem rather than revealing it. But debugging is a religious art with competing sects, each with their own dogma.

Once I had working output routines for the uartlite and the keyhole, a stripped version of the ML300 code for the E12 and modified Kconfig and Makefiles for the E12, I was ready to build a kernel and try it. The normal Linux build process for the PowerPC leaves a kernel image in ELF format in arch/ppc/boot/images as zImage.elf. I copied this from the PowerBook I was using to build Linux kernels onto the host computer for the E12, and I used PicoUtil to replace the current Linux kernel image on the

E12 Flash. I used the E12's monitor to execute the ELF file. The Linux boot process is similar across platforms and boot methods. In my instance, the zImage.elf file loaded at 0x40000000 and started with a small wrapper that did some early hardware setup, decompressed and relocated the actual Linux kernel and then jumped to the early Linux setup code. I copied the simple character output routines for the keyhole and uartlite into the files arch/ppc/boot/simple/keyhole_tty.c and uartlite_tty.c, and these provided debugging output during the wrapper execution.

My first big problem was that the memory map of the E12 had the Flash starting at physical address 0 and the RAM at a higher physical address. Advice I received on the Linux PPC embedded mailing list suggested I really, really did not want to try to port Linux to a board without RAM at 0, if it was humanly possible to persuade the board designers to change the memory map. There have been previous and subsequent efforts to modify Linux to support systems where RAM does not start at physical address 0. I believe that is less of an issue now. Still, I took the advice, and after a few hours of begging, Pico agreed to re-organize memoryped to 0. The soft hardware meant that they were able to provide me with a new bit image with RAM mapped to 0 within a few hours.

For a while, I also ran my own customized version of the monitor program, passing a board information structure Linux expected with a small amount of information on memory size, processor speed and the mac address to use for the NIC. Eventually, these modifications were incorporated into the standard Pico monitor.

The best documentation for the boot process as it applied to my system was in the MontaVista comments at the top of xilinx_ml300.c. These did not cover the decompression and relocation wrapper, but exposed the rest of the boot process.

First Breath

The next significant problem was in arch/ppc/kernel/head_4xx.S. Here, Linux does basic MMU and exception handling setup, then uses an rfi instruction to transition from "real" mode to "virtual" mode and continue with the kernel initialization. I was able to execute right up to that rfi. I was able to check all the obvious conditions for successfully executing the rfi. However, I never ended up at start_here—where the rfi should have continued. I spent days developing an understanding of the Linux Virtual Memory system—most of the documentation x86-specific. And, I became more knowledgeable about the PowerPC MMU, a fairly simple device compared to the x86 MMU. It is basically a 64-entry address translation table. Virtual memory OSes inevitably use more than 64 virtual-physical addresses mappings, region sizes and privileges. A reference to a virtual address not in the MMU, or one that violates the privilege bits set for that entry, causes an exception, and it is the OS's responsibility to sort it out using whatever algorithms, methods and data that suits it.

The fault processing might take longer, as it is not handled in hardware, but it is more flexible, adaptable and less resource-intensive. There are no gigantic fixed mapping tables in dedicated regions of physical memory, as required on some other processors.

But, I still could not figure out why the rfi was not executing correctly. I added all kinds of additional entries to the MMU, assuming that I was actually successfully switching to virtual mode but unable to communicate, because my I/O ports were no longer accessible. I sprinkled the equivalent of "I am here" debugging markers throughout head_4xx.S and got my first clue. I was continuously looping through an exception handler. Every time I switched to virtual mode, I lost control of the PPC, regaining it again in real mode in the exception handler. I had the critical clues to figure things out, but I was still mystified.

Every problem can be solved if it can be divided into smaller pieces. Eventually, I realized that it was possible to transition from real to virtual mode in smaller increments rather than all at once as the rfi did. I was able to turn on address translation for data and turn it back off without ill effects. I was able to add 1–1 physical to virtual address mappings for my keyhole debug port to the MMU, turn it on to do some output and turn it off. With more effort, I was able to turn on instruction address translation execute code and turn it back off.

That is when it finally dawned on me that the problem had nothing to do with switching from real to virtual mode, but that something else being set by the rfi must be enabling an exception that was not occurring otherwise. So, I tested the bits in MSR_KERNEL—the PPC machine status register value Linux uses—one bit at a time, until I discovered that anytime I set MSR_CE, enabling machine check exceptions, I lost control. I redefined the macro that set MSR_KERNEL so that it did not set MSR_CE for the E12 and reported to Pico that I thought there was a hardware problem in the E12. Pico never found the problem, but six months later, updates to Xilinx's firmware building blocks corrected the problem.

After working around the machine-check problem, I suddenly found Linux booting all the way through to setting up the serial/console driver. I was stalled for a few days while I actually finished the serial drivers for the keyhole and uartlite. Linux needs a place to hold the root filesystem. There are many possibilities. Frequently, the norm for embedded systems is to place the root filesystem for an embedded development environment on an NFS share on another machine. This requires a working Ethernet driver. My confidence in my serial drivers was not high at that point. Further, the Pico minimalist mantra does not include networking as part of the base Linux, and many E12/Linux applications do not need it.

The root filesystem can be on a hard disk (none readily available in the E12) or in Flash. The E12 uses a very simple Pico File System, but one that is not suited for a root filesystem.

Another alternative was to put the root filesystem on a RAM disk. Linux provided the ability to use and populate a RAM filesystem as an intermediate step in the boot process. One objective was to migrate as much of the Linux boot code out of the kernel to user space as possible. Linux systems going back many years boot through initrd, then execute a pivot_root to switch the root filesystem from the initrd RAM disk to the disk-based root filesystem. Using initrd requires the loader to copy the compressed Linux image and the separate image of the contents of the initial RAM disk into memory, and provides Linux with a pointer to the initial RAM disk data.

Linux 2.6 introduced a new variation— initramfs. One difference between initramfs and initrd was that with initramfs, the contents of the initial root RAM disk filesystem were compressed into the Linux image during build, so there was only one file—in my case an ELF file—to load. This meant that the Pico monitor would not need changes. This initramfs approach proved to be extremely clean, simple and easy to use. Getting it working was complex and time consuming, because initramfs is fairly new. The primary documentation is a collection of posts to LKML. To create an initramfs for the Pico E12, I determined that I needed to

create a directory on my build system and populate it with the files for the root filesystem. I enabled the initramfs option using menuconfig and told menuconfig where to find the directory that represented my root filesystem. There are a few other ways to do this, but that was the simplest. Initially, I decompressed the initramfs from the Gentoo Linux install on my PowerBook. I eventually switched to a cross-compiled BusyBox when I erroneously thought I might be having problems with my boot image, because the binaries were built for the PowerBook, not a PPC405.

First Words
After this, I hit my next problem. Linux was booting all the way though to executing /init where it just stopped. I wrote a trivial version of ls and included it in the kernel, calling it prior to exec'ing /init. Everything was fine. But on exec'ing /init, Linux became deaf and dumb. Debugging can be particularly difficult when the horses look like zebras. I spent a lot of time tracing through the Linux exec process, which was remarkably ingenious in many instances, doing minimal work and loading a process through page faults. Unfortunately, this made tracing what was happening very difficult and led me once again to the (almost) erroneous

conclusion that I had a virtual memory problem. I wrote a Linux version of "Hello World" in PPC assembler with no external libraries and was able to execute it as /init. But, I could not exec anything more complex. I eventually found and enabled system call tracing and was able to watch as /init executed. The system always died while in the middle of virtual memory operations. I ended up with failure cases when Linux would go dumb right in the middle of outputting some debug string—again, always during a VM operation. I could actually change the point of failure by inserting additional debugging. I was a victim of the Heisenberg uncertainty principle—observation changed the observed behavior.

I was sure something was wrong with my serial drivers, despite the fact that this did not make sense, but how else could output stop in the middle of a string? All the critical clues were present to solve this problem, though one of them was buried as an artifact of the machine-check problem. This was a VM problem, in a twisted sense, and it was a serial driver problem. I will not confess to how long it took for the answer to dawn on me. Let's just say I rewrote the serial drivers several times before I saw that although the serial drivers requested and saved a virtual address for the memory mapped hardware—partly as an error induced by using the 8250 serial driver as a starting point—the virtual address for the serial port was subsequently getting overwritten by the physical address of the port. Because in my efforts to debug the machine-check problem I put a 1–1 physical-virtual mapping directly into the MMU Translation Lookaside Buffer, I/O continued to work until the Linux VM system overwrote my temporary TLB entry. After recognizing this, it took less than 30 minutes to correct, and I was able to boot up Linux to a bash prompt.

The End—the Beginning
Little matches the thrill of seeing a new machine reach a shell prompt and knowing I made it happen. I had completed my base Pico E12 Linux port. Well, that is not quite true—no port is ever finished. When I completed my Pico E12 port, I was unaware of any other port of Linux to a Xilinx V4 chipset. Subsequently, a Linux port for the Xilinx ML403 by Grant Likely started working its way through Linux embedded PPC development trees and has been accepted into the distribution kernel. The Pico E12 is distinct from the ML403, but they are more similar to each other than the older ML300 from which I started. Grant's ML403 port reflected changes that were impacting the whole Linux PPC development tree, so I made my Pico E12 port track those developments.

I have depended on the keyhole port for hosted development work, and as a result, the keyhole serial driver gradually has grown smaller and more consistent with the direction in which Linux serial drivers seem to be headed. I will have to update the uartlite driver to catch up.

I am currently on my third iteration of a Linux network driver for the E12, and Pico is on

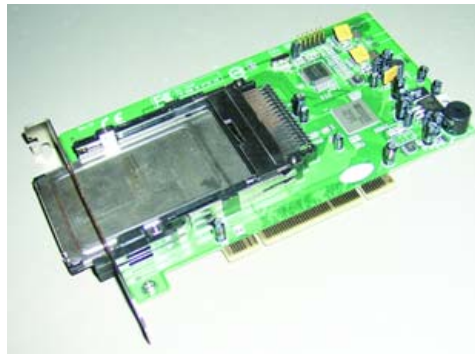Figure 3. Xilinx ML403 Development System

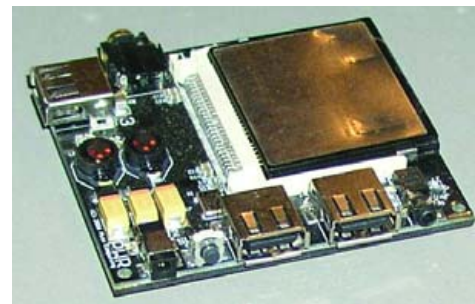Figure 4. Pico E14, Bigger, Faster, More Memory, More FPGA

Figure 5. E12 in Little Brother Board Daughter Card

its second iteration of the underlying network hardware. The new network hardware is interrupt-driven requiring a PIC.

A second Pico board has matured, and with minimal changes, the Pico E12 port has evolved into the Pico E1X port.

I am working to get the Linux Memory Technology Device (MTD) system to work with the Pico Flash. This is complicated by the fact that the Flash in Pico hardware can be read and written to by both Linux and the host, and Pico is eventually planning on Flash device sizes that should be windowed into Linux memory instead of mapped in their entirety.

Once the Linux MTD work is completed, Pico wants a Linux (and Windows) filesystem driver for its simple filesystem—PicoFS.

Pico is considering changing the keyhole port so that on the host side (and possibly the target side), it sufficiently and closely resembles an 8250-compatible UART to use only the OS's native serial drivers.

> "The bigger game was 'pinball'. You win one game, you get to play another. You win with this machine, you get to build the next. Pinball was what counted."—Tracy Kidder, *The Soul of a New Machine*.
>
> **I won at pinball.**

Later, Pico developed a daughterboard for the E12 called the Little Brother Board that allows using the E12 in a non-hosted environment and includes three USB ports, an LED and several other hardware components. In one application, the E12/LBB combination is being used as a very high-performance Webcamera.

The E12 also can be hosted as a grid on a bus board called the supercluster. Currently, that configuration is used for blazingly fast code cracking, using FPGA hardware without OS support, but Linux HPC support is on the wish list. Higher performance can no longer be achieved simply by doubling the clock every 18 months. Clusters are a significant alternative; 16 E12s provide enormous horsepower while occupying little space and consuming little power.

There have been several iterative releases to Linux 2.6 since the E12 port, and occasionally, these require changes to the port.

I would like to get the Pico E1X port and drivers I wrote for it included into the Linux distribution kernel. Within the Linux embedded PPC mailing list, there has been some interest in seeing that happen. The code has been moved into git to make it easier to merge with new Linux iterations and to produce patch files for submission to LKML.

I got Linux up and running on new hardware, and other opportunities with other hardware and with other embedded OSes have occurred. Board bring-up for the E12 was hard. Somewhere on Kernel-Newbies I read advice to newbie kernel hackers to lurk on the mailing lists for a few years before attempting anything serious—advice I am glad I did not take. I did not start this as a complete novice. I had a lot of experience that made this much easier. It was thrilling, mythical and magical. I can call myself a Linux Kernel Developer—though maybe not too loudly around Linus Torvalds, Andrew Morton or Alan Cox. But, it was not more difficult than many other software tasks—just more rewarding.

Acknowledgement

Resources for this article: www.linuxjournal.com/article/9462.

David Lynch is a software consultant. Programming is like art or music, he does it because he loves it. He is always seeking new and challenging software projects such as embedded board bring-up— preferably Linux/open source (www.dlasys.net).

