CouchDB | Caffeine | OpenWrt | Arduino | wview | OSWALD | Kdenlive

LINUX JOURNAL

Since 1994: The Original Magazine of the Linux Community
AUGUST 2010 | ISSUE 196 | www.linuxjournal.com

Build Custom Firmware with OpenWrt

The OSWALD Project: Open Source and Education

Transparent Firewall Primer

Cool Projects

Drivers for an FPGA-BASED ROBOTIC CONTROLLER

Control Your Refrigerator with Linux

Make a Weather Station with wview

Create Real-Time Plots with kst and Arduino

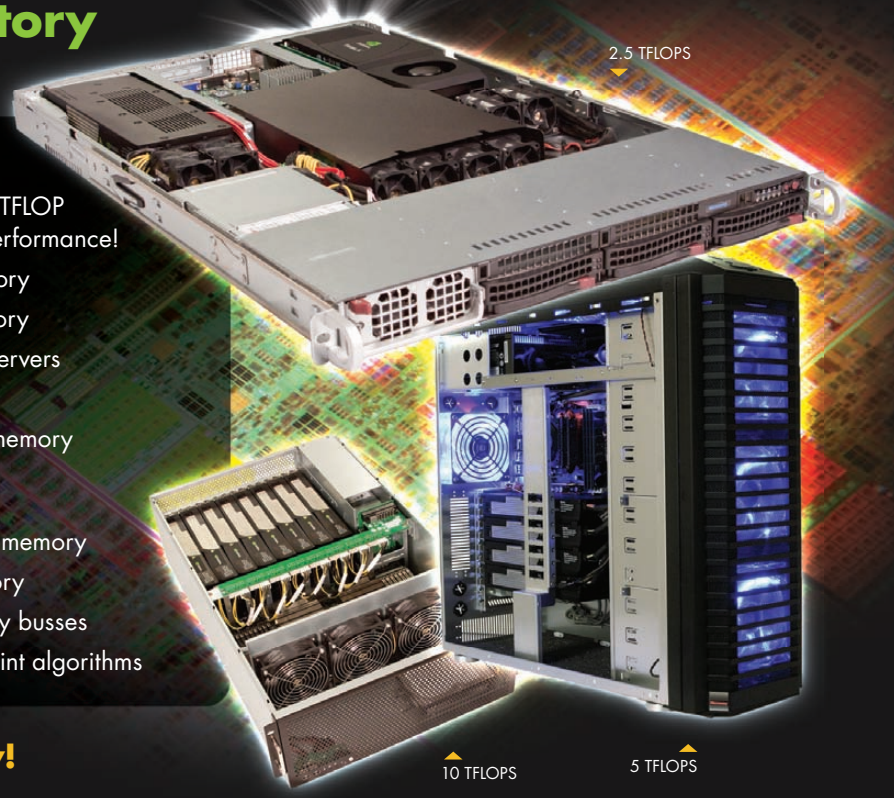
PLUS

Point/Counterpoint:
SOLID-STATE DRIVES vs. ROTATIONAL MEDIA

$5.99US $5.99CAN
08
0 09281 03102 4

More TFLOPS, Fewer WATTS

Microway delivers the fastest and greenest floating point throughput in history



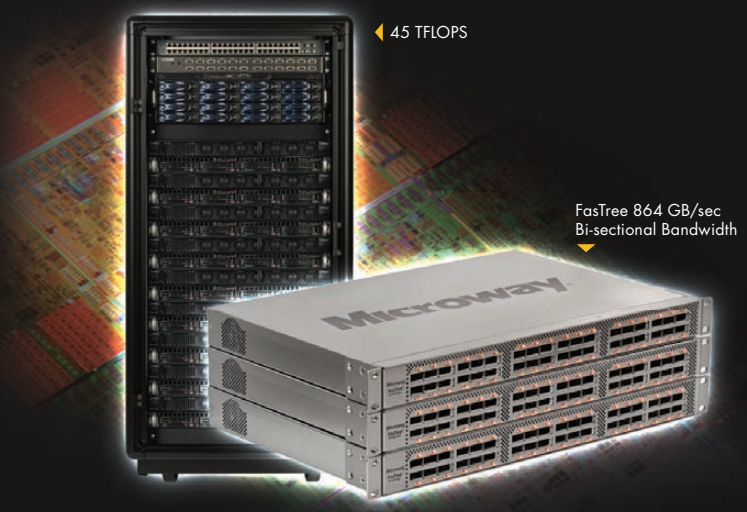
Enhanced GPU Computing with Tesla Fermi

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

New Processors

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

Configure your next Cluster today!
www.microway.com/quickquote
508-746-7341



FasTree™ QDR InfiniBand Switches and HCAs

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

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

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

NVIDIA Tesla CUDA | Intel Channel Partner Premier 2010 | AMD Fusion Partner Premier | GSA

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

Microway®
Technology you can count on℠

1 YEAR FOR FREE!*

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

1&1® HOME PACKAGE

- 2 Domain Names Included (.com, .net, .org, .info or .biz)
- 150 GB Web Space
- UNLIMITED Traffic
- 10 FTP Accounts
- 25 MySQL Databases (100 MB)
- Extensive Programming Language Support: Perl®, Python®, PHP 5/6 (beta) with Zend® Framework
- 1,200 E-Mail Accounts (IMAP/ POP3)

1 YEAR FREE

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

1&1®

MEMBER OF united internet

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

www.1and1.com

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

CONTENTS

AUGUST 2010
Issue 196

FEATURES

COOL PROJECTS

ON THE COVER

COVER IMAGE: the "Biclops" by Bob Smith shows how easy it is to build robots when you have Linux on a Chumby and the peripherals on an FPGA card.

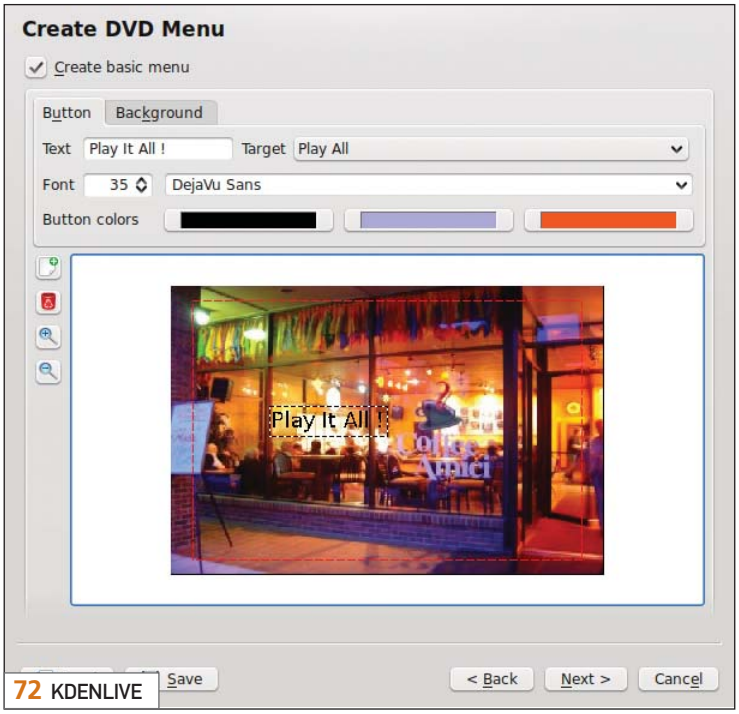
CONTENTS AUGUST 2010 Issue 196

COLUMNS

INDEPTH

IN EVERY ISSUE

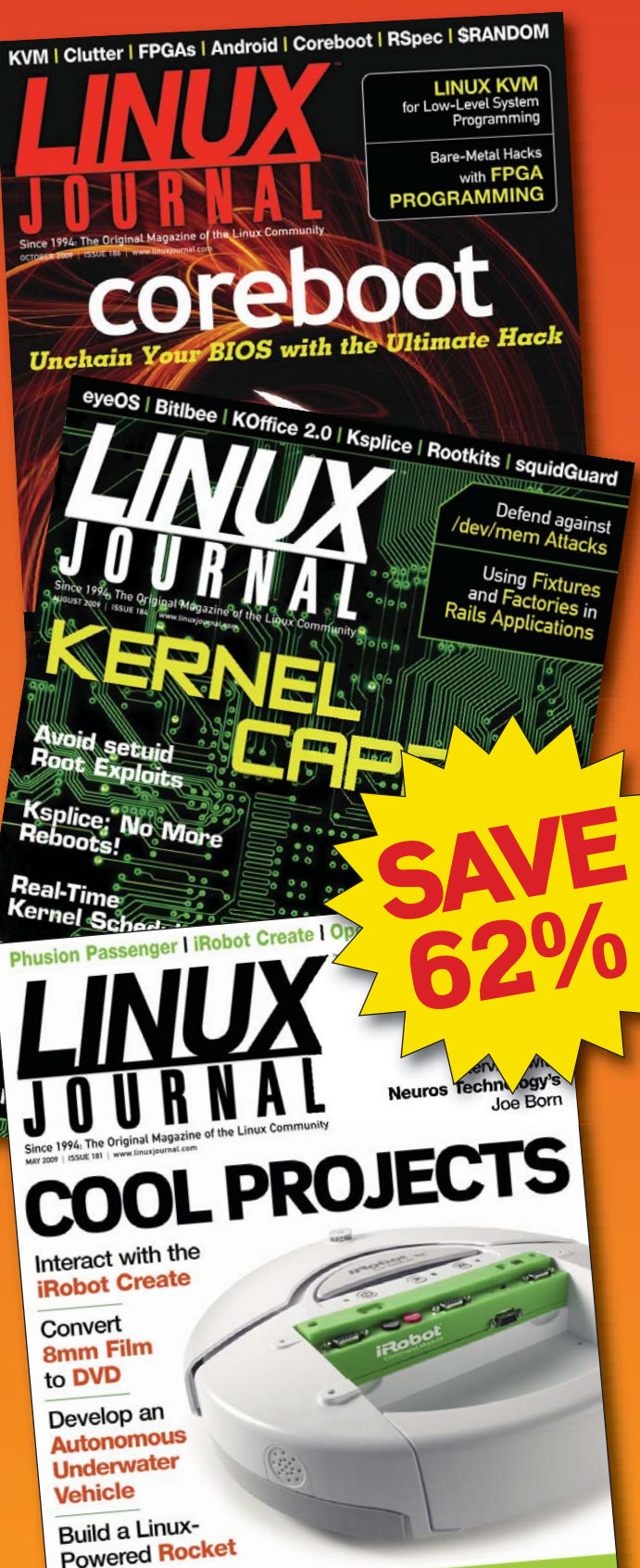
68 THE OSWALD PROJECT



72 KDENLIVE

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

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



SAVE 62%

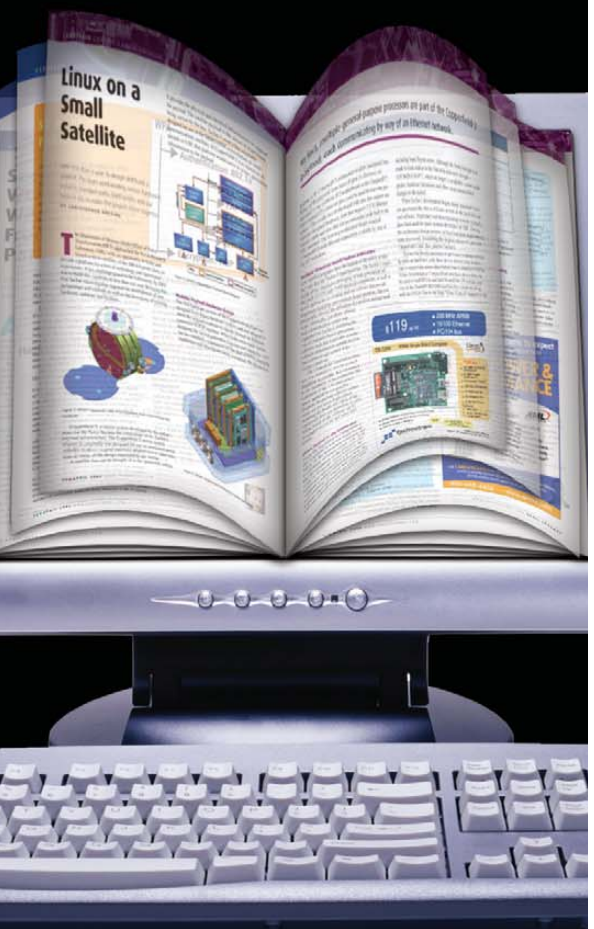
- In-depth information providing a full 360-degree look at featured topics relating to Linux
- Tools, tips and tricks you will use today as well as relevant information for the future
- Advice and inspiration for getting the most out of your Linux system
- Instructional how-tos will save you time and money

Get *Linux Journal* delivered to your door monthly for 1 year for only $29.50! Plus, you will receive a free gift with your subscription.

SUBSCRIBE NOW AT:

WWW.LINUXJOURNAL.COM/SUBSCRIBE

Offer valid in US only. Newsstand price per issue is $5.99 USD; Canada/Mexico annual price is $39.50 USD; International annual price is $69.50. Free gift valued at $5.99. Prepaid in US funds. First issue will arrive in 4-6 weeks. Sign up for, renew, or manage your subscription on-line, www.linuxjournal.com/subscribe.



LINUX JOURNAL

ABERDEEN
SERVERS AND STORAGE

SUCCESS!
I'VE FOUND IT!

Scream it from the mountain top, "ZFS storage I can afford!"

We hear you. Deliver simultaneous file and block-level storage without volume limitations, and throw in enterprise class data protection features. Aberdeen gets it. The AberSAN Z-Series brings enterprise level benefits with a sub-$9,000 starting price.

PROTECTION

Real-time block-level mirroring
Unlimited snapshots

SCALABILITY

Unlimited array size with storage pooling
Expand capacity to beyond a Petabyte

DEDUPLICATION

No partion limitations
Active/Active storage fail-over

VIRTUALIZATION

Thin provisioning
VMware Certified management

RELIABILITY

Inline bidirectional replication
Intel® Xeon® processors 5600 series
Standard 5-year warranty

888-297-7409
www.aberdeeninc.com/lj034

SHAWN POWERS

The Mad Scientist Starter Kit

There's something about the Cool Projects issue that always makes me think of science fiction. In my youth, I used to tape electric motors to popsicle sticks to form makeshift propellers on fanciful airplanes. Of course, I also used to make shocking devices from the ignition coils of old abandoned cars, so it's not as if I were terribly innocent in my tomfoolery. This month, although we don't have any stun guns to fend off bully attacks, we do have some extremely fun and interesting projects bound to inspire those creative juices of youth.

Anton Borisov starts off the issue with an interview with Alberto Broggi. Alberto is designing a car that will drive from Italy to China. That may sound insignificant, but it's important to realize I'm being literal. The car *itself* will be driving from Italy to China. The best my car does on its own is veer left of center if I start to fall asleep. Alberto is hoping to create computer systems to stop needless accidents by putting tireless computers behind the wheel. One part terrifying and one part awesome, it's a project I'm excited to read about.

If the thought of computer-controlled vehicles stresses you out, you might want to speak with Kyle Rankin. This month, he shows off his Linux-powered refrigerator, which he happens to use to brew beer. I've never had Kyle's beer, but knowing Kyle, the recipe is probably open sourced. He most likely would give us all the gruesome details if we asked.

Although Kyle may be using Linux to take over his fridge, the mad scientist in me thinks he might be setting his scope too small. Perhaps taking over the world with killer robots is more your line of thinking. If so, read Bob Smith's article on controlling robotic peripherals with Linux. It seems clear to me that criminal masterminds would want to use Linux for their killer robot army, and with Bob's article, your world takeover will be a bit easier.

Perhaps world domination and computer-driven cars are little more than you're up for this month. That's fine, we need to save something for next year's Cool Projects issue. Check out Mark Teel's article on wview, a Linux-based program for communicating with weather station devices. Although wview won't necessarily enable you to control the weather, it might help a bit with predicting it. Maybe if the local weatherman used Linux, he'd be correct about the weather here in Michigan more often.

Do you like tinkering with Arduino boards? If so, Rob Reilly's article on kst might tickle your fancy. Rob shows how to make data plots with kst and a Linux machine. Or, perhaps you want to roll your own firmware. If you have a compatible router (there are quite a few), you can turn your off-the-shelf router into a full-fledged server with OpenWrt. Mike Petullo shows how.

Finally, we have OSWALD. Don't confuse the cute name for WALL-E; rather, the OSWALD unit is an open computing device developed at Oregon State University. It was designed to get computer science students excited about developing. With its unique open nature, students are able to bend OSWALD to their whims and come up with creative new programming in an era when students often feel all the cool innovations already have been invented. Hopefully, OSWALD will change that.

If all this talk of gadgets, robots and death rays isn't really your cup of tea, that's okay. We have our full lineup of columns focusing on the things that keep us excited about Linux month after month. Reuven M. Lerner continues telling us about using CouchDB. Mick Bauer discusses transparent firewalls. Dave Taylor shows how to use signals in shell scripts. Dirk Elmendorf has a few pages on smaller projects you'll want to check out. And, of course, Bill and Kyle are at it again. Whether you prefer rotating magnetic media or the newer solid-state devices, it's always fun to see them battle out their opinions in the Point/Counterpoint column.

As always, we have new product announcements, tech tips, and all the other things you look forward to in *Linux Journal*. In my part of the world, summertime is well underway now, so perhaps I'll avoid that hot sun by building a few projects at home. Hopefully, unlike in my youth, I won't do dumb things like fill my basement with chlorine gas. But, that's a story for another issue.■

Shawn Powers is the Associate Editor for *Linux Journal*. He's also the Gadget Guy for LinuxJournal.com, and he has an interesting collection of vintage Garfield coffee mugs. Don't let his silly hairdo fool you, he's a pretty ordinary guy and can be reached via e-mail at shawn@linuxjournal.com. Or, swing by the #linuxjournal IRC channel on Freenode.net.

servers DIRECT®

MORE PRODUCTS, BETTER SERVICE, GUARANTEED.

GO STRAIGHT TO THE SOURCE!

1.877.727.7887 | www.ServersDirect.com

WANT YOUR BUSINESS TO BE MORE PRODUCTIVE?

The ServersDirect® Systems powered by Intel® Xeon® Processor provides the quality and dependability to keep up with your growing business.