Polywell Server and PC Solutions
Specialize for Small to Medium Size Business

High-End Servers and Workstations

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

Low-Cost Linux PCs

1U 4-way, 64GB DDR, 2TB RAID

- 2 x AMD® Opteron™ Dual-Core Processors 2218+
- with Hyper Transport Technology
- Upto 64GB 667/533MHz ECC DDR2 (16 Sockets)
- Upto 2TB 4 x 500G Swap Drive per 1U Rack
- 1 x PCI-X 133MHz or 1 x PCI-E RISER Solt
- Dual Gigabit Ethernet, ATI Graphics, 4 SATA-RAID
- CD-ROM Drive, Optional DVD-RW or CDRW
- SATA RAID-5 or SCSI Controller on-board
- 1U 24" Depth Rack Chassis with upto 600W PS
- 4 x Swappable Drive Bays (SATA or SCSI)
- Supports Linux, FreeBSD or Windows
- Custom Configuration Available
- Please call for other Options

1114SAS-2500A, 2218+,32GB, 2TB **$8,599**
1114SAS-2500A, 2212+,16GB, 1TB **$4,399**
1112ES-2500A, 2210E, 2GB, 500GB **$1,999**

4024AIS 4U 12TB SAS/SATA **$10,500**
4024AIS 4U 6TB StorageServer **$6,399**

Linux Appliance PCs

- Custom Made Odd Size Chassis
- AC or DC Power Supply
- Low Power Voltage AMD Sempron™ Processor
- or High Performance AMD Athlon™ 64 Processor
- Diskless or Flash OS Boot Drive
- Swapable Hard Drive, CD-ROM, FDD
- Integrated Graphics, Ethernet, USB
- Optional LCD LED Control Module
- IS2 Audio, MPEG2/4 Hardware Video
- Upto 4 Ethernet Ports or 4 Serial Ports
- We have over 18 years OEM Experience in Set top Box, Digital Media Player, POS Kiros, Thin Client, Networking Appliance,

Allows users to run 32- and 64-bit applications as they desire – without sacrificing performance

626TD112-800LX
OEM Appliance starts at **$299**

64GB DDR 4-way Workstation

AMD64 architecture reduces I/O bottlenecks, increases bandwidth, and reduces memory latency. Critical information gets to those who need it quickly and efficiently.

- 2 x AMD® Opteron™ Dual-Core Processors 285+
- Upto 64GB ECC DDR (16 Memory Sockets)
- 2 x 133MHz PCI-X, 1 x PCI-E x16 Slots
- Dual Gigabit Ethernet, 4 x SATA-RAID Controller
- Quiet and Cool 11-Bay Tower + 1400W 80Plus P/S
- 2x320G HD, DVD-RW, Floppy, Optional Card Reader
- 8-Layer Motherboard with Special Quiet Cooling
- Optional Sound Card, 1394 firewire
- Supports 64/32-bit Linux, FreeBSD or Windows
- Special for Large Memory Intensive Applications
- Built-to-Order or Configure-to-Order

PolyStation 2050M
4-Way Two Dual-Core Processors
Up to 64GB memory, 16 Sockets

64G RAM, 2x285+, QuadroFX5500 **$20,999**
32G RAM, 2x265+, QuadroFX3450 **$8,499**
16G RAM, 2x246+, QuadroFX1500 **$4,650**

DDR2 Socket-F is now available

1U Power Saving ISP Server

- AMD Sempron™ or Opteron™ Processor
- 512M DDR 400MHz Memory
- 80GB Hard Drive
- 10/100Mbit Ethernet
- Drive Image Service
- 1U 14" Short Rack, allow 2 x 1U per Rack
- Low Power Usage, Data Center Cost Saving
- Perfect Entry Level ISP Server or Appliance System
- Disk-Less, IDE Flash Drive Boot Option
- Supports Linux, FreeBSD or Windows
- Custom Configuration is Available
- Please call us to discuss your specification

Order# VX2500SP1U-17LJ11A
starts at **$399**

Pick the style for your application

2U 8-way, 5U 16-way Servers

- 8 or 4 AMD® Opteron™ Dual-Core Processors with Hyper Transport Technology
- Upto 128GB DDR Memory for 16-way (32 sockets)
- Upto 64GB DDR2 Memory for 8-way (16 sockets)
- 4x Gigabit LAN, 8x SATA RAID-5 for 16-way
- 2x Gigabit LAN, 4x SATA, U320 SCSI for 8-way
- 4 x 133/100/66MHz PCI-X Slots for 16-way
- 2 x 133, 2 x 66MHz PCI-X, 1x PCI Slots for 8-way
- On-board ATI Graphics, USB 2.0
- 5U 26" Rack 1300W 3+1 Redundant P/S 16-way
- 2U 27" Rack 700W PFC P/S for 8-way
- Supports Linux, FreeBSD or Windows
- Custom Configuration Available
- Please call for other Options

5U 16-way 865, 32GB, 2TB, 8801T5U **$19,999**
2U 8-way 8212, 8GB, 900GB, 8450A **$7,499**

Mainstream Linux Systems

- AMD® Athlon™64 or X2 Dual-Core Processors with Hyper Transport Technology
- 512MB to 4GB Dual Channel DDR Memory
- 80GB to 500GB Hard Drive (SATA-RAID-0/1/5)
- 2 PCI, 2 PCIe, Gigabit LAN, 7.1 Audio, 1394a
- nForce 430 Chipset, GeForce 6100 Graphics
- Desktop, Tower, MiniBox or Rackmount Chassis
- Internet Server or Linux Appliance Configuraiton

939NV3000TD12LX starts at **$399**

Polywell OEM Services, Your Virtual Manufacture
Prototype Development with Linux/FreeBSD Supports
Small Scale to Mass Production Manufacturing
Fulfillment, Shipping and RMA Repairing

AMD64 Investment Protection
Migrate to 64-bit platforms seamlessly.
Add 64-bit application as necessary.

888.765.9686
www.Polywell.com/us/LJ

POLYWELL

Polylwell has been in business since 1987, Our professional engineers have many years of experiences helping small to mid-size business to custom make or migrade products into Linux Appliances. We also provide Linux/FreeBSD Driver Development Services.

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.

An Interview with Marten Mickos of MySQL

MySQL has become a household word and a profitable business. GLYN MOODY

GM: I believe you knew one of the founders of MySQL a long time before you joined?

MM: I met Monty, the CTO, in 1981. We enrolled at the Helsinki University of Technology to study Technical Physics. Monty at that time was not an open-source developer, he was just an alpha geek. And I thought he was spoiling his life by not going to the parties and having fun but just working and programming all the time. But, he did build some amazing games and stuff that we played with.

When he started MySQL, I worked for this other small database company, Solid Information Technology. I told Monty that his project was just going to fail, and that it was a stupid thing to do, and that he didn't have a chance because we had a chance.

GM: What was your view of the Free Software world when you were at Solid—were you even aware of it?

MM: I was getting more aware of it, and I was getting excited about it. At Solid, I drove an initiative of not open-sourcing the product, but making it very popular on the Linux platform—and that was why I was an advertiser in *Linux Journal*, because we were the leading Linux database in the world in 1996. We gave it away free of charge, so we had taken a step in that direction.

Then Solid decided to cancel the project and just focus on high-end customers, and that's when I left the company. So in that sense, when I got to MySQL, I had some unfinished business. By that time, I had completely bought into the notion of code being open.

GM: What attracted you to MySQL?

MM: Now it sounds like we're reading some great wisdom into my decision, and I don't think there was any. MySQL at the time was not administered at all as a company. There was virtually no bookkeeping. There were no offices, no contracts, nothing. So in that sense, it was a garage startup and a big mess. But I

"With ten million installations worldwide, we're used for everything that relates to data."

knew about the enormous potential of the technology. Monty has saved an e-mail I sent to him in '97 where I said—I'm referring to some URL—"hey guys, you seem to be getting some traction." And that's the first time I admitted MySQL had a future.

GM: Unusually for an open-source project, all the copyright of the source code is held by the company MySQL AB. Where did that idea come from?
MM: There are some natural reasons for it. One is that the vast majority of the original source code was written by one man—Monty. Now his portion is much, much smaller, but at that time, most of the code was written by him. So it was natural that the copyright was held by the company. But second, Monty and David learned from the Ghostscript project. They were the first implementers of the dual-licensing model where you retain copyright but at the same time you release it under open source.

GM: Why did the company decide to adopt the GNU GPL in 2000?
MM: Initially, they had another dual license that said it's free on Linux but you pay on UNIX and Windows. And at some point, they realised to get included in the Linux distros, you needed a license that people could readily accept. People had nothing against the MySQL license, but it took time for them to read through it and accept it. And they argued that if they would adopt the GPL, there would be no questions asked.

When they made the decision, monthly sales fell to 20% of what it had been. So it was a huge risk financially for them—they had no financial backers, no VCs. There was a half year of slower sales and then they were back on track.

GM: You still have a commercial license alongside the GNU GPL. For what reasons do people choose the commercial license?
MM: The interesting thing is that we are known for the dual-licensing model, and as pioneers of it, but today our main business is not on dual licensing, because we are now becoming a major player in the enterprise market and with Web sites, and they don't buy commercial licenses from us, they buy subscriptions.

GM: You mean they use the GNU GPL license and pay for support?
MM: Yes. So dual licensing was a good starting

model for us, and it works well in the OEM space, where people "OEM" the code from us and put it into their own products that they ship to customers. And that's where it works very well. But if you look at our most famous customers, like Google and Yahoo, Travelocity and Craigslist, they do not use our commercial license.

GM: You have some very high-profile customers. What do they use MySQL for?
MM: With ten million installations worldwide, we're used for everything that relates to data. We're used for structured data, unstructured data, transactional data, non-transactional data. We're used in Web applications and business applications.

Take one example, Google. The system for its commercial ads, AdSense and AdWords, those two run on MySQL, so when you get ads popping up on your Google screen, you know we are there.

GM: What about Yahoo?
MM: They started by using it in Yahoo Finance, where they built a publishing system called Jake. All the news items and whatever content they publish came out through Jake and MySQL databases. And from there, it spread out to many of the gaming solutions and hundreds of applications within Yahoo.

GM: And Travelocity?
MM: There, MySQL is used for the airfare searches. So if you make an airline reservation, it still goes into the same HP NonStop [SQL] database that they've had there for some time, but all airfare searches go into our databases. Interestingly enough, it is the airfare searches that grow exponentially. There aren't too many seats being sold, because there aren't too many airplanes being flown today. But to make one reservation consumers can make tens, hundreds or thousands of searches first. So it shows a change in the landscape; it's not just the travel agents and the professionals who make very specific airline reservations and searches, it's everybody.

GM: What other kind of applications run on your database?
MM: Slashdot runs on MySQL. The *Spiderman* movie site runs on MySQL. The special effects in *Lord of the Rings* were built using MySQL. The Mars Rover has an earth-based control program that runs on MySQL.

GM: Do you think that the use of the LAMP stack—GNU/Linux, Apache, MySQL and Perl/PHP/Python—has become almost a given for a Web 2.0 startup?
MM: I think that's correct. In the Internet bubble, many companies had the thinking that they needed tons of VC money, and with that they bought Sun hardware, Oracle databases and BEA Web application servers. And today, you don't do that. You buy inexpensive hardware; you run the LAMP stack on it; you just get going. And then when you start scaling, that's when you need commercial help. So I think the interesting thing today is that you can start small, start on a single Intel-based server, and it costs you virtually nothing. And then, when you get going, you can scale it horizontally, without throwing away the original.

GM: Does that mean MySQL is not really up against Oracle as a competitor—that you tend to go for new companies?
MM: I would put it differently: they are not up against us when it comes to Web 2.0; we are among the pioneers there, the leaders there.

GM: What about in the traditional markets, do you find that you are starting to compete against Oracle?
MM: We do, but it's not a main area of focus for us. This is the major difference between us and the other open-source databases. Most of the others are trying to become a replacement for Oracle, so if you look at PostgreSQL, EnterpriseDB, Ingres and all those guys, they try to mimic the old-style databases so that they one day can claim that space. But my guess is that by that time, the space will be gone.

GM: What effect did Oracle's purchase of Innobase, which supplies InnoDB, one of the main database engines for MySQL, have on you and your customers when it was announced last year?
MM: I think it sent shock waves through the industry. People took it very seriously, especially the financial analysts and journalists. And even our customers saw it as a risk, and they came to us to ask what this meant and whether they were safe or not. And I think what we have shown in the last six months is that open source is such a self-healing ecosystem that if InnoDB truly had been taken out of the equation, there would quickly have been replacements. And there are replacements today.

GM: Where did MySQL's pluggable architecture, which allows different database engines to be used, come from?
MM: It was a smart design decision by Monty back in '95. He had built the first MySQL engine. He realised he needed to revise and upgrade the storage engine. But he was lazy, so he didn't want

to move over abruptly from one to the other. So he thought, what if I allow both of those to coexist at the same time? And when he did that, he had to create an API between the upper layer and the lower level. He didn't know at the time what a fantastic design decision it was.

In Web 2.0, the usage of data is much more varied today than it used to be in the old client-server world. If you have a big Web site, you have some data that is transactional, you have other data that is read-only but is needed in milliseconds, and then you have logging and archiving data that you typically don't need [immediately] but which needs to be available somewhere. By using different storage engines, you can cater to those various needs within the same database installation.

GM: What was the thinking behind your decision to work with SCO at a time when it was taking legal action against IBM that was seen as threatening to the Open Source world?
MM: We are not supportive of SCO's legal actions, and when they ask us for advice, we tell them to stop it and just get out of it and ask for forgiveness. We don't share their thinking there, but they have customers who need a database. Why wouldn't we sell our stuff there? With the money we get, we can hire more developers to develop more, cheaper software.

I think it's so easy to be black and white, but if you think twice, you realise this could be the best way to deal with the situation. Because now SCO cannot go out and say open source is bad, because they just bought a database license from us. Of course that won't change the litigation, but every little step counts.

GM: In March 2006 you joined the Eclipse foundation. What took you so long?
MM: That's a very relevant question. We just don't know how we could be asleep at the steering wheel like that. We should have joined a long time ago. It's just when you get too caught up in your own stuff, you don't act fast enough. But it was just wrong, we should have joined earlier.

GM: Moving on to corporate matters, at what points did you take funding?
MM: We did that in 2001, when I joined, and we got a professional board of directors, but it was only 4 million Euros. They said, this is exploding in our hands and we need to grow it, but we also need funding to grow it properly. So I helped them raise the first round. Without even having decided to join, I just said, I'm helping these guys.

And then in 2003, we raised our next round, 13 million Euros, mainly from Benchmark Capital and Index Ventures. And then early this year, we did a series C round, although we hadn't even consumed the previous round—we still had plenty of money left.

GM: Investors obviously expect something back at some point, so are you looking to get bought or to do an IPO?
MM: We're aiming for an IPO. We're actually aiming for an independent existence and to do that you need to do an IPO, but the IPO is not the aim, the IPO is just a step. People ask, "What is your exit plan?", and we say that we're not going to exit.

We think that when markets mature they tend to go horizontal, so you have players who specialise in certain components of the stack. Intel is a fantastic example on the hardware side. They produce processors for all vendors in the world, and nobody has acquired Intel. And, it makes sense for them to stick to their knitting and focus on what they are good at. We think that the database has a similar role in software—that it makes sense to have a team dedicated to data management: storing data, retrieving data, sorting data.

GM: Would you contemplate broadening your portfolio to include non-database products?
MM: I don't think so. I think we are fairly certain that we would not go into applications—that's for our partners to do. I don't think we would go down the stack into operating systems. But, I can see us being fairly innovative when it comes to dealing with data. Traditionally, a database was just a database. Then you had databases with replication. Now we have databases with different storage engines, and maybe you'll have databases with backup solutions and databases with storage solutions. So there's a world of expansion opportunities without having to go into applications.

GM: Looking at the broader open-source sector, do you expect there to be more consolidations in the wake of Red Hat's acquisition of JBoss?
MM: A few years ago, the common discussion was that open source is capable of competing with Microsoft and the closed source vendors specifically, because it isn't concentrated in one company, but it's a best of breed of a group—who's the enemy when there are so many? So, it was seen as a strength of open source. Now the winds are slightly different. People say Red Hat has grown so strong and look, they have acquired JBoss, but I think this discussion will go from side to side. One occurrence of an acquisition doesn't mean that there has to be more of them.

GM: Moving on to threats to open-source software, I notice there's a "No software patents" sticker and link throughout your Web site. How dangerous do you think software patents could be?
MM: We think software patents are the biggest threat not only to open-source vendors but also

to closed source vendors. And not only to vendors, but also to users, because software is being developed in bigger volumes by users than by vendors. The patents that are now being granted are so silly, so detailed, on such a low level, it is just inevitable that there will be enormous conflicts once the owners start thinking they must get a payback for the money they spent on acquiring them. We don't think it's specifically an open-source problem; we think the open-source companies and open-source people are the first to see the problems. It will harm the whole industry.

GM: Are you actively talking to people within the European Union on this subject?
MM: All the time. We have been surprisingly successful so far. We have had campaigns with poor funding but great results; whereas the pro-software patents camp has had great funding and poor results. But, it's a very difficult time because they come back every year with new proposals. So I'm very proud of what we've achieved so far, but I'm actually fairly pessimistic about the situation.

GM: What about in the US? Are you working to fight software patents there too?
MM: Not as much as in Europe, because in Europe, the legislation is still being written, whereas in the US it already exists. But it works both ways. In the US, they already see so much of the trouble with software patents, so there's a stronger movement against them, and Europe is still sleeping.

GM: Do you see any other major threats to open source?
MM: No. I think it's just a superior production model. Whatever happens with legislation or licenses, nothing can stop it when it's just inherently a superior model. People ask about GPL 3—when will it come out and will it be good and will people use it? It's an interesting question, but it doesn't affect the future of open source that much. There will be open source no matter what.

GM: Against this background, what do think will happen to Microsoft?
MM: They'll ultimately become an open-source company. I've never met Bill Gates, I don't know whether he's conservative or not, but I would actually expect him not to be. When he started Microsoft, he did the best thing he could, so why wouldn't he do it again? Open source wasn't available as an opportunity back then, so he couldn't choose it. If he started a company today, I bet it would be open source.■

Glyn Moody writes about free software and open source at opendotdotdot.

Evaluate. Integrate. Innovate.

Real-World Open Source Solutions in the Enterprise

Roll up your sleeves. Linux and open source already play a vital role in the enterprise. Now it's time to push it further—with proven best practices from IT pioneers who've used open source to enable robust, innovative solutions that deliver solid business impact. Two full days of real-world peer case studies, intensive technical training, insightful keynotes, and cutting-edge solutions prepare you for the next generation of open initiatives. Your datacenter will never be the same.

30+ In-depth Technical Sessions in 7 Tracks

Intensify your open source skill set with detailed technical instruction from open source pioneers like Jeremy Allison, Larry Augustin, Fabrizio Capobianco, Gerald Carter, Seth Grimes, and Mark Radcliffe in comprehensive tracks devoted to:

- Security
- Network Management and Interoperability
- Applications and Best Practices
- Virtualization
- Linux on the Desktop
- Legal
- Case Studies

> Get complete details at linuxworldsummit.com

Real-world Peer Case Studies

Analyze step-by-step case studies from the front lines of open source solution implementation presented by IT leaders from top enterprises like Nationwide, Pfizer, Solvay, Novell, SourceForge.net, BackCountry.com, and the Department of Defense. Find out what worked, what didn't—and how to apply the lessons learned in your own organization. **A key focus on six vertical markets** reveals best practices that can be applied in any business and industry:

- Financial Services
- Retail
- Media
- Public Sector
- Healthcare
- Pharmaceutical

Keynotes

Go beyond the hype as IT pioneers show how they're putting Linux and open source to work today to solve key enterprise integration and development challenges—and take home insights to power your own open innovation.

Bruce Schneier
Founder and CTO, Counterpane Internet Security, Inc.
The Economics of Information Security

Debra Anderson
CIO, Novell, Inc.
Open Source Adoption- A Real Life Story

Randy Allen
Corporate VP, AMD
Accelerating the Open Data Center

Solution Showcase

Get up-close with the latest solutions from the top Linux and open source vendors including :

AMD	Centrify	Google
Appro	Cleversafe.org	SugarCRM
Barracuda Networks	Egenera, Inc	SWsoft, Inc
CentricCRM	EMC Corporation	XenSource

And many more...

Register Today! Go to linuxworldsummit.com/register and use Priority Code SAMAG

LINUXWORLD OpenSolutions SUMMIT

February 14-15, 2007
New York Marriott Marquis, New York City

Creating a Lulu Book Cover with Pixel

Add the Pixel graphics program to your LyX-created book to finish the ultimate Lulu on-line book. DONALD EMMACK

Last month, I dove into LyX, a graphical LaTeX typesetting program for Linux. Because LyX produces commercial quality documents, it's an ideal match for the Lulu.com self-publishing Web site.

I'm reasonably certain most authors send material that comes from a standard word-processing program, such as Microsoft Word. Look back at the article in the December 2006 issue of *LJ* where I show you some output between a word processor and LyX to see what a difference it makes for your final product.

This time, I finish our book publishing tutorial. Publishing to Lulu.com is a two-step process. First, you need to upload your finished text files to Lulu. Then, you need to design a cover for the final product.

Final Touches

After Lulu accepts your text file, it asks you to format or upload a cover for the book. You have two choices: use one of Lulu's pre-designed cover backgrounds, or upload a custom book cover that you created.

The sample book covers Lulu provides vary and may suit your publication just fine. If you decide to use one of those, the on-line system will help you add text to the image. Lulu's on-line program will place the author, ISBN and copyright notice for you as well. However, creating a book cover yourself is a great opportunity to examine Pixel.

Pixel

In my estimation, Pixel is not a direct competitor of The GIMP. Pixel is a commercially produced application that runs on many operating systems. The Pixel Web site (**www.kanzelsberger.com/pixel**) notes at least six different operating systems, including Linux, Windows and Mac OS X.

By its own description, Pixel seems suited for advanced graphic artists. In fact, the first non-beta version of Pixel should include Photoshop plugins and .psd import and export. See **www.kanzelsberger.com/pixel/?page_id=60** for more details and a list of features for the next release.

In Support of Commercial Software

Pixel is not free. It is available as a demonstration software package, and users can buy a license for $32 US. This fee includes unlimited support and all updates until the next major release.

The demonstration copy does have a significant drawback. Any image created with Pixel contains a Pixel watermark. There is also a small "nag" screen reminding you to buy the product to get full use of the software. So, you can't use any graphics produced with the demo version.

I don't find this objectionable. Pixel's author intends to steer the software to compete with the leaders in the graphics industry. Although the expected version 1 price is around $100 US, it's still less than the competition. I like competition; it helps keep prices down.

Installation

Because Pixel is not open source or free, it's not likely you will find it in a major distribution's repository. So, download a demo copy from **www.kanzelsberger.com/pixel/?page_id=4**. Installation is straightforward. Download the Linux .tar file, unpack it and click on the file to start program installation. Follow the instructions on-screen to finish the setup.

Once complete, start Pixel from the command line or your system menu, and the home screen appears (Figure 1). At first impression, Pixel looks similar to other top-line commercial software. It's also different from The GIMP, because it covers your entire screen. In addition, many tools are in full view by default (Figure 2).

The screen layout is clean. Some of the icons are a bit troublesome to identify due to their size. But, trying to find hundreds of unique pictures for tools must be challenging for any programmer.

Now you're ready to begin. Because Pixel is still in beta release, documentation is scarce. In March 2006, the Pixel support forum explained that no documentation is ready as the focus is on product development. There is a help system available by pressing F1 (Figure 3), but currently no embedded tutorial exists. Even without much documentation, Pixel's layout is somewhat intuitive. If you know Adobe Photoshop, you're in luck—it's nearly the same.

The Best-Seller Cover

To review, in the last article we uploaded our sample text document in .pdf format to the Lulu Web site. Now, we need to create our custom cover. With Lulu, you can upload two different images, one for the front and one for the back. Or, you can design your own wrap-around cover, including the spine.
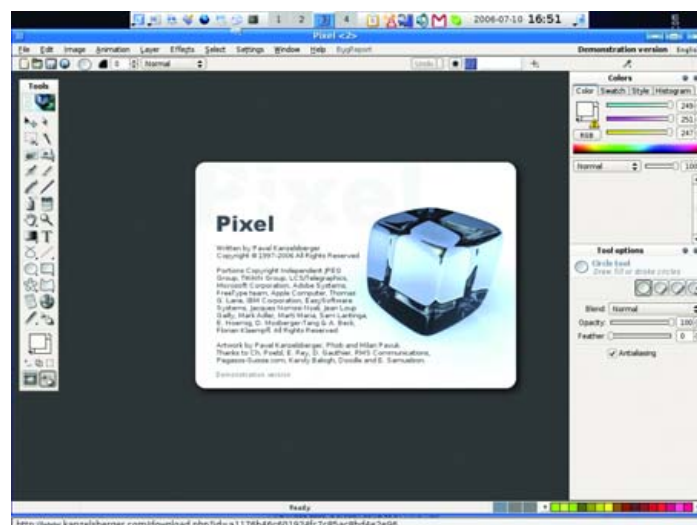

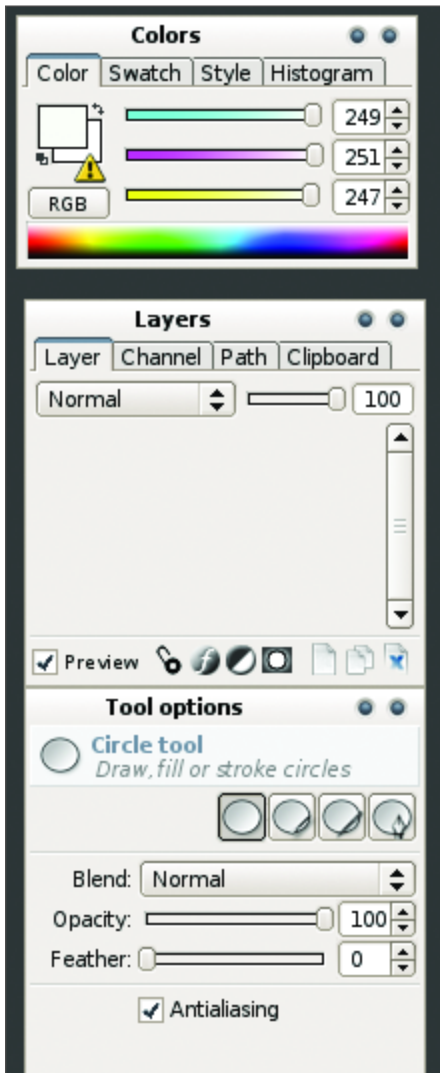Figure 1. The Home Base of Pixel

Figure 2. Multiple Toolboxes Default on the Screen
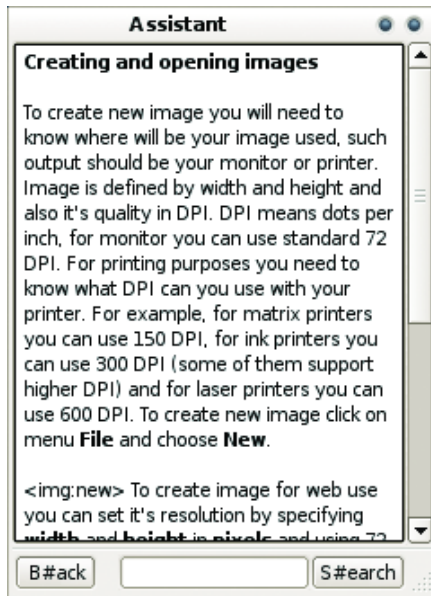