HIGH EFFICIENCY POWER & HIGH STORAGE CAPACITY SERVER
SDR-S4314-T36

ENTERPRISE-LEVEL HIGH CAPACITY STORAGE SERVER

+ Maximum 3.5"" hot-swap drives density
  - 36x (24 front + 12 rear) Hard Drive Bays
  - E1: Expander supports SAS
+ Redundant (1+1) 1400W Gold Level power supply with PMBus function
+ Redundant, Hot-pluggable cooling system, Power Supplies, Hot-swap drives
+ Support up to 192GB DDR3 1333/ 1066/ 800MHz ECC Reg DIMM
+ Intel® 82576 Dual-Port Gigabit Ethernet Controller

80 PLUS

SDR-S1343-T04

1U INTEL® XEON® PROCESSORS 5500 SERIES SERVER W/ 4X 3.5" HOT-SWAP SATA DRIVE BAYS

- Supermicro 1U Rackmount Server with HE 560W Power Supply
- Supermicro Server Board w/Intel® 5520 Chipset
- Support up to Dual Intel® 5500 series Xeon® Quad/ Dual-Core, with QPI up to 6.4 GT/s
- Support up to 96GB DDR3 1333/ 1066/ 800MHz ECC Reg. DIMM
- 4x 3.5" Hot-swap SATA Drive Bays
- Intel® 82576 Dual-Port Gigabit Ethernet Controller

80 PLUS

SDR-S2318-T06

2U INTEL® XEON® PROCESSORS 5500 SERIES SERVER W/ 8X 3.5" HOT-SWAP SAS/SATA BAYS

- Supermicro 2U Rackmount Server w/650W HE Power Supply
- Supermicro Server Board w/Intel® 5520 Chipset
- Support up to Dual Intel® 5500 series Xeon® Quad/ Dual-Core, with QPI up to 6.4 GT/s
- Support up to 192GB DDR3 1333/ 1066 / 800MHz ECC Reg DIMM
- 6x 3.5" Hot-swap SAS / SATA Drive Bays
- Dual Intel® 82574L Gigabit Ethernet Controller

80 PLUS

SDR-S3308-T08

3U INTEL® XEON® PROCESSORS 5500 SERIES SERVER W/ 8 X 3.5" HOT-SWAP DRIVES TRAYS

- Supermicro 3U Rackmount Server w/650W HE Power Supply
- Supermicro Server Board w/Intel® 5520 Chipsets
- Support up to Dual Intel® 5500 series Xeon® Quad/Dual-Core, with QPI up to 6.4 GT/s
- Support up to 192GB DDR3 1333/ 1066 / 800MHz ECC Reg DIMM
- 8 x 3.5"" Hot-swap Drives Trays 6 x SATA Hard Drives Supported
- Dual Intel® 82574L Gigabit Ethernet

SDR-S4309-T08

TOWER/4U INTEL® XEON® PROCESSORS 5500 SERIES SERVER W/ 8 X 3.5" HOT-SWAP DRIVES TRAYS

- Supermicro Tower 800W Redundant Power Supply
- Supermicro Server Board w/ Dual Intel® 5520 Chipsets
- Support up to Dual Intel® 5500 series Xeon® Quad/Dual-Core, with QPI up to 6.4 GT/s
- Support up to 192GB DDR3 1333/ 1066 / 800MHz ECC Reg DIMM
- 24x 3.5" Hot-swap SATA Drive Bay
- Intel® 82576 Dual-port Gigabit Ethernet Controller

80 PLUS GOLD

SDR-S4314-T08

TOWER/4U INTEL® XEON® PROCESSORS 5500 SERIES SERVER W/ 8X 3.5" HOT-SWAP SATA BAYS (SUPPORT UP TO 4 DOUBLE-WIDTH GPU)

- 4U Rackmountable / Tower with 1400W Gold Level Redundant Power Supply Optional Rackmount Kit
- Supermicro Server Board w/ Dual Intel® Dual Chipset ICH10R
- Support up to Dual Intel® 5500 series Xeon® Quad/Dual-Core, with QPI up to 6.4 GT/s
- Support up to 192GB DDR3 1333/ 1066 / 800MHz ECC Reg DIMM
- "• 8 x 3.5"" Hot-swap Drives Trays Supports up to 6 SATA Drives"
- Intel® Dual 82574L Gigabit Ethernet
- **4x NVIDIA Tesla C1060 GPU Cards**

SDB-S7002-T00

9U INTEL® XEON® PROCESSORS 5500 SERIES SERVER W/ 50X HOT-SWAP SATA II / SAS BAYS

- Supermicro SBE-714D-D28 Enclosure chassis with two 1400W power supplies. Up to 14 hot-plug processor blades
- Support Intel based blades
- One Management modules comes standard with each enclosure
- One hot-plug management modules providing remote KVM and IPMI 2.0 functionalities
- Support one hot-plug Gigabit Ethernet switch. Pass - Through Module

10th year

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

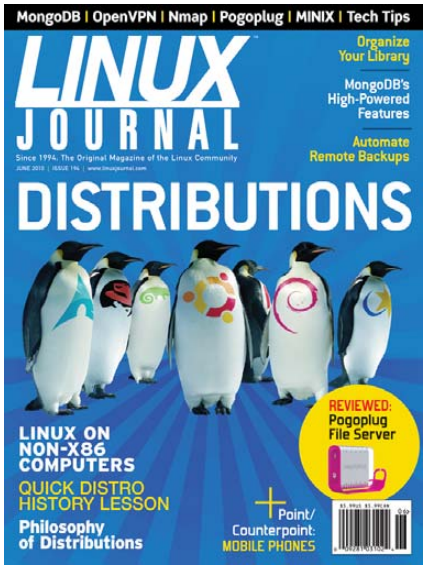
Our flexible on-line products configurator allows you to source a custom solution, or call and our product experts are standing by to help you to assemble systems that require a little extra. Servers Direct - your direct source for scalable, cost effective solutions.

1.877.727.7886 / www.ServersDirect.com

intel Xeon inside

Powerful. Intelligent.

letters



Commons Interests

Regarding Doc Searls' EOF in the June 2010 issue: you didn't do anything wrong with your ice pictures. They are yours to distribute as you like. All the critics of how you handled things are just jealous! My understanding of the Creative Commons license is just like the GPL—the license means others can use your copyrighted work without explicit permission as long as they abide by the license. If they want to use it in another way, they must get your permission, just as they would if you hadn't placed it under Creative Commons (or GPL, etc.).

NBC requested a special license, and you gave them one. So what?

If you had never placed your work under a Creative Commons license, NBC never would have found it, and it never would have been used. Bonus to you!

The license requirements under Creative Commons, GPL and similar licenses, apply only to use without permission. If you get permission from the copyright owner to use the object under different terms, you are free to do so.

NBC probably should have asked for something in writing just to protect themselves and verify the alternative license (and perhaps they did), but otherwise, I don't see how they were bound by the terms of the Creative Commons license you published the photos under.

--
Alan Shea

Which Is It?

Just curious as to which Distro logo is the one between Ubuntu and Debian on the front cover of the June 2010 issue (**www.linuxjournal.com/forums/june-issue-194**).

--
David

Novell.—Ed.

Grandparents Are Power Users Too

I am a subscriber to your magazine, and I do not appreciate Dan Sawyer's comment "grandparents incapable..." [see "Philosophy and Fancy" in the June 2010 issue]. I have installed many dual-boot distributions, and I program a little when I need to in shell, awk and some C. And, with great assistance from my children, I installed LFS from source. I've also worked on recompiling the kernel, but I don't like to do it.

--
GrandparentNancy

Dan Sawyer replies: *The hypothetical grandparent who is unfamiliar with/afraid of technology is a longstanding hypothetical user in the geek world, just as is the teenager who knows everything about computers. Neither of these hypothetical people are meant to imply that all teenagers are geek gods or that all grandparents are technophobic. It's just a long-established shorthand for talking about idealized user types in a way that's instantly accessible, in the same way that "Soccer Moms" are an idealized voter demographic.*

Apologies for the offense, and welcome to the power-user club!

The Magazine and the Digital Edition

I subscribed about a year ago, but having read the issues I received, I decided to not renew, because almost all the articles were well above my capability of understanding. My wife and I are merely ordinary computer users who, having become annoyed with Microsoft Windows and its myriad problems, decided to try Ubuntu Linux two years ago. Both she and I have Ubuntu installed on our respective computers (9.10, soon to be upgraded to 10.04), and we've never looked back.

As I said, the overwhelming majority of your articles are of no interest to me, but then came the May 2010 issue with *two* extremely interesting and useful articles (useful to me, and I'm sure others): "Full Speed Ahead with Handbrake" by Anthony Dean and "Comparing Five Music Players" by Bruce Byfield.

These are the type of articles I would like to see more of in the magazine! After reading Mr Dean's article, I immediately installed Handbrake, and following his detailed instructions, I was able to use it to convert some of my videos. And, one of these days (when I can overcome inertia), I'd like to comment on Mr Byfield's article.

So I decided right then and there to renew my subscription, and I added the digital edition as well.

I received my first digital edition today, and I have to state that I am disappointed. The disappointment comes from the way it's formatted. Like most digital magazines, one must scroll up and down the pages. If I adjust the file so each page fits completely on one screen, the print is too small to read. (My computer has a 15" screen.)

I had hoped that *Linux Journal* would have its digital edition formatted in the manner of *Full Circle Magazine*. Have you seen that magazine? *That* is the way a digital magazine should be formatted. Each page fits completely on the computer screen, and the print is easily read. Reading columns is quite easy in this format; no scrolling up and down and up and down is necessary. And to read further, you merely click on "next page" button.

So, in conclusion, I'd like to say that I

sincerely hope you will publish more such "basic" articles as those two I mentioned above; more of these would make the magazine truly useful to computer "unsophisticates" such as myself. (Of course, your current types of articles should continue to be published for those who can appreciate them.) And, I'd like to suggest that you consider reformatting the magazine's digital edition in the manner of *Full Circle Magazine*. Thank you for reading this and for considering my suggestions.

--
Lawrence H. Bulk

Thank you for bringing the "tech level" matter to our attention. It's always our goal to bring content that covers our entire readership, but sometimes we need to be reminded when we are leaving a group feeling a bit left out.