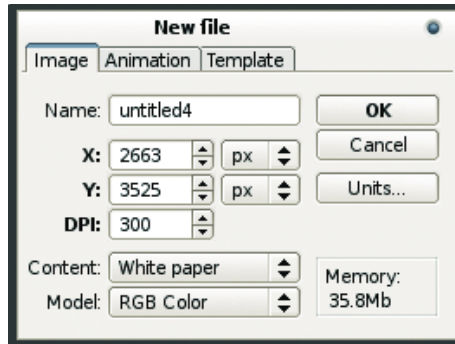Figure 3. Press F1 for the help assistant.
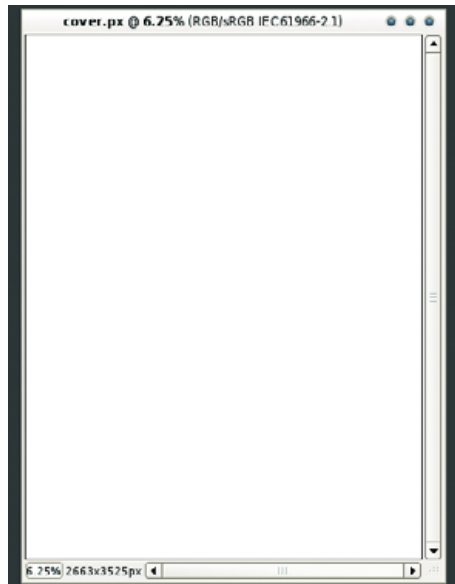

Figure 4. New File Creation Box

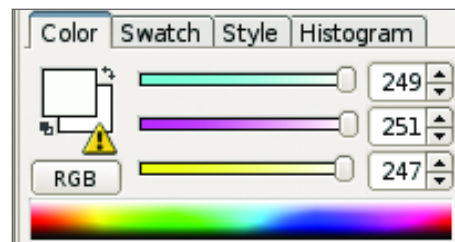Figure 5. The 8.5 x 11 Background for the Cover

Figure 6. Selecting Colors for the Book Cover
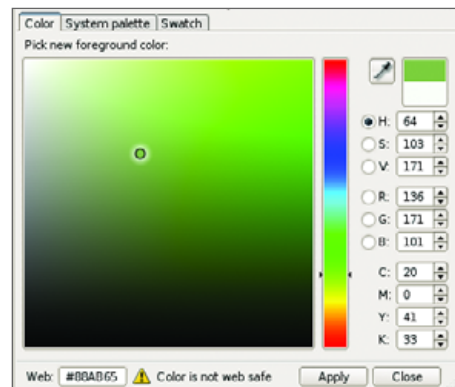

Figure 7. Pixel's Custom Color Picker
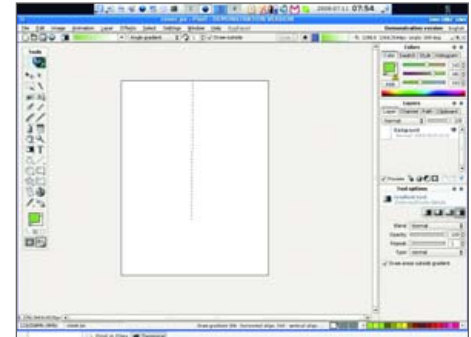

Figure 8. Select the Gradient Tool

Figure 9. Drag your mouse to tell Pixel the direction of the gradient.
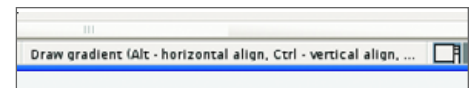

Figure 10. Pixel's Help Assistant

I chose to upload two different images for this tutorial. Wrap-around images look nice, but making one for an 8.5 x 11 book requires 1242 x 810 PostScript points. At 300 dpi, a file in Pixel can be large and difficult to manipulate.

Lulu's standards for an 8.5 x 11 book cover are 2663 x 3525 pixels and no less than 300 dpi. In Pixel, go to File→New, and a creation box opens (Figure 4). Enter the dimensions as shown in the example. In the lower-right corner, you will see the memory requirements for this file are 35.8MB. Press OK, and Pixel creates a blank document template (Figure 5). Now you have a blank page to create your cover art.

To keep artwork simple for the tutorial, use your mouse to change the foreground color as shown in Figure 6. Pixel opens a color chooser for you to select nearly any shade you want (Figure 7). Choose your colors wisely. Not all will transfer into the shade you expect during printing. You should consider using a color management system if you have specific needs.

Next, I decided to use Gradient G to spice up the background of the cover. Use your mouse to select the gradient button on the left of your screen (Figure 8). Drag your mouse by pressing the left-mouse button from the top of your cover to the bottom. This tells Pixel which direction to draw the gradient (Figure 9). I mentioned earlier that Pixel's operation is intuitive. When you select the gradient button, look
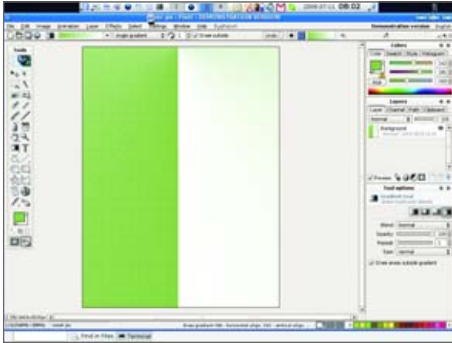
Figure 11. How Gradient G looks after screen refresh.

to the bottom right of the screen; the program gives you hints on how to use the feature or effect (Figure 10).

After the screen updates, you should have a sample cover that looks similar to the one shown in Figure 11.

Adding Text

Because a blank cover won't do much good on the bookshelf, we need to add text. Adding text is similar to adding color and gradients.

Use your mouse to select the text button (Figure 12). Position your cursor over any area on the working cover and use it to expand the text box. When complete, type and format the text for the cover (Figure 13). After typing the text, use the character controls in the bottom right-hand side of the screen to adjust any preferences with the text.

At this point, you can add other colors, images or just about anything else you like for the cover art.

Save and Upload Your Cover

When complete, go to File→Save As. Name your cover, and use the drop-down file-type list to choose .jpg. Pixel will prompt for further characteristics of the JPEG file. The default settings are acceptable for the tutorial.

My test book cover is shown in Figure 14. Log in to your Lulu.com account and upload your completed book cover into the system. This completes the design for your front cover.

For the back cover, I decided to use a solid black background. Lulu provides a sample black background for you to use. Simply use the on-line tools to "Choose Gallery Image". Lulu adds the cover into your publishing project. Select Save and Continue.

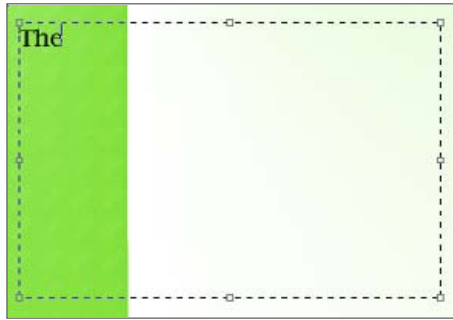Figure 12. Select the Text Tool

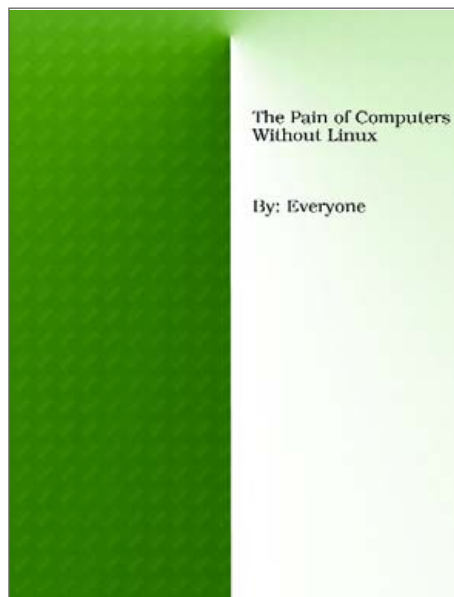Figure 13. Entering Text with the Text Box

Figure 14. Sample Book Cover

Figure 15. Lulu Cover Proof with Trim Marks

Final Publishing Steps

Lulu provides a snapshot of your final book cover in your Publishing section of the Web site. My final cover is shown in Figure 15. Watch the trim marks. Be careful that no important graphic or text is beyond the cutting line. Once you accept the cover design, it's easy to price your final publication and order a proofreading copy.

Publishing Wrap-Up

My first article described the benefits of quality text formats by using LyX to get typesetting output for your publication. Now we have completed the project by making a book cover and getting to the final proofreading stage with Lulu.com.

Using Pixel as a graphical editing package may cause some frustrations if you are a longtime GIMP user—not that Pixel can't match up to GIMP. Quite the contrary, Pixel targets a high-end graphical artist environment; however, it takes some time to become familiar with how to use the software.

Conclusion

Using Lulu.com, nearly anyone with the itch to write a book or magazine can create professional printed media. Using LyX and Pixel as tools for high-quality output may be the ideal combination for report and book formats. Scribus may better fit publications, such as magazines and newsletters. Lulu can print many types of documents—even calendars.
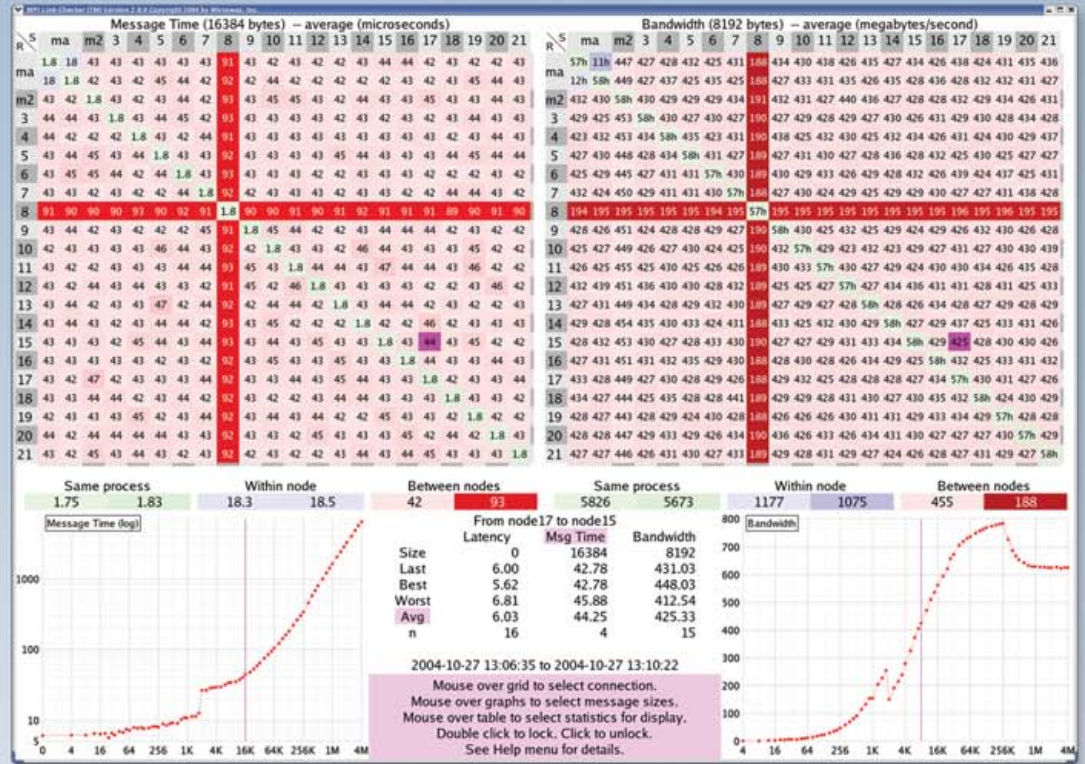
LyX is open source, and it appears to have support for further development as a GUI LaTeX editing package. On the other hand, Pixel is proprietary, and it seems its maintenance and development credit a small cadre of programmers, with Pavel Kanzelsberger as the leader. Other research on the Internet describes Mr Kanzelsberger as the only developer, yet the information screen of Pixel gives kudos to others. So, it might be a little risky to become too involved in Pixel until it matures a little more. I don't think you will need to wait long; the final release is expected soon.

Pixel is sharp. The likeness to Adobe Photoshop is sure to win the attention of graphic artists. I think it's safe to spend $32 US for the current version and look toward a final product in the next few months.■

Donald Emmack is Managing Partner of The IntelliGents & Co. He works extensively as a writer and business consultant in North America. You can reach him at donald@theintelligents.com or by cruising the 2 meter amateur RF bands in the Midwest.

An Automated Reliable Backup Solution

Creating an unattended, encrypted, redundant, network backup solution using Linux, Duplicity and COTS hardware. ANDREW DE PONTE

These days, it is common to fill huge hard drives with movies, music, videos, software, documents and many other forms of data. Manual backups to CD or DVD often are neglected because of the time-consuming manual intervention necessary to overcome media size limitations and data integrity issues. Hence, most of this data is not backed up on a regular basis. I work as a security professional, specifically in the area of software development. In my spare time, I am an open-source enthusiast and have developed a number of open-source projects. Given my broad spectrum of interests, I have a network in my home consisting of 12 computers, which run a combination of Linux, Mac OS X and Windows. Losing my work is unacceptable!

In order to function in my environment, a backup solution must accommodate multiple users of different machines, running different operating systems. All users must have the ability to back up and recover data in a flexible and unattended manner. This requires that data can be recovered at a granularity ranging from a single file to an entire archive stored at any specified date and time. Because multiple users can access the backup system, it is important to incorporate security functions, specifically data confidentiality, which prevents users from being able to see other users' data, and data integrity, which ensures that the data users recover from backups was originally created by them and was not altered.

In addition to security, reliability is another key requirement. The solution must be tolerant of individual hardware faults. In this case, the component most likely to fail is a hard drive, and therefore the solution should implement hard drive fault tolerance. Finally, the solution should use drive space and network bandwidth efficiently. Efficient use of bandwidth allows more users to back up their data simultaneously. Likewise, if hard drive space is used efficiently by each user, more data can be backed up. A few additional requirements that I impose on all of my projects are that they be visually attractive, of an appropriate size and reasonably priced.