As far as the digital issue goes, it's been an ongoing discussion and struggle for more than a year now. I'm confident someday we'll have a perfect eBook version of Linux Journal *available, but right now, the PDF offers fewer sacrifices in layout quality and maintains the look of the print magazine. We realize it's not ideal, but so far, it is the best option we have. I haven't seen* Full Circle Magazine*, so I'll have to check it out.—Ed.*

LJ Infects Another

I have been buying Linux magazines, but most of them are *so* expensive. Then I found a copy of *Linux Journal* without DVD—really, who needs it? Digital subscription—ooh yeah! You just "infected" another Linux user in the country of cheese (The Netherlands).

--
Chihwah Li

I'm thrilled you found a copy of Linux Journal*! For folks in other countries, the digital edition can be a great way to get a timely and affordable subscription. I'm glad it works for you. Welcome to the family!—Ed.*

webOS Left Out

In the June 2010 Point/Counterpoint ("Mobile Phones"), it was obvious neither party knew of the Palm webOS mobile platform. This is by far the most Linux hacker-friendly mobile OS out there. For example, OpenOffice.org and Xorg were just ported to the Palm Pre (**bit.ly/aUlWOv**)! Palm (and hopefully now HP) openly supports the Linux community. See **www.webos-internals.org/wiki/Main_Page** for more.

--
Michael

Kyle Rankin replies: *We are both well aware of the webOS platform, but we decided to limit our scope to Android, Maemo and iPhone devices (the latter mostly so I could troll Bill). We had to draw the line somewhere, so mobile OSes like webOS, Symbian and Windows Mobile were left out. Plus, at the time we wrote the column, the future of webOS looked pretty grim, but hopefully, we'll see new interesting things on the platform now that HP has breathed new life into it.*

Bill Childers replies: *Although webOS is on my radar (I have a Pre+ sitting on my desk neglected and unused here), Kyle and I tend to write about what we know best. Point/Counterpoint is all about immediate off-the-cuff responses. The column wasn't meant to be a shootout of mobile platforms, but rather it was aimed at arguing about what works for each of us in daily life. Thanks for mentioning webOS though!*

Tech Tip

On the *Linux Journal* 101 Tech Tips DVD, you talk about using rsync with copying large amounts of files. There is also a command-line option for the cp tool that does the same thing. It is -u or --update, for example, `cp -u /media/cd/*/srv/copyofcd`. I just thought I'd pass this along—thought it might be helpful.

--
Michael F. Trombley Jr.

Thanks for dropping us a line. I wanted to point out that `cp -u` *is not a good or reliable substitute for rsync. Using* `cp -u` *causes cp to copy the file only if the destination is missing or is newer than the source. Try the following:*

```
$ cd /tmp
$ mkdir a b
$ echo TEST >a/c
$ echo test >b/c
$ cp -u a/* b/
$ cat b/c
```

You'll see that b/c was not updated to match a/c (it still contains "test" and not "TEST"). Because b/c was created after a/c, cp doesn't update it. Now, try:

```
$ rsync -a a/ b/
$ cat b/c
```

Here, you'll see that b/c equals a/c. rsync updates files if the contents of the source files are different from the contents of the destination files.

Furthermore, if cp decides to update a file, it copies the entire file from the source to the destination. On the other hand, rsync essentially breaks the file into chunks and copies only the chunks that have been modified. It's not a big deal on small files or across your local network, but with large files or across slow networks, this can make a huge difference in how long the update takes.—Ed.

Thanks for the June 2010 Issue

I just wanted to let you know how much I enjoyed the June 2010 issue of *LJ*. I always read Mick Bauer's Paranoid Penguin column, but I also enjoyed Mike Diehl's Pogoplug review as well as Dirk Elmendorf's article on library programs (I am a former Library Board President as well as a user of Alexandria). Bruce Byfield's article on MINIX was interesting to read, as was the article by Dan Sawyer. I currently use Ubuntu 10.4, but I also have used Fedora in the past and other members of my LUG use Mint and even Slackware. We have experimented with SUSE, Debian and MEPIS as well. Michael J. Hammel's article was a good reminder of why we do backups. In short, I found lots of great reading in the issue and thought I'd pass along my compliments to you.

--
Brian Clark

Every once in a while, a particular issue feels like it was written "just for me". It looks like the June 2010 issue did that for you. Thank you for your kind words, it's great to hear positive feedback. Hopefully, you'll keep getting "Brian Clark Issues" in the mail!—Ed.

Distro Chart Correction

Regarding the Linux Distribution chart in the June 2010 issue: I've been using

Linux (to a greater or lesser degree) since the mid-1990s (I still have my Yggdrasil Plug-and-Play Linux CD somewhere) and thought the chart very interesting (especially the column on "Best For"), because I'm looking for a replacement for Ubuntu 9.04 right now. However, whoever researched the chart needs to learn a little more, as you have the "First Release" date for Debian as 8/16/2003. According to DistroWatch, its first release was 6/17/96 (and I still have an old Infomagic CD set with Debian 0.93R6 dated April 1996 as physical proof). Tables are a great way to order information, but only if it's accurate information. Other than that, thank you for your magazine. It's been a treasure to read for many years.

--
Jamie Dimmel

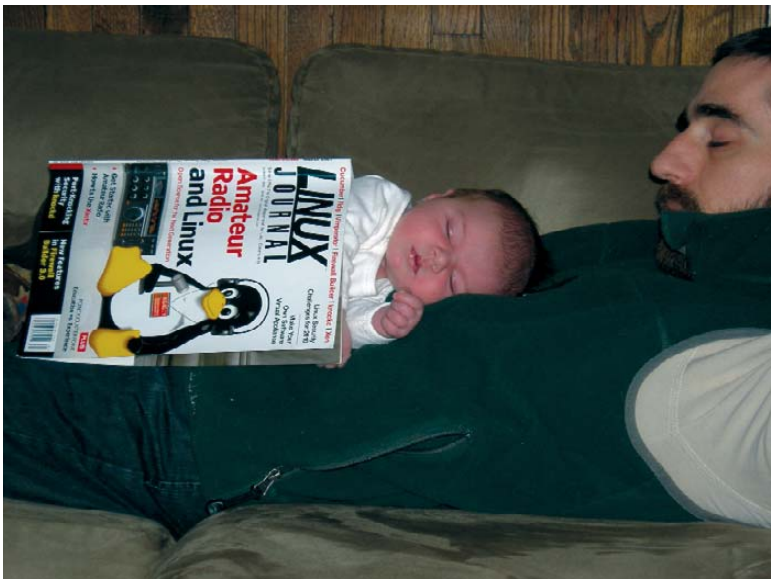
Justin Ryan replies: *Thanks for bringing this to my attention. I noticed it, much to my horror, shortly after the issue went out, but I knew a sharp-eyed reader would provide the opportunity to correct it. Sadly, like software, articles sometimes suffer from bugs, and some inevitably slip through. Thankfully, "given enough eyeballs, all bugs are shallow". I appreciate that you were on the (eye)ball.*

*The date in question was intended to be August 16, 1993, the date Ian Murdock founded the Debian Project. According to Debian's A Detailed History, "Debian 0.01 through Debian 0.90 were released between August and December of 1993" (***www.debian.org/doc/manuals/project-history/ch-detailed.en.html***). The June 1996 releases were the 1.0 series, released after Murdock left the project and Bruce Perens took on the role of Debian Project Leader.*

Thanks for your attention to detail and for providing the opportunity to highlight Debian's important place in the Linux community—as one of the oldest distributions still in active development (after Slackware), as the mother-distro of many other well-respected distributions, and as a reliable, innovative and rock-solid distribution in itself.

PHOTO OF THE MONTH

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



Two-week-old William Eidsness finds comfort in his open-source future. Father of two-week-old finds comfort in anything that makes William sleep. Submitted by Andrew Eidsness

diff -u

WHAT'S NEW IN KERNEL DEVELOPMENT

Jonathan Cameron had a bit of a disappointment when he tried to introduce a new subsystem for **ambient light sensors**. His reasoning was not poor. There were several existing drivers for ambient light sensors, and they were all scattered throughout different subsystems. His creation of a new subsystem amounted to little more than gathering those drivers together and writing up a little documentation. But, **Linus Torvalds** vetoed the whole plan, saying that ambient light sensors essentially were input devices and shouldn't be given their own subsystem. So, Jonathan's new **ALS subsystem** may not come to pass, but presumably folks now are working on gathering those drivers together into a saner organization.

Constantine Shulyupin is soliciting input on how to update his **map of the Linux kernel**, located at **www.makelinux.net/kernel_map**. It's a high-level view of the overall structure of the kernel, complete with links into **lxr.free-electrons.com**, where the various identifiers are defined and tracked.

Greg Kroah-Hartman has been maintaining the **staging tree**, but recently he started falling behind in considering patch submissions. **Joe Perches** asked why, and it turned out Greg had been giving presentations, writing articles and moving houses, with the result that there was now a 600-patch backlog. Various folks discussed possible changes to Greg's standard process, which included him just looking at patches as they came in and making decisions about them. Joe volunteered to help out. Greg was reluctant to change his work flow, especially since the problem had been the result of a confluence of events, but he did take Joe up on his offer and asked him to handle the incoming patch requests, while Greg continued to plow through the backlog.

Phillip Lougher hit a snag with **SquashFS**. He tried to update some very ugly code, and Linus Torvalds asked him to fix the ugliness first. Now Phillip is stuck with this thorny batch of ugliness that touches a number of different architectures, which aren't directly related to the SquashFS Project, but which he can't avoid working on if he wants his SquashFS patches to make it into the kernel. **Tim Bird** recently asked about the status of SquashFS, and that's basically what Phillip told him in reply. The patches will have to wait until Phillip finds the time to delve into those messy include files.