I first attempted to find an existing solution. I found a number of solutions that fit into two categories: single-drive network backup appliances and RAID array network backup appliances. A prime example of a solution in the first category is the Western Digital NetCenter product. All of the products I found in this category failed in most, if not all, of the functionality, security, reliability and performance requirements. The appliances found in the second category are generally designed for enterprise use rather than personal use. Hence, they tend to be much more expensive than those found in the first category. The Snap Server 2200 is an example of one of the lower-end versions of an appliance that fits under the second category. It generally sells for about $1,000 US with a decent amount of hard drive space. The products I found in category two also failed in most, if not all, of the functionality, security, performance and general requirements.

Due to the excessive cost and requirements issues of the readily available solutions, I decided to build my own unattended,

encrypted, redundant, network-based backup solution using Linux, Duplicity and commercial off-the-shelf (COTS) hardware. Using these tools allowed me to create a network appliance that could make full and incremental backups, which are both encrypted and digitally signed. Incremental backups are backups in which only the changes since the last backup are saved. This reduces both the required storage and the required bandwidth for each backup. Full backups are backups in which the complete files, rather than just the changes, are backed up. These tools also provided the capabili-

Figure 1. Silver Venus 668 Case (Front)

Figure 2. Silver Venus 668 Case (Back)

ty of restoring both entire archives and single files backed up at a specified time. For, example, suppose I recently received a virus, and I know that a week ago I did not have the virus. This solution would easily allow me to restore my system as it was one week ago, or two months ago, or as far back as my first backup.

Duplicity, according to its project Web page, is a backup utility that backs up directories by encrypting tar-format volumes and uploading them to a remote or local file server. Duplicity, the cornerstone of this solution, is integrated with librsync, GnuPG and a number of file transport mechanisms. Duplicity provides a mechanism that meets my functionality, security and performance requirements.

Duplicity first uses librsync to create a tar-format volume consisting of either a full backup or an incremental backup. Then it uses GnuPG to encrypt and digitally sign the tar-format volume, providing the data confidentiality and integrity required. Once the tar-format volume is encrypted and signed, Duplicity transfers the backups to the specified location using one of its many supported file transportation mechanisms. In this case, I used the SSH file transportation mechanism, because it assures that the backups are encrypted while in transit. This is not necessary, as the backups are encrypted and signed prior to being transported, but it does add another layer of protection and complexity for someone trying to break in to the system. Furthermore, SSH is a commonly used service that eliminates the need to install another service, such as FTP, NFS or rsync.

The Hardware

Once I had committed to building this backup solution, I had to decide which hardware components I was going to use. Given my functionality, reliability, performance and general requirements, I decided to build a RAID 1—mirrored—array-based network solution. This meant that I needed two hard drives and a RAID controller that would support at least two hard drives.

I started by looking at small form-factor motherboards that I might use. I had used Mini-ITX motherboards in a number of other projects and knew that there was close to full Linux support for it. Given that this project did not require a fast CPU, I decided on the EPIA Mini-ITX ML8000A motherboard, which has an 800MHz CPU, a 100Mb network interface and one 32-bit PCI slot built in to it. This met my motherboard, CPU and network interface requirements and provided a PCI slot for the RAID controller.

After deciding on the form factor and motherboard, I had to choose a case and power supply that would provide enough space to fit a PCI hardware RAID controller, the Mini-ITX motherboard and two full-size hard drives, while complying with my general requirements. I compared a large number of Mini-ITX cases. I found only one, the Silver Venus 668, that was flexible enough to support everything I needed. After choosing the motherboard and case, I looked at the RAM requirement, and I chose 512MB of DDR266 RAM. I had great difficulty finding US Mini-ITX distributors. Luckily, I found a company, Logic Supply, which provided me with the motherboard, case, power supply and RAM as a package deal for a total of $301.25 US, including shipping. At this point, I had all of the components except the RAID controller and hard drives.

Finding a satisfactory RAID controller was extremely difficult. Many RAID controllers actually do their processing in operating system-level drivers rather than on a chip in the RAID controller card itself. The 3ware 8006-2LP SATA RAID Controller is a two-drive SATA controller that does its processing on the controller card. I acquired the 3ware 8006-2LP from Monarch Computer Systems for a total of $127.83 US, including shipping.

At this point, I needed only the hard drives. I eventually decided on buying two 200GB Western Digital #2000JS SATA300 8MB

Figure 3. Silver Venus 668 Case (Inside with Hardware)

Cache drives from Bytecom Systems, Inc., for a total of $176.69 US, including shipping. At this point, I had all of my hardware requirements satisfied. In the end, the hardware components for this system cost a total of $604.77 US—well below the approximate $1,000 US cost of the RAID array network appliances that failed to satisfy most of my requirements.

File Server
After building the computer, I decided to install Debian stable 3.1r2 on the newly built server's RAID array because of its superior package management system. I then installed an SSH dæmon so that the file server could be accessed securely. Once the SSH package was installed, I created a user account for myself on the file server. The user account home directory is where the backup data is stored, and all users who want to back up to the server will have their own accounts on the file server.

Client Setup
Once the file server was set up, I had to configure a computer to be backed up. Because Duplicity is integrated with GnuPG and SSH, I configured GnuPG and SSH to work unattended with Duplicity. I set up the following configuration on all the computers that I wanted to back up onto my newly created file server.

Installing Duplicity
I installed Duplicity on a Debian Linux computer using apt-get with the following command as superuser:

```
# apt-get install duplicity
```

SSH DSA Key Authentication
Once Duplicity was installed, I created a DSA key pair and set up SSH DSA key authentication to provide a means of using SSH without having to enter a password. Some people implement this by creating an SSH key without a password. This is extremely dangerous, because if people obtain the key, they instantly have the same access that the original key owner had. Using a password-protected key requires people who get the key also to have the key's password before they can gain access. To create an SSH key pair and set up SSH DSA key authentication, I ran the following command sequence on the client machine:

```
$ ssh-keygen -t dsa
$ scp ~/.ssh/id_dsa.pub <username>@<server>:
$ ssh <username>@<server>
$ cat id_dsa.pub >> ~/.ssh/authorized_keys2
$ exit
```

The first command creates the DSA key pair. The second command copies the previously generated public key to the backup server. The third command starts a remote shell on the backup server. The fourth command appends the public key to the list of authorized keys, enabling key authentication between the client machine and the backup server. The fifth and final command exits the remote shell.

GnuPG Key Setup
After setting up SSH key authentication, I created a GnuPG key that Duplicity would use to sign and encrypt the backups. I created a key as my normal user on the client machine. Having the GnuPG key associated with a normal user account prevents backing up the entire filesystem. If I decided at some point that I wanted to back up the entire filesystem, I simply would create a GnuPG key as the root user on the client machine. To generate a GPG key, I used the following command:

```
$ gpg --gen-key
```

Keychain
Once both the GnuPG and SSH keys were created, the first thing I did was make a CD containing copies of both my SSH and GnuPG keys. Then I installed and set up Keychain. Keychain is an application that manages long-lived instances of ssh-agent and gpg-agent to provide a mechanism that eliminates the need for password entry for every command that requires either the GnuPG or SSH keys. On a Debian client machine, I first had to install the keychain and ssh-askpass packages. Then I edited the /etc/X11/Xsession.options file and commented out the use-ssh-agent line so that the ssh-agent was not started every time I logged in with an Xsession. Then I added the following lines to my .bashrc file to start up Keychain properly:

```
/usr/bin/keychain ~/.ssh/id_dsa 2> /dev/null
source ~/.keychain/`hostname`-sh
```

After that, I added an xterm instantiation to my gnome-session so that an xterm in turn starts an instance of bash, which reads in the .bashrc file and runs Keychain. When Keychain is executed, it checks to see whether the key is already cached; if it is not, it prompts me once for my key passwords every time I start my computer and log in.

Using Duplicity
Once Keychain was installed and configured, I was able to make unattended backups of directories simply by configuring cron to execute Duplicity. I backed up my home directory with the following command:

```
$ duplicity --encrypt-key AA43E426 \
--sign-key AA43E426 /home/username \
scp://user@backup_serv/backup/home
```

After backing up my home directory, I verified the backup with the following command:

```
$ duplicity --verify --encrypt-key AA43E426 \
--sign-key AA43E426 \
```

```
scp://user@backup_serv/backup/home \
/home/username
```

Suppose that I accidentally removed my home directory on my client machine. To recover it from the backup server, I would use the following command:

```
$ duplicity --encrypt-key AA43E426 \
--sign-key AA43E426 \
scp://user@backup_serv/backup/home \
/home/username
```

However, my GnuPG and SSH keys are normally stored in my home directory. Without the keys I cannot recover my backups. Hence, I first recovered my GPG and SSH keys from the CD on which I previously saved my keys.

This solution also provides the capability of cleaning up files on the backup server for a specified date and time. Given this capability, I also added the following command to my cron tab to remove any backups more than two months old:

```
$ duplicity --remove-older-than 2M \
--encrypt-key AA43E426 --sign-key AA43E426 \
scp://user@backup_serv/backup/home \
```

```
/home/username
```

This command conserves disk space, but it limits how far back I can recover data.

Conclusion

This solution has worked very well for me. It provides the key functionality that I need and meets all of my requirements. It is not perfect, however. Duplicity currently does not support hard-links; it treats them as individual files. Hence, in a backup recovery that contains hard-links, individual files are produced rather than one file with associated hard-links.

Despite Duplicity's lack of support for hard-links, this is still my choice of backup solution. It seems that development of Duplicity has recently picked up, and maybe this phase of development will add hard-link support. Maybe I will find the time to add this support myself. Either way, this provides an unattended, encrypted, redundant network backup solution that takes very little money or effort to set up.■

Andrew J. De Ponte is a security professional and avid software developer. He has worked with a variety of UNIX-based distributions since 1997 and believes the key to success in general is the balance of design and productivity. He awaits comments and questions at cyphactor@socall.rr.com.

Ajax Timelines and the Semantic Web

Explore anything that has a time component with a little Timeline Ajax code. BEN MARTIN

Timeline uses Asynchronous JavaScript and XML (Ajax) to provide a nice interface for browsing information that has a time component. The Timeline Web site describes Timeline as "...Google Maps for time-based information".

Timeline lets you view points and durations of time in an intuitive manner. I refer to these as time events or just events when the context is clear. Many bands at different granularities—hour, day, month, year and so on—can show you how events relate to each other. You can use the mouse to drag around the display, or double-click on the Timeline to center at that time. All events can have click bubbles showing a little HTML with links and images.

Using Timeline itself requires no software installation on the client or Web server. Although there are no requirements for installing Timeline, while developing Timeline Web sites, you can improve reload speed by installing Timeline on the local machine. To do this, check out a copy of Timeline from Subversion, and change the script path in your Timeline HTML files to point to your local copy.

Listing 1. Get Timeline from Subversion for quicker reloads.

```
$ svn checkout \
http://simile.mit.edu/repository/timeline/
```

Generating a Timeline

Timelines are normally generated in the onLoad() JavaScript function of the HTML page body. An HTML div element is defined where the Timeline itself is to be generated. Call Timeline.create() in the onLoad() JavaScript function, passing the ID of this div element and the information to use for the Timeline.

Many day, week, month and year sliders can be created using the Timeline.createBandInfo(), which selects the time unit and screen size relative to the entire Timeline that each band will consume. The Timeline is populated with time event data from an XML file using Timeline.loadXML(). An update function also should be called in onResize() to allow the Timeline to redraw itself.

An HTML file showing a Timeline is provided in Listing 2. First, we include the timeline-api JavaScript file directly from mit.edu. The bulk of the work is done in the onLoad() function that generates two bands: one showing days and the other months. The two bands are passed as an array into Timeline.create(), along with the HTML ID of the div tag where we want this Timeline to be. The bands are connected to an event source object, through which we then load our Timeline XML file. The syncWith setting makes sure that when you drag one time band the other will follow. Our OnResize() function makes sure that Timeline.layout() is called to update our Timeline. The rest of the HTML file simply defines a few other elements and a div tag where we want our Timeline to be created.
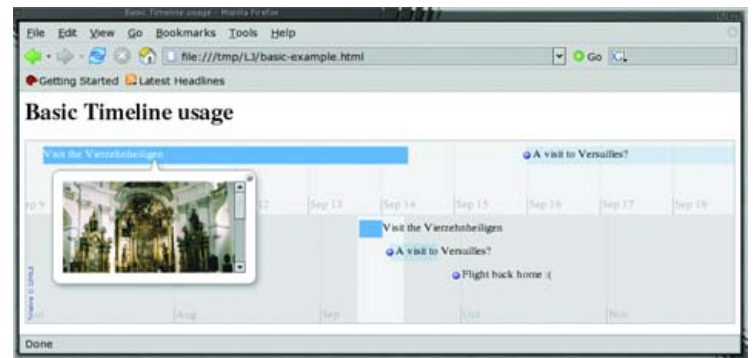

Figure 1. A Basic Timeline in Firefox

The XML file containing the dates is shown in Listing 3. This contains two types of durations: one we are sure of and one that is just a rough window of time. Because the XML file does not contain isDuration="true" for the Versailles event, it will be shown differently on the Timeline. The final event is a fixed single point in time when our flight leaves.

Events can have links, images and an HTML content associated with them. The screenshot in Figure 1 shows how this example is rendered by Firefox. Here, I have clicked on the Vierzehnheiligen event to show its image, and below that will be the HTML associated with this event.

A band on the Timeline can be nonlinear. For example, this band could display days as its default unit until it hits a hectic period, at which point it shows hour units for a three-day period before reverting to days as its default unit. This is done using Hot Zones, which are created by calling Timeline.createHotZoneBandInfo() instead of Timeline.createBandInfo() and passing an array of band information.