The **Linux Plumber's Conference** will be held in Cambridge, Massachusetts, this year, November 3–5. The overarching theme will be projects that span multiple interfaces, and thus, will involve groups of developers who might not normally work directly with one another. The format will be a set of brainstorming "tracks", where folks will try to figure out the best way to deal with these various cross-project issues. Check out **wiki.linuxplumbersconf.org/2010:topics** for the current list of "tracks" being considered.

The **SCST generic SCSI target subsystem** has migrated from using **ProcFS** for configuration to using **SysFS**. The new interface is documented in README files in the source tree. **Vladislav Bolkhovitin** announced the change recently, adding that the old ProcFS interface was now officially obsolete, which means that developers maintaining drivers, dev handlers and management utilities should start updating their code, with the expectation that the ProcFS interface will be going away at some point. The way this typically would work is that all active maintainers would migrate their code, and then someone would try to remove the SCST ProcFS interface. Next, someone would come out of the woodwork to say there's this or that driver that breaks, and then there would be another waiting period. The process would repeat until it became clear that any remaining code using the ProcFS interface is either completely unmaintained or has no users at all. Then the ProcFS interface will be removed, and possibly the projects still depending on it might be removed from the kernel as well.

—ZACK BROWN

HTML5, New Technology for Old Dogs

Everyone (myself included) is excited about HTML5 and how it will allow us to have richer content without proprietary plugins. Those who hate Adobe are excited about breaking away from Flash's dominance in the Internet scene, but there's also a group of folks excited to have *any* option on the Internet multimedia scene—PowerPC users.

Open-source alternatives to Flash certainly exist, but unfortunately, they don't work well on the majority of sites requiring Flash. Because Adobe has never released a PPC version of Flash for Linux users, those of us hoping to repurpose old Apple computers are left without any answer when users ask about Flash—and Flash is everywhere!

Will HTML5 be the knight in shining armor for folks running Linux on G4 and G5 computers? Will the Internet's transition to HTML5 take longer than what the old PowerPC beasties have left in them to run? We'll see pretty soon. I sure know I'd love to see our old Apple computers go back into service!

—SHAWN POWERS

Ubuntu One: Selling Free

Love it or hate it, cloud computing is here to stay. Handfuls of cloud services are being offered to computer users now, and like I've mentioned before, Canonical is doing the same. As it matures, Ubuntu One is beginning to roll more and more services into its feature set, including:

- File syncing (like Dropbox or Mobile Me for Macs).
- Preference syncing (Mobile Me does a bit of this).
- Contact syncing (Mobile Me and Google do this).

I'm sure as more developers take advantage of the open-source client for Ubuntu One, we'll see more and more applications take advantage of syncing, backup and off-line functionality.

Unfortunately, although Canonical does offer free storage in the cloud for all users, that doesn't mean all the features will be included in the free install. Contact syncing, for example, uses Funambol's technology to sync with mobile devices. That means Canonical has to pay for contact syncing and must pass that fee to the end users.

Certainly paying for a service isn't a bad thing. The unfortunate part is that Google *does* provide the vast majority of its services, including syncing contacts on mobile devices, for free. Hopefully, Canonical continues to keep the Ubuntu One API as open as possible, and the community will come alongside them in order to provide features that otherwise would cost money. I don't mind paying for storage in the cloud, but paying for features that are provided free elsewhere is a tough pill to swallow.

—SHAWN POWERS

LJ Store's Featured Product of the Month: the Ultimate Linux Gift Pack

The Ultimate Linux Gift Pack includes a one-year (12 issues) gift print subscription to *Linux Journal*. In addition to the one-year subscription, your recipient will receive a gift pack immediately. Items in the package include:

- A copy of the most recent issue of *Linux Journal* so the recipient can start reading before the subscription kicks in.
- A Powered by Linux magnet.
- A 2010 Linux Odyssey T-shirt.
- A pack of assorted stickers.
- A Penguin Party bumpersticker.
- An I Want YOU! button.
- A *Linux Journal* Archive CD-ROM featuring more than 180 back issues of the magazine.
- A 2010 Tux the Penguin wall calendar.

Retail value is $120. Yours for only $59.95 ($79.95 outside the US). Order yours today at **www.linuxjournalstore.com**.

LJ Index August 2010

1. Transistor count of an Intel 8080 microprocessor (in thousands): **4.5**
2. Transistor count of an Intel 80286 microprocessor (in thousands): **134**
3. Transistor count of an Intel 80386 microprocessor (in thousands): **275**
4. Transistor count of an Intel Pentium microprocessor (in thousands): **3,100**
5. Transistor count of an Intel Pentium 4 microprocessor (in thousands): **42,000**
6. Transistor count of an AMD K8 (Athlon 64) microprocessor (in thousands): **105,900**
7. Transistor count of an Intel Core i7 microprocessor (in thousands): **731,000**
8. Transistor count of an Eight Core Xeon Nehalem-EX microprocessor (in thousands): **2,300,000**
9. Transistor count of an RV820 ATI/AMD GPU (in thousands): **2,154,000**
10. Transistor count of a GF100 NVIDIA GPU (in thousands): **3,000,000**
11. Hard drive cost per gigabyte in 1990: **$53,000**
12. Hard drive cost per gigabyte in 1995: **$850**
13. Hard drive cost per gigabyte in 2000: **$20**
14. Hard drive cost per gigabyte in 2005: **$1**
15. Hard drive cost per gigabyte in 2010: **$.10**
16. RAM cost per gigabyte in 1990: **$119,706**
17. RAM cost per gigabyte in 1995: **$33,024**
18. RAM cost per gigabyte in 2000: **$1,598**
19. RAM cost per gigabyte in 2005: **$189**
20. RAM cost per gigabyte in 2010: **$21**

Sources:
1–10: en.wikipedia.org/wiki/Transistor_count
11–14: ns1758.ca/winch/winchest.html
15, 20: www.newegg.com
16–19: www.jcmit.com/memoryprice.htm

What Hardware Do I Have?

Often you may not necessarily know what kind of hardware you have—you may have a no-name box from a smaller company or a used machine. This month, I present the tools you can use to find out what you have installed.

First up is lshw. This utility LiSts HardWare (lshw). If you run it as a regular user, it actually warns you to run it as root. So go ahead and run `sudo lshw`. You should see screens of information for your system. The first section will be general information and should look something like this:

```
jbernard-eeepc
    description: Notebook
    product: 700
    vendor: ASUSTeK Computer INC.
    version: 0129
    serial: EeePC-1234567890
    width: 32 bits
    capabilities: smbios-2.5 dmi-2.5 smp-1.4 smp
    configuration: boot=normal chassis=notebook
    ➥cpus=1 uuid=XXXXXX-XXXXX-XXXXX-XXXXX
```

This is what I get when I run it on my little ASUS EeePC. Right away you can find the manufacturer of this little beast (ASUSTeK), the BIOS version (0129), and the fact that it's a 32-bit machine with one CPU. More information is broken down into the following categories:

```
core
   firmware - motherboard and BIOS information
   cpu - CPU information
      cache - cache information
   memory - memory information
      bank - specific bank memory information
   pci - PCI bus information
      display - PCI display adapter
      multimedia - PCI audio adapter
      pci - other PCI devices
      network - PCI network adapter
   usb - USB devices
   ide - IDE information
      disk - individual disks
         volume - volumes on this disk
```

For an idea on how much information is available, the main memory section shows this about my EeePC:

```
*-memory
     description: System Memory
     physical id: 1f
     slot: System board or motherboard
     size: 512MiB
   *-bank
        description: DIMM DDR2 Synchronous 400 MHz (2.5 ns)
        product: PartNum0
        vendor: Manufacturer0
        physical id: 0
        serial: SerNum0
        slot: DIMM0
        size: 512MiB
        width: 64 bits
        clock: 400MHz (2.5ns)
```

This utility is basically an all-in-one tool that spits out everything on your system in one go. But, what if you want information only about specific subsystems in your machine? An entire suite of utilities exists for this, and they might be more useful when you need some specific piece of information or want to do some system querying in a script.

You may want to look at the CPU. The lscpu utility provides output similar to the following:

```
Architecture:          i686
CPU op-mode(s):        32-bit
CPU(s):                1
Thread(s) per core:    1
Core(s) per socket:    1
CPU socket(s):         1
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 13
Stepping:              8
CPU MHz:               571.427
```

From this, you can see the manufacturer, whether it's 32-bit or 64-bit, the exact version and model, as well as the current CPU frequency.

If you want to know whether your video card is supported by X11, or whether you need to find a third-party driver, you can use lspci. This utility gives a list of all the devices plugged in to your PCI bus. The output looks something like this:

```
00:02.0 VGA compatible controller: Intel Corporation
 ➥Mobile 915GM/GMS/910GML Express Graphics Controller (rev 04)
00:02.1 Display controller: Intel Corporation
 ➥Mobile 915GM/GMS/910GML Express Graphics Controller (rev 04)
```

This information shows that the video controller in my EeePC is an Intel controller. So, if you wanted, you now could search Google with this information to learn about your video card and how best to configure it. If you want to see what USB devices are on your system, use lsusb. On my EeePC, I have an SD card installed, and it shows up as this:

```
Bus 001 Device 002: ID 0951:1606 Kingston Technology
```

If you're interested in the disk subsystem, you can find out what your system has with the blkid utility. This utility prints out all the available filesystems, with the following output format:

```
/dev/sda1: UUID="XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX" TYPE="ext2"
/dev/sda2: UUID="XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX" TYPE="swap"
/dev/sda3: UUID="XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX" TYPE="ext2"
/dev/sdb1: UUID="XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX" TYPE="ext2"
```

With this utility, you can learn what devices are available and

what filesystems are being used on them. The associated UUIDs also are available if you want to use them in the entries in /etc/fstab.

Now that you know what kind of hardware you have on your system, the last thing to check is to see whether your kernel actually is using the available hardware. On most modern distributions, the kernel is compiled to use modules. You can check to see which modules are loaded by using the lsmod command. You will get a list that looks like this:

```
agpgart                31788  2  drm,intel_agp
lp                      7028  0
video                  17375  1  i915
output                  1871  1  video
```

You can see that the agpgart module has a size of 31788 bytes and is used by the drm and intel_agp modules.

Now, hopefully, you can configure and optimize your hardware so that you get the most out of it. If you find other utilities not covered here, I would love to hear about them. **—JOEY BERNARD**

Android: Too Much of a Good Thing?

Android is everywhere. Really. It runs phones, tablets and recently, I even saw it running on an iPhone. Just a few years ago, that would have thrilled me to no end. Truthfully, it still does, but I'm more skeptical now. See, two years ago, Linux was everywhere on Netbooks. I thought it was a big break—Linux finally hit the mainstream.

But, vendor customization and "dumbing down" made Linux look like an inconsistent kludge rather than a free and powerful choice. So far, Android looks fairly consistent across hardware. So far, many apps work, regardless of the Android version your device supports. Hopefully, vendors will see the mistakes made with Netbooks, and keep their "branding" to a minimum. There are many ways phone vendors and wireless carriers could mess up our world domination efforts. Again.

Dear Vendors, please don't try to sell more phones by adding proprietary software on top of Android. If you add software, contribute it back to the community. If you want to sell more phones, make better phones than your competition. (Please!)

—SHAWN POWERS

NON-LINUX FOSS

Just because your phone is running Linux (and let's hope it is), that doesn't mean you can't talk to it with your Windows computer. Droid Explorer is an open-source program that allows you to manage your Android device in the same way that Windows Explorer allows you to manage your local files.

Droid Explorer supports multiple devices, allows you to copy files to and from your device easily, assists in applying system updates, provides a shell window, allows you to reboot into recovery mode, provides a package manager to install/uninstall Android Packages (APK), and it even lets you take a screenshot of your Android device. And, that's just for starters.

Droid Explorer is written in C# for the .NET platform. The current version is 0.8.4.3-Beta, and it lives at **de.codeplex.com**. If you're a Mono programmer looking for a new project, the developer is looking for someone to pitch in and help convert the UI to run on Mono.

Selecting Your Droid

Managing Your Droid's Packages

—MITCH FRAZIER

Readers' Choice Awards 2010

Vote for your Linux and open-source favorites in this year's Readers' Choice Awards at **www.linuxjournal.com/rc10**. Polls will be open through August 20, 2010. Winners will be announced in the December 2010 issue of *Linux Journal*.

AUTONOMOUS CARS RUNNING LINUX

This summer, two electric cars will be traveling from Milan, Italy, to Shanghai, China, on an ancient silk route (**vislab.it/Projects/view/32/VisLab%27s_new_adventure_on_the_Silk_road**). The driving will be completely autonomous, controlled by Linux recognition and processing software. The cars are being developed at the Artificial Vision and Intelligent Systems Lab of Parma University by **Alberto Broggi** and his team. Read on for an interview with Alberto about the project.

Anton: You recently announced the Milan-Shanghai initiative. Could you explain the main purpose behind it?

Alberto: We design driver-assistance systems to help drivers and reduce accidents (increasing road safety). With this trip, we are pushing our technology to the limit and trying to give the vehicles a lot of intelligence to be able to drive by themselves completely.

Anton: Do you mean that with this trip you're trying to get as much road statistics as possible?

Alberto: Not only that. The vehicle, instead of only helping the driver in some dangerous situations, will fully manage itself (it will have full control of gas, steering and brakes) for the *whole* trip. In other words, it will not only be active for a given set of situations, but also for the whole 13,000km. We also will record all the data, so we can play back the whole trip many times once we return to Parma. In the end, we plan to test other driving assistance systems in a huge number of completely different situations (such as mountain, flat, hilly, traffic, no traffic, rain, dust, off-road and so on).

Anton: What organizations helped you design the car and supportive technologies?

Alberto: It's just us, thanks to a grant I received from the ERC (European Research Council).

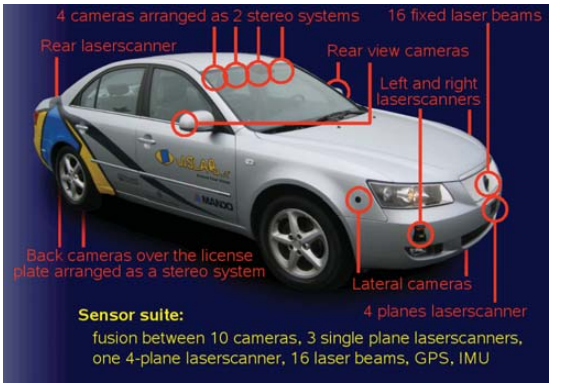
Anton: So you've got an ordinary Fiat car, you created supportive software, and here it goes?

Alberto: No, we have electric vehicles manufactured by Piaggio, an Italian motor company. The vehicles we will use on the China trip are electrical and will have a solar panel to power the electronic pilot. Originally, we wanted to recharge the driving batteries with the solar panel, but current technology is not mature enough. Therefore, we are powering only the sensors, the PCs and the actuators (namely, the whole automatic pilot). The vehicle we are using right now (but not the vehicle that will travel to China) is called BRAiVE (**www.braive.vislab.it**), and it includes a superset of the technology that we will have on our China vehicles. BRAiVE is a traditional petrol-based car.

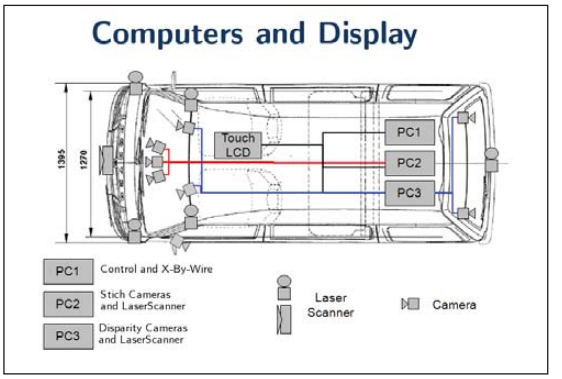
Anton: Elaborate on the vehicle's architecture—especially the software part of it.

Alberto: The architecture is simple: seven cameras, five laser scanners, GPS, IMU (Inertial Measurement Unit)—all connected to three Linux PCs. The PCs get the data, process it and send commands to actuators that drive the steering wheel, gas and brakes. All the software was developed by us during the last 15 years. We have been using this software for other challenges like the DARPA Grand Challenge or Urban Challenge.

Anton: The DARPA Challenge is performed in dry and sunny places. Did you try to run tests somewhere in more severe conditions, like in rain or snow?



A half-dozen cameras and sensors, and here's an automated Linux vehicle.



Three Linux PCs process data and generate a 3-D representation of the surrounding environment.

Alberto: We did tests in 2006 at the US Yuma base in Arizona (where we automated a PLS vehicle) and in 2001 in Antarctica (we automated a snowcat that followed tracks left by preceding vehicles). BRAiVE also works under light rain. Heavy rain is indeed a problem (as is heavy fog or snow).

Anton: How many prototypes had been developed before this roadrunner?

Alberto: Well, a good number! BRAiVE, TerraMax T2, Oshkosh PLS, Mini, Hummer—just have a look at our prototypes page: **vislab.it/Prototypes**.

Anton: How did you come to use Linux?

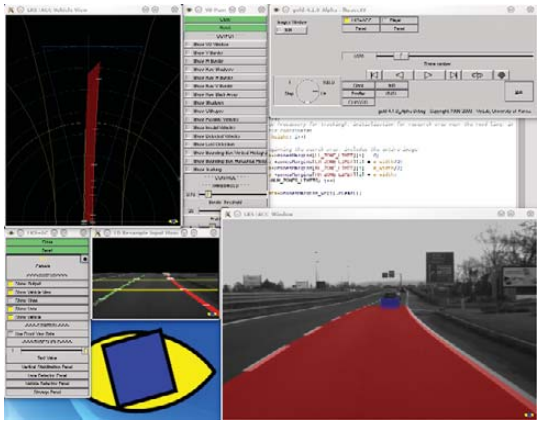
Alberto: I have been using Linux since it was created! Linux is more stable than Windows, and for these critical applications, it delivers the performance we need.

Anton: What flavor of Linux exactly do you use, and how do you process data (recognize video objects)—is this an HPC cluster or an ordinary PC?

Alberto: We started with Fedora 11, and we customized it to remove all features (services and software packages) typical of a desktop distribution, keeping only the minimum. We use Fluxbox as a window manager and gcc-4.4.3, kdevelop-3.9.99, gdb-7.0.1 as a development environment. We use normal PCs on a local LAN (Core 2 Duo T7100 at 1.8GHz 2MB Cache L2 FSB 800MHz, 1GB DDR2).

Anton: What components were programmed by your team and what has been done by the Open Source community?

Alberto: The real substance of our work is visual recognition and data processing, so my team has been busy with this in-house task for the past 15 years. Environmental reconstruction is the most difficult task. Once you have a 3-D representation of the surrounding environment, controlling the vehicle is



Driving conditions are recognized, so the vehicle continues its way forward.

not very complex. Our software is written in ISO C++, using some minimal additional open-source libraries like boost, loki and wxWidgets.

Anton: What general obstacles can your vehicle bypass now, and what obstacles can't it bypass yet?