Theme Your Timeline

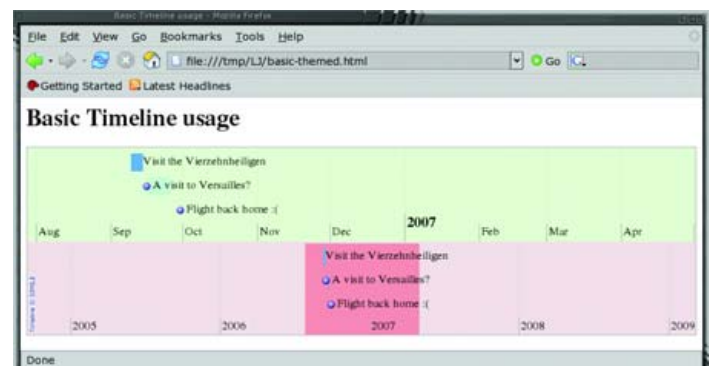The default timeline theme is low contrast grey on grey for the font

Figure 2. A Themed Timeline with Bands Changed to Months and Years

Listing 2. HTML Showing a Basic Timeline

```html
<html>
   <head>
     <title>Basic Timeline usage</title>
     <script src=
"http://simile.mit.edu/timeline/api/timeline-api.js"
       type="text/javascript">
     </script>

     <script>
function onLoad() {
   var eventSource =
     new Timeline.DefaultEventSource();

   var bandInfos = [
     Timeline.createBandInfo({
         eventSource:     eventSource,
         date:            "Sep 14 2006 00:00:00 GMT",
         width:           "40%",
         intervalUnit:    Timeline.DateTime.DAY,
         intervalPixels:  100
     }),
     Timeline.createBandInfo({
         eventSource:     eventSource,
         date:            "Sep 14 2006 00:00:00 GMT",
         width:           "60%",
         intervalUnit:    Timeline.DateTime.MONTH,
         intervalPixels:  200
     })
   ];
   bandInfos[1].syncWith = 0;
   bandInfos[1].highlight = true;
   tl = Timeline.create(
     document.getElementById("my-timeline"),
     bandInfos);
   Timeline.loadXML("basic-example.xml",
     function(xml, url) {
       eventSource.loadXML(xml, url); });
}

var resizeTimerID = null;
function onResize() {
   if (resizeTimerID == null) {
     resizeTimerID = window.setTimeout(function() {
       resizeTimerID = null;
       tl.layout();
     }, 500);
   }
}
     </script>
   </head>

   <body onload="onLoad();" onresize="onResize();">
     <h1>Basic Timeline usage</h1>

     <div id="my-timeline"
       style="height: 250px; border: 1px solid #aaa">
     </div>

   </body>
</html>
```

```
<data>
 <event
   start="Sep 9 2006 09:00:00 GMT"
   end="Sep 14 2006 09:00:00 GMT"
   isDuration="true"
   title="Visit the Vierzehnheiligen"
   image="vierzehnheiligen-thumb.jpg"
 >
 Visit this impressive church in Germany. More
 information can be found at its
 &lt;a href=
 "http://en.wikipedia.org/wiki/Vierzehnheiligen"
 &gt;Wikipedia page&lt;/a&gt;.
 </event>

 <event
   start="Sep 16 2006 00:00:00 GMT"
   end="Sep 26 2006 00:00:00 GMT"
   title="A visit to Versailles?"
   image="versailles-thumb.jpg"
   link="http://www.chateauversailles.fr/en/"
 >
 Sometime in this window I should
 get out to Versailles.
 </event>

 <event
   start="Sep 30 2006 00:00:00 GMT"
   title="Flight back home :("
 >
  The joy has to end sometime :(
 </event>
</data>
```

```
<script src=
"http://simile.mit.edu/timeline/api/timeline-api.js"
type="text/javascript"></script>

<style type="text/css">
.timeline-ether-marker-bottom {
 width:        5em;
 height:       1.5em;
 border-left:  1px solid #aaa;
 padding-left: 2px;
 color:        black;
}
.timeline-ether-marker-bottom-emphasized {
 width:        5em;
 height:       2em;
 border-left:  1px solid #aaa;
 padding-left: 2px;
 color:        black;
 font-size:    120%;
 font-weight:  bold;
}
</style>

<script>
function onLoad() {
var eventSource = new Timeline.DefaultEventSource();

 var theme = Timeline.ClassicTheme.create();
 theme.ether.backgroundColors[0] = '#DFD';
 theme.ether.backgroundColors[1] = '#EDD';
 theme.ether.highlightColor = '#E00';
 theme.ether.highlightOpacity = '30';
 theme.event.bubble.width = 520;
 theme.event.bubble.height = 120;

 var bandInfos = [
  Timeline.createBandInfo({
   ...
   intervalPixels: 100,
   theme:          theme
  }),
  Timeline.createBandInfo({
   ...
   intervalPixels: 100,
   theme:          theme
  })
 };
...
```

and background with blue highlights for events. This can be customized using a combination of JavaScript and Cascading Style Sheets (CSS), depending on what you want to change. To change the background colors and some of the time bands, you can create an instance of the default theme JavaScript object, make modifications to that object and then pass it to Timeline.createBandInfo(). The font colors are set using CSS.

Listing 4 shows the changes needed for the previous HTML file to modify the band colors and font information. After including the timeline-api, we override two of the CSS classes to change the font color and enlarge the major date markers. The band colors and click bubble size are properties of the theme object. This modified theme object is then passed as a parameter to the Timeline.createBandInfo() function when creating the bands. The result is shown in Figure 2.

Showing syslog on a Timeline

syslog is a great source of highly time-related information. Perl makes it easy to convert syslog files into the XML format required by Timeline. In this example, I convert from the format used by /var/log/messages in Fedora Core 5 into a Timeline XML file, shown in Listing 5. The main complication is that, by default, the year is

not included in the date and time specification in the syslog file. This makes the regular expression to split the input more complicated, as we want to get the date and time separately, so we can insert the year between them in the output.

Making the Timeline higher and including three bands makes jumping around in the logs easier, as shown in Listing 6.

```perl
#!/usr/bin/perl
use XML::Writer;
my $writer = XML::Writer->new();
$writer->xmlDecl();
$writer->startTag('data');

$thisyear=((localtime)[5]+1900);

while( <> ) {
  # The if() is all one line.
  if( /([a-zA-Z ]+[0-9]+) ([0-9]+
:[0-9]+:[0-9]+) ([^:]+):(.*)/)
  {
    $date=$1; $time=$2;
    $src=$3;
    $msg=$4;
    $writer->startTag(
      'event',
      'start' => "$date $thisyear $time",
      'title' => $src
    );
    $writer->characters( $msg );
    $writer->endTag('event');
  }
}

$writer->endTag('data');
$writer->end();
```

Timelines Meet the Semantic Web

Generating and updating Timelines becomes simpler when com-
bined with some Semantic Web technologies. The two main ones
of use here are an RDF store supporting the SPARQL query lan-
guage and an XSLT engine to generate JavaScript Object Notation
(JSON) files.

Using RDF lets you maintain a single store of information and
choose whatever data is of interest using queries. Also, with RDF you
can merge information from multiple sources easily into a single
Timeline. For example, it might be handy to see the modification times
of files along with syslog events on a single Timeline.

Using JSON allows the JavaScript for a page to access time events
as normal JavaScript objects. So, you can, for example, center the page
by default on the oldest, newest or a named event from the JSON
data. This is very handy if the time events change, as the JavaScript will
still center the page correctly without modifying the HTML file to point
to the desired time explicitly.

RDF is the Resource Description Framework that is the lowest layer
of the Semantic Web. Everything is described in terms of triples in
RDF—for example, Ben, programs, C++.

Unlike the previous example, triples in RDF are constructed
using Uniform Resource Identifiers (URIs) and Objects. A URI is very
similar to a URL. The main difference is that URIs are not expected
to resolve to something that you can browse on the Net but are
intended only to identify something uniquely. Many people use
http:// URLs as URIs. The previous example would more likely be
expressed in RDF as shown in Listing 7. Normally, people would

not be identified uniquely by their first name only.

The additional verbosity of URIs is not really a concern, because most
things dealing in RDF will let you define namespaces similar to XML. For
example, setting kvo to expand to http://www.kvocentral.org/rdf/
would shorten the first part of the example triple to kvo:person/Ben.
The three parts of a triple are referred to as the Subject, Predicate
and Object. It is convenient to think of the subject as defining
the thing you are describing, the predicate as defining what
part of the subject you are describing and the object as the
description itself.

SPARQL is a query language for RDF data. SPARQL borrows
some notation from SQL. Variables in SPARQL are defined using
?varname. When a variable appears more than once in the where
clause it must have the same value for each appearance. For exam-
ple, the SPARQL query in Listing 8 can return multiple ?x, ?name
pairs, but each ?x returned will have a location of Sydney. The
optional clause means that if ?x happens to have a digital longi-
tude associated with it, that will be returned as well.

Some of the following code is from or based on the ESW
SparqlTimeline page (see the on-line Resources), in particular, the
sparql2timeline.xsl file.

I attempted to use the Redland and Rasqal combination for

Listing 6. Three Bands to Best Move around Daily Events

```
var bandInfos = [
  Timeline.createBandInfo({
      eventSource:     eventSource,
      date:            "Sep  7 2006 00:00:00",
      width:           "40%",
      intervalUnit:    Timeline.DateTime.MINUTE,
      intervalPixels: 100
  }),
  Timeline.createBandInfo({
      eventSource:     eventSource,
      date:            "Sep  7 2006 00:00:00",
      width:           "30%",
      intervalUnit:    Timeline.DateTime.HOUR,
      intervalPixels: 200
  }),
  Timeline.createBandInfo({
      eventSource:     eventSource,
      date:            "Sep  7 2006 00:00:00",
      width:           "30%",
      intervalUnit:    Timeline.DateTime.DAY,
      intervalPixels: 200
  })
];
bandInfos[1].syncWith = 0;
bandInfos[1].highlight = true;
bandInfos[2].syncWith = 0;
bandInfos[2].highlight = true;
```

Listing 7. A Simple RDF Triple

```
http://www.kvocentral.org/rdf/person/Ben,
http://www.kvocentral.org/rdf/activity/programs,
http://www.kvocentral.org/rdf/programming-language/C++
```

Listing 8. A SPARQL-like Query for Blogs

```
SELECT ?x ?name ?dlat
WHERE {
  ?x has-name ?name .
  ?x has-location "Sydney" .
  OPTIONAL { ?x geospat:longitude ?dlat } .
}
```

RDF+SPARQL but ran into troubles with SPARQL processing. Redland is still developing its SPARQL query implementation. I then moved to using Jena for RDF processing. The Jena Project is well known for being a feature-rich and robust RDF library. For more information on playing with RSS blog feeds with Jena, see my article "Creating a Planet Me Blog Aggregator", which appeared in the April 2006 issue of *Linux Journal*.

Jena is written in Java, and thus, you'll need a JRE. Jena itself

Listing 9. Setting Up Jena 2.4

```
$ cd ~
$ unzip Jena-2.4.zip
$ edit ~/.bashrc
# append a handy classpath setup
JenaSetup() {
    for if in ~/Jena-2.4/lib/*.jar; do
        export CLASSPATH=$CLASSPATH:$if;
    done
}
$ . ~/.bashrc
$ JenaSetup
```

is easy to install; simply unzip it somewhere and add its jar files to your CLASSPATH environment variable. For a bash shell, this is shown in Listing 9.

Blogs and Timelines

Individual Blogs and the Planet Blog aggregator normally offer RSS 1.0 feeds. The shell commands to show a Planet on a Timeline are shown in Listing 10. The planet GNOME RSS feed URL could have been included directly into the Jena SPARQL command. Keeping it separate allows you to archive your blogs or combine many blogs into a single

Listing 10. Generate a Timeline for Planet GNOME.

```
wget -O planet-gnome.xml \
  http://planet.gnome.org/rss10.xml
java jena.sparql \
  --data planet-gnome.xml \
  --query rss.rq --results xml \
  >|planet.xml
xsltproc sparql2timeline.xsl planet.xml \
  | tr '\n' ' '  >|planet.json
```

Listing 11. SPARQL Query for Blogs

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rss:  <http://purl.org/rss/1.0/>
PREFIX rssc: <http://purl.org/rss/1.0/modules/content/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>

SELECT distinct ?title ?link ?date ?description
WHERE {
  ?x rdf:type rss:item .
  ?x rss:link ?link .
  ?x rss:title ?title .
  ?x dc:date ?date .
  ?x rssc:encoded ?description
}
ORDER BY DESC(?date)
```

```
var moveRightOffetInHours = 4;
var gmtd = new Date();
var ms = gmtd.getTime()
    + (gmtd.getTimezoneOffset() * 60000)
    - moveRightOffetInHours *3600000;
var d =  new Date(ms);

var bandInfos = [
  Timeline.createBandInfo({
   eventSource:    eventSource,
   date:           d,
...
```

```
function onLoad() {
 ...
 tl.loadJSON("planet.json", function(json, url) {

  if( json.events.length ) {
    var td = Timeline.DateTime.parseIso8601DateTime(
                json.events[0].start);
    tl._bands[0]._ether.setDate( td );
    tl._bands[1]._ether.setDate( td );
  }
  eventSource.loadJSON(json, url);
  tl.layout();
 });
...
```

RDF file for querying.

The final command converts the XML file containing the results of the SPARQL query into a JSON file. Because the XSLT outputs plain text, there could be many newlines in places where a browser does not like them. The main offender here is newlines inside of a blog's HTML content. Because the output is JSON, the blog entry's content has to be contained in a JavaScript string declaration. Having a JavaScript string declaration extend over multiple lines by just ending each line with a newline will confuse many browsers. A simple remedy is to use the tr(1) utility to replace newlines with harmless space characters.

The SPARQL query itself is shown in Listing 11. Each Blog post is an RSS item. The first line in the WHERE clause restricts results to news items (blog posts). The subsequent lines select the information about each blog post we are interested in for the SELECT clause.

There are a few changes that can be made to the driving HTML file to make viewing the results of blog queries simpler. The first option is to set the default target date to be a few hours before the current time. We shift a few hours back from the current time because the finest granularity time band on the Timeline is hours. This places the most recent posting to the right of the Timeline instead of in the center. The fragment that needs to change revolves around the bandInfos declaration, as shown in Listing 12.

One major advantage of using JSON to keep the time events is that they are accessible as a JavaScript array object. To support viewing the output of arbitrary queries, it is convenient to have the JavaScript in the HTML center the display on the most recent time event on the Timeline. Although getting at the date is quite easy, unfortunately, we have to poke around in some private areas of the Timeline API to do this, which requires a call to layout() in order for the Timeline to update its labels to reflect the time change. This is shown in Listing 13. The Timeline is shown in Figure 3.

Timelines and Evolution

Evolution supports time events on a calendar display. Because Timeline is lightweight and completely browser-based it also can be used on many pocket-sized devices. It might be handy to export your Evolution calendar information into a Timeline file to take on the road with you.

I'm using Evolution version 2.6.3; later versions may have fixed some of the following issues.

To export your Evolution calendar, right-click on On This Computer/Personal, and choose Save to disk. There are two ways to arrive at an RDF result: directly exporting as RDF and exporting to iCalendar format and converting that to RDF later.
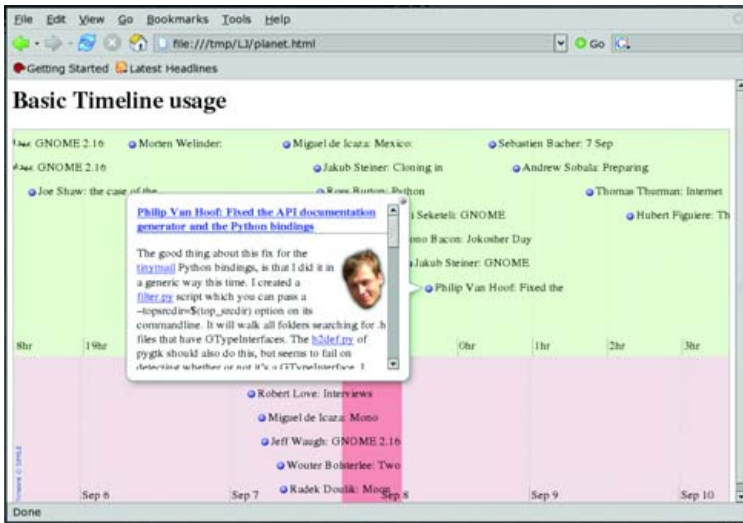
Figure 3. Planet GNOME on a Timeline

Listing 14. Namespace the about tags using stdout.

```
$ sed 's/<Vevent about=/<Vevent rdf:about=/g' \
    mycal.rdf >|mycal-clean.rdf
$ java jena.sparql \
    --data mycal-clean.rdf \
    --query  evolution-to-timeline.rq \
    --results xml >| evolution.xml
$ xsltproc sparql2timeline.xsl evolution.xml \
    | tr '\n' ' '  >| evolution.json
```

Listing 15. SPARQL Query for Evolution Calendars (evolution-to-timeline.rq)

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX ical: <http://www.w3.org/2002/12/cal/ical#>

SELECT distinct ?uid ?title ?date ?enddate ?description
WHERE {

  ?x ical:uid      ?uid .
  ?x ical:summary ?title .
  ?x ical:dtstart ?date .
  ?x ical:dtend    ?enddate .
  ?x ical:description ?description

}
ORDER BY DESC(?date)
```

Listing 16. A Slight Modification to sparql2timeline.xsl to Translate Evolution Calendar Data to JSON

```
...
<xsl:variable name="date">
   ...
</xsl:variable>
<xsl:variable name="enddate">
  <xsl:call-template name="escape">
  <xsl:with-param name="text"
 select="res:binding[@name='enddate']/res:literal"/>
  </xsl:call-template>
</xsl:variable>
...
</xsl:variable>
 {'start': '<xsl:value-of select="$date" />',
 'end': '<xsl:value-of select="$enddate" />',
 'title': '<xsl:value-of select="$title" />',
...
```

The major problem in exporting events from Evolution is exporting recurring events. In a direct RDF export, only the first instance of a recurring event will be present in the result. In an iCalendar export, you will have an RRULE tag for the event that contains the information about the recurrence. Unfortunately, the w3.org's fromIcal.py (which converts iCalendar to RDF) is confused by this RRULE.

When exporting directly to RDF, you might encounter the use of the deprecated RDF feature of not explicitly namespacing the rdf:about tag. Jena provides warnings about the implicit name-spacing, and unfortunately, they are on stdout instead of stderr. We want stdout to contain only a valid RDF document from our query. The little bit of sed at the top of the commands in Listing 14 will properly namespace the about tag and thus silence Jena. The mycal.rdf is exported from Evolution.

The SPARQL query shown in Listing 15 uses the same names in the SELECT clause as the blog query SPARQL. Because many calendar events will have a duration, I have added the enddate to the SELECT clause.

By using the same names in the SELECT clause, we can use the

same sparql2timeline.xsl file with a few minor modifications to produce our JSON data for the Timeline. The differences to sparql2timeline.xsl are shown in Listing 16.

The driving HTML file can simply be a copy of the planet.html, modified to include evolution.json instead of planet.json.

Timelines from Your Files

Filesystem information could be written directly to an XML Timeline file as was done in the syslog section above. Generating RDF from

Listing 17. Installing Redland Perl Bindings

```
tar xzvf redland-bindings-1.0.4.1.tar.gz
cd redland-bindings-1.0.4.1
./configure  --with-perl
cd ./perl
make
make install
```

Listing 18. Glue to Transform find Results to RDF

```perl
#!/usr/bin/perl

use POSIX;
use File::Basename;
use RDF::Redland;

$storage=new RDF::Redland::Storage(
 "hashes", "test", "new='yes',hash-type='memory'");
$model=new RDF::Redland::Model($storage, "");

$rdfns = "http://witme.sf.net/rdf/filesystem/";

$/="\0";
while( <> ) {
 $url=$_;
 # remove pesky null char at end-of-string
 chomp($url);
 ($dev, $ino, $mode, $nlink, $uid,$gid,$rdev,
   $size,$atime,$mtime,$ctime) = lstat($_);

 $model->add(
  new RDF::Redland::URI( "${rdfns}${ino}" ),
  new RDF::Redland::URI( "${rdfns}inode" ),
  new RDF::Redland::LiteralNode( "$ino" ) );
 $model->add(
  new RDF::Redland::URI( "${rdfns}${ino}" ),
  new RDF::Redland::URI( "${rdfns}url" ),
  new RDF::Redland::LiteralNode( "$url" ) );
 $model->add(
  new RDF::Redland::URI( "${rdfns}${ino}" ),
  new RDF::Redland::URI( "${rdfns}basename" ),
  new RDF::Redland::LiteralNode(basename("$url")));

$model->add(
 new RDF::Redland::URI( "${rdfns}${ino}" ),
 new RDF::Redland::URI( "${rdfns}title" ),
 new RDF::Redland::LiteralNode(
   substr basename("$url"), 0, 25 ) );
$model->add(
 new RDF::Redland::URI( "${rdfns}${ino}" ),
 new RDF::Redland::URI( "${rdfns}mtime" ),
 new RDF::Redland::LiteralNode( strftime(
   "%Y-%m-%d %H:%M:%S", localtime($mtime)) ) );
$model->add(
 new RDF::Redland::URINode( "${rdfns}${ino}" ),
 new RDF::Redland::URINode( "${rdfns}size" ),
 new RDF::Redland::LiteralNode( "$size" ) );
$model->add(
 new RDF::Redland::URI( "${rdfns}${ino}" ),
 new RDF::Redland::URI( "${rdfns}content" ),
 new RDF::Redland::LiteralNode( "$url<br>" ) );

$desc = "<a href=\"${url}\">$url</a><br></br>"
. "<iframe src=\"${url}\" "
. "width=\"95%\" height=\"75%\"></iframe>";
$model->add(
 new RDF::Redland::URI( "${rdfns}${ino}" ),
 new RDF::Redland::URI( "${rdfns}description" ),
 RDF::Redland::Node->new_xml_literal( $desc ) );
}

$model->sync();
print $model->to_string() , "\n";
```

Using RDF and SPARQL can be a great advantage when creating Timelines for new data sources.

filesystem searches allows you to use different SPARQL queries at a later time to refine your Timeline.

The results of the find command can be turned into RDF quickly with Perl and Redland. The Redland library follows the ./configure; make; make install; three-step process. Installing the Perl bindings requires that you configure the bindings package enabling the Perl wrapper, as shown in Listing 17.

The script shown in Listing 18 transforms null-separated output from a find invocation into an RDF file. The inode for each file forms the subject in the output RDF. The metadata for each file is associated with its inode subject. A few things of note: I create a shortened version of basename to serve as the label on the Timeline, and the mtime is converted into a string representation in RDF. Currently, Timeline doesn't display any label for time event labels that are too long. Also, the description will show the file's contents in the click bubble for each event.

The SPARQL query is shown in Listing 19. The sparql2timeline.xsl

can be reused from any of the above examples. The commands also are very similar, as shown in Listing 20. The evolution.html can be copied to filesystem.html and modified to include filesystem.json, and we have a new Timeline.

Conclusion

Using RDF and SPARQL can be a great advantage when creating Timelines for new data sources. The sparql2timeline.xsl file can be reused to convert SPARQL query results to JSON. The two main things required are getting the data into RDF and the SPARQL query itself. I've touched on only some possibilities of SPARQL in this article. With SPARQL, it's easy to ensure that a value in the results matches a regular expression or has some other property, such as being between two dates. Results can come from multiple data sources using the UNION keyword. For example, it is easy to combine any of the above SPARQL queries into a single query to show multiple types of time events on a single Timeline.∎

Resources for this article: www.linuxjournal.com/article/9463.

Ben Martin has been working on filesystems for more than ten years. He is currently working toward a PhD at the University of Wollongong, Australia, combining Semantic Filesystems with Formal Concept Analysis to improve human–filesystem interaction.

Controlling Spam with SpamAssassin

How to set up SpamAssassin and teach it to recognize spam. COLIN MCGREGOR

The people who produce unsolicited commercial e-mail (UCE), or spam, are the big thieves of the Information Age, spewing out messages for pharmaceuticals, timepieces, fast money and fast women. Large chunks of bandwidth that we have to pay for is eaten up by these crooks. After getting these messages, we have to waste time going through our inboxes and deleting the garbage. Further, unlike magazines, newspapers, commercial radio and television, where the advertisements reduce the cost or make the content free, spam gives nothing back to us as readers or viewers.

Although we cannot stop spam, some tools exist to make spam easier to deal with. One such tool is SpamAssassin, which looks at each incoming e-mail message and rates the probability that the e-mail is spam. Messages that are given a high probability of being spam get flagged as such, and other programs, such as Evolution, KMail or Procmail, can deal painlessly with the flagged e-mail.

SpamAssassin works by going through e-mail messages and looking for things that are associated with spam or non-spam e-mail, which add or subtract points from an e-mail's score. So, for example, the word Viagra, and close misspellings of Viagra (as they are used in many pharmaceutical spam messages), adds to the total score. On the other hand, a valid Sender Policy Framework (SPF) record in the e-mail, which shows that the sender location was not forged, subtracts from the score. By default, any message that gets a total score of five or more is assumed to be spam.

One problem with the above calculations is that it is a fair bit of work for your computer, so if your machine is currently straining under the workload it has, or if you deal with a lot of e-mail, you may want to look at a hardware upgrade (faster CPU chip and/or more memory) before starting up SpamAssassin.

A number of Linux distributions include SpamAssassin by default. If yours isn't one of them, it should be very simple to add. If you have a Debian-based distribution, it should be as simple as starting up a terminal window and typing:

```
sudo apt-get install spamassassin
```

Once installed, you can start tweaking SpamAssassin's settings. SpamAssassin's configuration file can be found at ~/.spamassassin/user_prefs. The first setting is required_score:

```
required_score          5
```

SpamAssassin is not perfect, no matter how you set things. There will be some spam e-mail allowed through, and some valid e-mail will be classed as spam. The goal with the configuration process is to make sure this happens as seldom as possible. The score of five is an excellent compromise for most people. But, if you find yourself getting a lot of spam coming through as non-spam, even after taking the configuration steps noted below, you may want to lower that number to a four or three (or possibly even lower). If, on the other hand, you find after configuration you have a lot of real e-mail identified as spam, you might want to raise the required_score.

There are some people that you always want to hear from, or at least, always want their e-mail to come through, such as coworkers and family members. There also are people that you never want to hear from again, such as annoying exes. SpamAssassin deals with these situations by having a whitelist and blacklist. An e-mail from someone on the whitelist gets 100 subtracted from the score; anyone on the blacklist gets 100 added to the score. To add someone to your white/blacklist, you need to add something like the following to user_prefs:

```
whitelist_from          niceperson@somedomain.somewhere
blacklist_from          nastyperson@somedomain.somewhere
```

Some people have specific reasons why they would want particular spam tests changed. For example, people working at a jewelry store, or watch collectors, might want to allow messages where the word Rolex has been emphasized, accepting that doing so also will increase the amount of replica-watch-related spam they will see. There is a list of SpamAssassin tests at **spamassassin.apache.org/tests.html**. For example, to change the score that an e-mail message gets when the word Rolex has been emphasized, reducing the chances that such a message would be tagged as spam, put the following line in user_prefs:

```
score EM_ROLEX 0
```

If too many legitimate Rolex-brand watch-related e-mail messages are still being tagged as spam, the above could be changed to a negative number.