Alberto: Any obstacle rising up from the road surface is detected. We can't detect small obstacles. For example, curbs are not always detected, but larger obstacles like pedestrians, vehicles and small animals are detected. Technically, we can locate obstacles that are sufficiently visible. Indeed, it takes time, and therefore, the system wouldn't be able to work at speeds such as 200mph (we'd need to increase our computer horsepower). Our electric vehicles run at slow speeds (the maximum speed so far is 70km/h), so we have no problems. Other systems (hardware and software) have been developed for other vehicles depending on their speed (based on more penetrating sensors and faster processing).

Anton: Where is such a technology (unmanned driving) already demanded, and in what spheres can you envision it will be useful to deploy?

Alberto: Definitely in the military field, but there's a huge demand in the agricultural domain. Think about autonomous tractors that move on the field automatically and work 24/7. And, we have been working (and are currently working) with Caterpillar for a number of years now, and they are interested in putting our technologies on their mining vehicles. Besides, through all these years we've done a couple auxiliary projects in aviation (project PEGASE), road construction (project TOPCON) and many others (**vislab.it/Projects**).

Anton: What stops them from using your technology right now?

Alberto: The environmental conditions are the most difficult challenge. We are now working to adapt our algorithms to have them working on very rough terrain, when the vehicle jumps on terrain bumps. Now we are able to reconstruct the world in 3-D with two cameras in real time (10Hz) and warn the driver about the presence of obstacles.

Anton: Tell us about your team.

Alberto: All the people on my team are engineers, almost all are PhDs and all of us have an electrical/CS background.

Anton: You've got an alumni procedure. What qualifications do you expect from a student joining your team to help further develop this prototype?

Alberto: Well, regarding this prototype, we already are almost done (we leave in less than three months). But for further projects (and we have plenty of them!), the ideal candidate should be well motivated and should be able to speak C++ fluently.

—ANTON BORISOV

They Said It

We all agree that your theory is crazy, but is it crazy enough?
—Niels Bohr

If you think education is expensive, try ignorance.
—Derek Bok

Pain is inevitable. Suffering is optional.
—M. Kathleen Casey

Bad is never good until worse happens.
—Danish Proverb

Dream as if you'll live forever. Live as if you'll die today.
—James Dean

Difficulty attracts the man of character because it is in embracing it that he realizes himself.
—Charles de Gaulle

Technology is like fish. The longer it stays on the shelf, the less desirable it becomes.
—Andrew Heller, IBM

The last good thing written in C++ was the Pachelbel Canon.
—Jerry Olson

Watch me not care.
—Dilbert

DISTRO SPOTLIGHT

EACH WEEK OR SO, we're choosing a Linux distribution to highlight on LinuxJournal.com. You'll see the Distro Spotlight on every page of our Web site. We hope you'll visit often and add a comment to cheer on your favorite, and also share your thoughts, tips and pitfalls. In the process, we hope to present some lesser-known gems, although we'll still celebrate crowd favorites. Be sure to check out the current pick, and maybe you'll find your new favorite distribution. If you're up to a fun challenge, try each distribution with us! Who doesn't love a reason to re-install Linux?

While you're at LinuxJournal.com, feel free to leave me a note to tell me how things at the site are working for you. You can find me at www.linuxjournal.com/users/webmistress.

—KATHERINE DRUCKMAN

CouchDB Views

Retrieve your CouchDB data using views and map-reduce functions.

REUVEN M. LERNER

Last month's column was an initial look at CouchDB, a non-relational, open-source database server, now sponsored by the Apache Software Foundation. CouchDB uses many Web-related standards: data is stored in JSON format, communication takes place using JSON and RESTful resources, and functions are written in JavaScript. CouchDB is not as speedy as some of the other non-relational (NoSQL) databases, such as MongoDB and Cassandra. However, CouchDB is designed to be dependable and easily replicated across multiple servers—a far cry from relational databases, for which replication remains slightly annoying at best.

Last month, I explained how once you have created a CouchDB database, you can use the curl utility to insert, update and remove documents. Each "document" is nothing more than a JSON object, which means it's basically a hash (a Python dictionary), which then may contain arbitrarily nested levels of arrays, hashes and scalar values (that is, strings and numbers). So, I can create a database with an HTTP `PUT` request:

CouchDB, however, offers a completely different query system, based on JavaScript functions and the map-reduce paradigm.

```
curl -X PUT http://localhost:5984/atf
```

Then, I can add some documents to that database with HTTP `POST` requests:

```
curl -X POST http://localhost:5984/atf
     -d '{"first_name" : "Atara", "middle_name": "Margalit",
          "sex":"f", "last_name" : "Lerner-Friedman",
          "birthday" : "2000-dec-16"}'

curl -X POST http://localhost:5984/atf
     -d '{"first_name" : "Shikma", "middle_name": "Bruria",
          "sex":"f", "last_name" : "Lerner-Friedman",
          "birthday" : "2002-dec-17"}'

curl -X POST http://localhost:5984/atf
     -d '{"first_name" : "Amotz", "middle_name": "David",
          "sex":"m", "last_name" : "Lerner-Friedman",
          "birthday" : "2005-oct-31"}'
```

Then, I can check to see that there are three documents, by using an HTTP `GET` request on the database:

```
bash-3.2# curl -X GET http://localhost:5984/atf

   {"db_name":"rmltest","doc_count":3,
    "doc_del_count":0,"update_seq":3,"purge_seq":0,
    "compact_running":false,"disk_size":12377,

"instance_start_time":"1273430793169153","disk_format_version":4}
```

As you can see, the `"doc_count"` attribute shows that there are, indeed, three documents in this database.

Now, if you have only three documents, querying them doesn't make much sense. But, if you have 300, or even 300,000 documents, you certainly are not going to want to iterate over them just to determine which is the best match and/or most appropriate.

If you were using a relational database server, you would use SQL to retrieve the rows that match a particular set of criteria. Even MongoDB, which I covered earlier this year, offers a query language that is vaguely SQL-like. CouchDB, however, offers a completely different query system, based on JavaScript functions and the map-reduce paradigm. CouchDB's syntax takes some getting used to, especially if you are relatively new at writing JavaScript functions. However, a few small functions can give you a great deal of power, which is (perhaps) the secret behind CouchDB's success.

CouchDB Views

I've already explained that CouchDB refers to each stored data item as a "document". CouchDB defined a special kind of document, known as a "design document", which contains a "view"—JavaScript code that is executed when you want to perform the query. (Design documents also may contain "show" functions, which can sort or otherwise modify the way in which data is displayed, but I won't discuss "show" functions in this column.) If you are developing only a database or applications, you might want to avoid the overhead of a permanent view by creating a temporary view instead. Temporary views take less time to set up and are a bit more flexible, but they execute much more slowly.

So, let's begin by creating a temporary view and some basic JavaScript views. For the data, I'm using the information I entered above, about my three children. (Feel free to substitute information about your own children, if you prefer.) I find it easiest to create temporary views using Futon, the Web-based administrative and maintenance tool that comes with CouchDB.

Simply point your Web browser at the server on which CouchDB is running, on port 5984, and go to your database of choice. Then, select temporary view from the pull-down menu in the top right-hand corner.

Your screen now should consist of two parts: on the left side, you have a simple JavaScript function, under the header "map":

```
function(doc) {
  emit(null, doc);
}
```

If you ever have used "map" in a language such as Ruby, Python, JavaScript or Lisp, this function already might make sense to you: your function is invoked repeatedly for a list of documents. If it produces a key-value pair, that pair is added to the output from the function running across all documents.

For instance, the example function (which is anonymous) takes a document as an argument, and returns a null key and the document itself as the value. If you click "Run" under the code, you get a set of results: three "null" keys on the left side and the original documents (with their mandatory _id field) on the right.

You can, of course, modify the function such that it outputs only information about girls. To do that, write:

```
function(doc)
{
    if (doc['sex'] == 'f')
    {
        emit(null, doc);
    }
}
```

Notice how by using a simple if statement, you can eliminate unwanted rows. Now, what if you're interested in getting all the documents, but sorted. CouchDB orders the results by their keys, which means the key you use is useful not only for identifying the resulting documents, but also for ordering them. You could, for example, sort the results by first name:

```
function(doc) {
        emit(doc.first_name, doc);
}
```

In my case, this means I first get the record for my son (Amotz), followed by Atara, followed by Shikma. Sorting by last name in this particular case doesn't help very much, because they all have identical last names. But, keys can be any data type, which means you even can use an array to arrange items, by last and then first name:

```
function(doc) {
       emit([doc.last_name, doc.first_name], doc);
}
```

You also can sort them by birthday:

```
function(doc) {
    emit(doc.birthday, doc)
}
```

However, this will not necessarily have the effect you want. The "birthday" field is a text string, which means the sorting will be done as a string, rather than as a date. (In the case of my children's birthdays, the sorting happens to work out fine, but this is a happy accident, not inherent to CouchDB.)

If you want to create a permanent view, there are a few ways to do so. You can use the Futon (Web-based) interface, and any temporary view can be turned into a permanent one by clicking on save as from the temporary view's screen. But another way, and one that's a bit more flexible if you're writing complex code, is to use curl to `PUT` a new design document on the server. This document contains JSON, like all other documents in CouchDB, but it has a number of fields that are treated specially by CouchDB. Here is my file, which I called simpleview.json:

```
{
    "_id" : "_design/example",
    "views": {
        "show_by_birthday": {
            "map" : "function(doc){ emit(doc.birthday, doc) }"
        }
    }
}
```

Then, I uploaded the contents of this file using curl, as follows:

```
curl -X PUT http://localhost:5984/atf/_design/simpleview
➥-d @simpleview.json
```

By using the -d flag and the @ sign, I was able to tell curl to upload the JSON from a file, rather than the command line. I uploaded it to a design document (as you can see from the `"_design/"` at the beginning of its name), with the view called simpleview. Once uploaded, I then could run it from Futon (by going to the menu item `"show_by_birthday"`), or by again using curl:

```
curl -X GET http://localhost:5984/atf/_design/simpleview/
➥_view/show_by_id
```

The results of the query are the same no matter what. Futon displays them in a nicer format, but it obviously would be easier for a program to work with the JSON output via HTTP. If you want to edit the view, you can either re-upload it via curl or use Futon to go to the view and edit the JavaScript function.