By default, SpamAssassin assumes e-mail in a number of Asian languages, most notably, but not exclusively Chinese, Japanese and Korean, are probably spam. This is a problem if you use one of those languages. To allow Asian languages, you need to uncomment some lines by removing the # character at the start of the last four lines of user_prefs.

Now, let's further refine SpamAssassin's taste. My first run-through with SpamAssassin was a disappointment. Out of some 2,200 spam messages, only about 10% were correctly identified as spam. Fortunately, with SpamAssassin there is a utility program called sa-learn that will "teach" SpamAssassin what you consider to be spam and ham (non-spam). This process greatly improves SpamAssassin's ability to identify spam messages correctly. The trick here is to create folders, one filled with spam and another filled with the sort of material you want to keep, and then feed each folder into sa-learn. Using the Evolution e-mail program, I created a folder called BULK, and then I manually placed all the spam messages into that folder. Next, I ran the sa-learn program with the following command:

```
sa-learn --mbox --spam ~/.evolution/mail/local/BULK
```

Evolution stores all its e-mail in the mbox mail format, thus the --mbox option in the command above. The command for the non-spam messages, which I keep in the Inbox folder, is:

```
sa-learn --mbox --ham ~/.evolution/mail/local/Inbox
```

The learning system SpamAssassin uses starts to become good at around 1,000 spam and 1,000 ham messages. With a semi-exception, the

system doesn't improve noticeably until after seeing more than 5,000 e-mail messages. The semi-exception relates to the fact that spam is a moving target. Some spammers are always looking for better ways to get around filter programs, changing their spam as they go. What this means is that you need to re-train SpamAssassin periodically with new spam and new ham. How often depends on your situation, but basically you need to re-train whenever you see a noticeable increase in the amount of spam getting past SpamAssassin. Still, with training, it is very possible to reach spam-detection accuracy rates of more than 99%.

Remember that SpamAssassin remembers what e-mail it has seen before, so although some people may be tempted to run the same 1,000 e-mail messages through sa-learn five times, all this will do is waste time.

Let's see how SpamAssassin, actually rates a sample e-mail. For a test, I created a simple text file, testmail.txt with the following:

```
From: MyUserID@SomeDomain.Somewhere
To: aliceithink@somedomain.somewhere
Date: Sat, 2 Dec 2006 13:34:50 -0400 (EDT)
Subject: Back from vacation

Alice, I am back from vacation, anything important
happen when I was away?

Colin McGregor
```

Then, I ran SpamAssassin as a test with the following command:

```
spamassassin -t testmail.txt
```

I received an output like the following:

```
From: MyUserID@SomeDomain.Somewhere
To: aliceithink@somedomain.somewhere
Date: Sat, 2 Dec 2006 13:34:50 -0400 (EDT)
Subject: Back from vacation
X-Spam-Checker-Version: SpamAssassin 3.0.3
(2005-04-27) on diamond
X-Spam-Level:
X-Spam-Status: No, score=-5.9 required=5.0
tests=ALL_TRUSTED,BAYES_00,
        NO_REAL_NAME autolearn=ham version=3.0.3
```

```
Alice, I am back from vacation, anything important
happen when I was away?

Colin McGregor
Spam detection software, running on the system
"diamond", has
identified this incoming email as possible spam. The
original message
has been attached to this so you can view it (if it
isn't spam) or label
similar future email. If you have any questions, see
the administrator of that system for details.

Content preview: Alice, I am back from vacation,
anything important
  happen when I was away? Colin McGregor [...]

Content analysis details:    (-5.9 points, 5.0
required)

 pts rule name          description
---- ---------------- --------------------------------
 0.0 NO_REAL_NAME     From: does not include a real name
-3.3 ALL_TRUSTED      Did not pass through any untrusted hosts
-2.6 BAYES_00         BODY: Bayesian spam probability is 0 to 1%
                        [score: 0.0000]
```

With a score of −5.9, SpamAssassin would not consider the above to be actual spam. By editing testmail.txt and repeating the above, you can see how SpamAssassin reacts to various sorts of keywords—in particular, terms commonly found in spam such as luxury brand-name watches, pharmaceutical products, financial service terms and/or various pornographic terms.

It isn't clear yet what the magic bullet will be to stop spam and regain the bandwidth spam steals from all of us—better technology, new laws or better enforcement of laws currently in place. Likely an end to spam will require a mixture of actions. In the meantime, SpamAssassin does make dealing with spam a less painful, but not pain-free experience.■

Colin McGregor works for a Toronto-area charity, does consulting on the side and has served as President of the Toronto Free-Net. He also is secretary for and occasional guest speaker at the Greater Toronto Area Linux User Group meetings.

EVOLUTION AND SPAMASSASSIN

The Evolution e-mail display program has a good filtering system for sorting out incoming e-mail, but it is a bit weak when it comes to identifying spam.
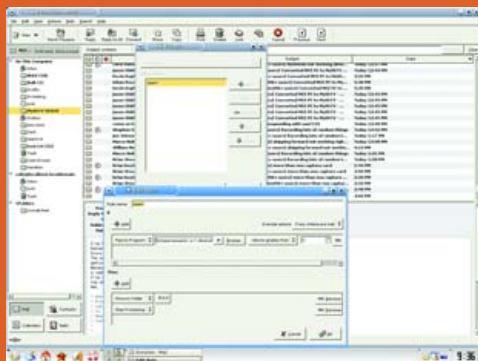

Figure 1. Creating and Editing Rules in Evolution

Fortunately, Evolution allows us to use external programs to help with sorting. From the main screen click on Tools→Filters. Then, click on +Add to create a new rule. You need a name for this rule, and spam should be just fine. Next, we want to send a copy of each e-mail to SpamAssassin and find out if SpamAssassin views the e-mail as spam; we do not care about the score SpamAssassin gives the e-mail, just a "yes" or "no". So, we Pipe to Program and then throw everything except the result code away. We do this with the instruction:

```
/usr/bin/spamassassin -e > /dev/null
```

If the above command returns a value of 0, it isn't spam. Anything more than 0 means we very likely have a spam and want it dropped into a separate folder. In the example shown in Figure 1, I am sending the e-mail into a folder labeled BULK. After

doing the above steps, we want the filter program to stop and wait for the next incoming e-mail.

As noted previously running sa-learn over the same e-mail twice is a waste of time. This raises another point when using Evolution and SpamAssassin, when you delete an e-mail message under Evolution, the program does not delete the e-mail from the ~/.evolution/mail/<file name> e-mail file, it just flags it for future removal. This way, if you make an error deleting an e-mail, you can get it back. To get rid of deleted e-mails completely under Evolution, you must click on Actions→Expunge. During your first days with SpamAssassin, when you might be running sa-learn several times over your BULK folder and your Incoming folder, you may not only want to delete e-mail previously seen by sa-learn, but also to Expunge it.

The GPLv2 vs. GPLv3 Debate

The question of whose freedom is more important. NICHOLAS PETRELEY
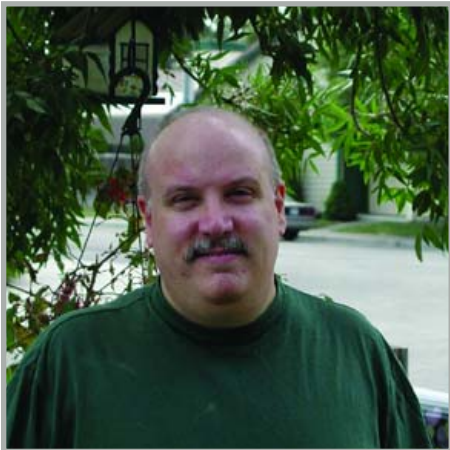
Nick Petreley, Editor in Chief

As of this writing, the GPLv2 vs. GPLv3 debate is still raging. For the record, GPLv3 doesn't bother me in the least as long as nobody is forced to use it. As for the debate itself, I'd love to say the debate boils down to one thing. But it doesn't. There's not enough room in this column to address all the issues involved; however, a few central issues deserve attention.

First, the debate presents a conflict between two different views of freedom. Linus Torvalds and other kernel developers want people to have the freedom to do whatever they want with Linux, hence their rejection of the current draft of GPLv3. This includes the freedom to produce a Linux-based device that implements some form of DRM, such as TiVo. I feel compelled to mention that Torvalds had made a good case that not all DRM is bad, but as you'll soon see, the argument surrounding DRM can be peripheral to the debate.

The FSF wants users to have the freedom to take the source code that was used to cre-

ate the binary that runs in any device, modify the code and run a binary of the modified version on the same device. Once again, the argument usually revolves around TiVo.

Forget TiVo for a moment. I would like to pose a different hypothetical scenario. I have two ulterior motives for doing so. The following example fits nicely with this issue's emphasis on embedded systems, and it circumvents the tendency to focus the controversy on DRM.

Think of a gadget manufacturer that wants to ship its gadgets with an operating system on ROM. The end user's freedom is not restricted by DRM. It is simply impossible to flash a ROM with the binary of a modified version of the source code.

One could argue that consumers are potentially harmed because they are stuck with any bugs in the software on ROM. Neither you nor the gadget provider has an easy way to update the software. Fair enough, but is that potential problem a good enough argument to draft a license to prevent this company from using free software?

It depends. The Volkswagen was originally designed with every conceivable cost-cutting measure to make it affordable to the working man. Some models lacked a gas gauge as late as 1959 or 1960 (I can't recall the exact date when the gauge was added). When you ran out of gas, you switched to a spare tank. If you weren't paying attention or overestimated how far you could go on that spare tank, you could run out of gas. That was quite a big inconvenience, but many consumers were happy with that trade-off because it was one reason they could afford to own a car.

Let's go back to that gadget with the software on ROM. Even if there is a bug in the software that is as annoying as running out of gas, many consumers may see it as an acceptable trade-off for being able

to own a device that was formerly out of their reach. The only people who are truly harmed by having the software on ROM are the tiny minority of hackers who want to run a modified version of the software on the gadget.

Someone will no doubt point out that not all users have to be hackers to get their gadgets modified. They may have hacker friends. But look around you. Of all the cell-phone users you see daily, how many do you think have ever had the thought enter their minds that they could get someone to modify the software on their phones? How many are brave enough to let them do it? What about people who use GPS navigation systems, DVD players, televisions, microwaves or other devices? What about devices that use Flash ROM, but do not provide a way for end users to update it?

So, you see, although the GPLv3 protects some end users, it is intellectually dishonest to say that, in all cases, the GPLv3 necessarily protects the freedoms of the vast majority of end users. It protects only a minority of hackers or brave, savvy users with hacker friends, and applies only to devices that provide ways to apply updates.

I'm sure this analogy has holes. No example or analogy is perfect. But I hope you can see that, regardless of which side gets your vote, both the GPLv2 and the GPLv3 can stomp on someone's freedom. So this debate is not simply about freedom. It is about whose freedom is more important.

Now let's get to the grand ulterior motive for this column. Do you see how much the debate can change once we stop focusing on DRM and fair use restrictions? I know what that tells me, but what does that tell you?■

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.

Rackspace – Managed Hosting Backed by Fanatical Support™

Fast servers, secure data centers and maximum bandwidth are all well and good. In fact, we invest a lot of money in them every year. But we believe hosting enterprise class web sites and web applications takes more than technology. It takes Fanatical Support.

Fanatical Support isn't a clever slogan, but the day to day reality our customers experience working with us. It's how we have reimagined customer service to bring unprecedented responsiveness and value to everything we do for our customers. It starts the first time you talk with us. And it never ends.

Contact us to see how Fanatical Support works for you.

1.888.571.8976 or visit **www.rackspace.com**

Thanks for honoring us with the
2005 Linux Journal Readers' Choice Award for
"Favorite Web-Hosting Service"

rackspace™
MANAGED HOSTING

The World's Fastest InfiniBand Switch

From a Company You've Trusted for 24 Years

Microway's **FasTree™** DDR InfiniBand switches run at 5GHz, twice as fast as the competition's SDR models. FasTree's non-blocking, flow-through architecture makes it possible to create 24 to 72 port modular fabrics which have lower latency than monolithic switches. They aggregate data modulo 24 instead of 12, improving nearest neighbor latency in fine grain problems and doubling the size of the largest three hop fat tree that can be built, from 288 to 576 ports. Larger fabrics can be created linking 576 port domains together.

▲ **72 Port FasTree™ Configuration**

Working with QLogic's InfiniPath InfiniBand Adapters, the number of hops required to move MPI messages between nodes is reduced, improving latency. The modular design makes them useful for SDR, DDR and future QDR InfiniBand fabrics, greatly extending their useful life. Please send email to fastree@microway.com to request our white paper entitled *Low Latency Modular Switches for InfiniBand.*

Harness the power of 16 Opteron™ cores and 128 GB in 4U

Microway's **QuadPuter®** includes four or eight AMD dual core Opteron™ processors, 1350 Watt redundant power supply, and up to 8 redundant, hot swap hard drives-all in 4U. Dual core enables users to increase computing capacity without increasing power requirements, thereby providing the best performance per watt. Constructed with stainless steel, QuadPuter's RuggedRack™ architecture is designed to keep the processors and memory running cool and efficiently. Hard drives are cooled with external air and are front-mounted along with the power supply for easy access and removal. The RuggedRack™ with an 8-way motherboard, 8 drives, and up to 128 GB of memory is an excellent platform for power- and memory-hungry SMP applications.

Call us first at 508-746-7341 for quotes on clusters and storage solutions. Find testimonials and a list of satisfied customers at microway.com.

◄ **QuadPuter® Navion™** with hot swap, redundant power and hard drives and four or eight dual core Opterons, offering the perfect balance between performance and density

QLOGIC®

AMD 64 Opteron™

Microway®
Technology you can count on℠

508.746.7341 microway.com