Reduce

So far, the examples here have focused on writing "map" functions, which take the list of documents and outputs a series of key-value pairs. But, as you might imagine, the map-reduce paradigm has two parts, the second of which is called reduce. The idea is that you use map to filter and transform the data into a list, and then use reduce to turn that list into something even more useful, typically a single value. For example, you can define the reduce function as follows:

```
function(keys, values, rereduce) {
    return 1;
}
```

From within Futon, this returns a list of three documents, the keys of which are the birthdays, and the values of which is the number 1. This isn't very interesting, to be honest, because you could have accomplished the same thing from within the "map" request. But, if you invoke the same query from curl, you get something else entirely:

```
{"rows":[
    {"key":null,"value":1}
]}
```

Why the difference? And, why is there only a single row now?

The answer is that reduce is designed to, well, reduce things. That means instead of returning a result for each row, it returns a single result from all rows. The reduce function actually can be invoked in two ways:

- The usual way, in which the "keys" and "values" represent document keys and values. In such cases, the "rereduce" parameter is set to false.
- For rereducing, the "rereduce" parameter is set to true, the keys are null, and the values represent values from a previous, partial run of the "reduce" function.

As the CouchDB Wiki states, this means the reduce function must be both commutative (that is, the order in which arguments are processed doesn't matter) and associative (that is, the order in which operations are executed doesn't matter). This often (but not always) means the reduce function ends up performing addition or multiplication, returning the result of executing the function on all documents. One example of a reduce function that I found calculates the standard deviation from mapped results, so you can find out how similar the documents are along at least one dimension.

Learning to use map-reduce in CouchDB can take some time. However, this paradigm has proven itself for a decade or so. For example, as the backbone for Google's search system, map-reduce has demonstrated excellent performance and flexibility. Granted, Google is using something (or some things) that are not CouchDB, but with a similar interface and paradigm.

Conclusion

If you like the idea of using JavaScript, MVCC, easy replication and JSON documents, CouchDB might be a good choice. It apparently is not as fast as some of its non-SQL competition, such as MongoDB and Cassandra. However, CouchDB's built-in (and sophisticated) Web interface, RESTful communication and the flexibility of map-reduce are all good reasons to use it. CouchDB is so easy to set up and use, why not try it out? Even if you end up not using it, CouchDB is a great way to learn about map-reduce and to try to create some small functions using it.■

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

Resources

The home page for CouchDB is at the Apache Project (**couchdb.apache.org**). There, you not only can download the software, but also read documentation, from tutorials to an active wiki. The CouchDB Web site also has links to drivers for the various languages you're likely to use when working with CouchDB.

If you're interested in the JSON format used by CouchDB, you can learn more about it at the main Web site: **json.org**.

Finally, two good books on CouchDB were released in the past few months. *Beginning CouchDB* by Joe Lennon, published by Apress, is aimed more at beginners, but it has a solid introduction to CouchDB, Futon and how you might use the system. *CouchDB: The Definitive Guide* by J. Chris Anderson, Jan Lehnardt and Noah Slater, published by O'Reilly, is a bit more advanced and meaty, but it might not be appropriate for beginning users of non-relational databases.

For a list of interesting map-reduce functions for CouchDB, along with explanations of how they work, see this page on the CouchDB Wiki: **wiki.apache.org/couchdb/View_Snippets**.


LINUXCON
NORTH AMERICA 2010
August 10th - 12th, 2010
Renaissance Boston Waterfront Hotel, Boston, MA
Keynote Speakers Include:
Ravi Simhambhatla
Vice President and Chief Information Officer at Virgin Atlantic
Wim Coekaerts
Senior Vice President, Linux and Virtualization Engineering at Oracle
Eben Moglen
Chairman at the Software Freedom Law Center (SFLC)
Rob Chandhok
President of Qualcomm Innovation Center, Inc. (QuIC)
Linux Kernel Roundtable Panelists:
James Bottomley
SCSI Subsystem Maintainer
Jon Corbet
Editor at Linux Weekly News
Dave Jones
Fedora Kernel Maintainer
Chris Mason
btrfs File System Creator
Ted Ts'o
Core File System Maintainer
THE LINUX FOUNDATION
REGISTER NOW FOR ONLY $400
http://events.linuxfoundation.org/

DAVE TAYLOR

Dealing with Signals

By handling signals in your bash scripts, you can provide features that are otherwise difficult, such as telling your script to reread its configuration file after it's already been started.

This month, I thought it would be interesting to take a bit of a detour from my usual multi-month programming projects and instead focus on a specific topic that is of great importance to people writing longer scripts: signal management.

Signals are numeric messages sent to running applications from the operating system, other applications or the user, and they generally invoke a specific response like "shut down gracefully", "stop running so I can put you in the background" or "die!"

Most likely, you've used the kill command to send signals to different programs, but if you've ever pressed Ctrl-C or Ctrl-Z to stop a running app, you've also sent signals to a running application.

A signal is managed in a cascading manner. It's sent to the application or script, then if the application doesn't have a specific handler (signal management or response function), it's pushed back to the shell or operating system. Some signals can't be managed within individual apps, like SIGKILL, which is caught by the operating system and immediately kills the running application (including the shell: SIGKILL your login shell and you just logged out).

Most likely, you've used the kill command to send signals to different programs, but if you've ever pressed Ctrl-C or Ctrl-Z to stop a running app, you've also sent signals to a running application.

To start this journey, let's find out what signals your version of Linux can handle. Do this by typing `kill -l` (that's a lowercase L, not the digit 1):

```
$ kill -l
1)  SIGHUP     2) SIGINT      3) SIGQUIT    4) SIGILL
5)  SIGTRAP    6) SIGABRT     7) SIGEMT     8) SIGFPE
9)  SIGKILL   10) SIGBUS     11) SIGSEGV   12) SIGSYS
13) SIGPIPE   14) SIGALRM    15) SIGTERM   16) SIGURG
17) SIGSTOP   18) SIGTSTP    19) SIGCONT   20) SIGCHLD
21) SIGTTIN   22) SIGTTOU    23) SIGIO     24) SIGXCPU
25) SIGXFSZ   26) SIGVTALRM  27) SIGPROF   28) SIGWINCH
29) SIGINFO   30) SIGUSR1    31) SIGUSR2
```

Most of these are uninteresting. The cool ones are SIGHUP, which is sent on a "hangup" or the user logging out; SIGINT, which is a simple interrupt (Ctrl-C, usually); SIGKILL, the "terminate with extreme prejudice" of signals; SIGTSTP, which is Ctrl-Z; SIGCONT, which is what the application gets from the shell commands fg and bg subsequent to a SIGTSTP; SIGWINCH, which is for window system events like a window resize; and SIGUSR1 and SIGUSR2, which are intended for interprocess communication.

Let's write some code to see what happens, shall we? Signals are caught with the "trap" built in, and the general format of these signal mapping commands is exemplified with:

```
trap 'echo "Ctrl-C Ignored" ' INT
```

How do we play with that as a shell script? Here's an easy way:

```
#!/bin/bash

trap 'echo " - Ctrl-C ignored" ' INT
while /usr/bin/true ; do
  sleep 30
done

exit 0
```

Did you catch the infinite loop there? It's barely using any resources because most of its time is spent in "sleep", but if you don't do something to end it, this script will run forever or until the Mayans are proven right two years from now—one of the two.

Let's look at a more flexible way to manage signals by creating shell script functions:

```
sigquit()
{
   echo "signal QUIT received"
}

sigint()
{
   echo "signal INT received, script ending"
   exit 0
}
```

```
trap 'sigquit' QUIT
trap 'sigint'  INT
trap ':'       HUP      # ignore the specified signals

echo "test script started. My PID is $$"
while /usr/bin/true ; do
  sleep 30
done
```

Run this then from another terminal window and shoot some signals at it.

Now, let's get that script started and watch what happens when we send a few different signals:

```
$ ./test.sh
test script started. My PID is 25309
signal QUIT received
signal INT received, script ending
$
```

Perfect! To send the signals, execute the following commands from a different terminal window:

```
$ kill -HUP  25309
$ kill -QUIT 25309
$ kill -INT  25309
```

Armed with this useful script, let's have a look at how to handle a more complex signal like Ctrl-Z within a shell script.

Stop! Don't Stop!

I'm going to create a scenario here rather than just going through the intellectual exercise. In a complex script, you realize that you have certain passages where you need to ignore the TSTP signal (SIGTSTP or Ctrl-Z or signal number 18) and other spots where it's fine to stop and restart. Can it be done?

To start working out a solution, I'll create a function that not only handles the specified signal, but also disables itself after a single invocation:

```
sigtstp()
{
  echo "SIGTSTP received" > /dev/tty
  trap - TSTP
  echo "SIGTSTP standard handling restored"
}
```

Invoke `trap - signal` somewhere else in the script,

and you've reset that signal handler, so if I have the line:

```
trap 'sigtstp' TSTP
```

right before the section where I don't want the Ctrl-Z to work, it'll ignore that first Ctrl-Z, then reset the signal handler and work as expected the second time you press that key.

More useful is to ignore all Ctrl-Z stop signals until you're ready to deal with them, and that's quite easily done with the minimalist:

```
trap : TSTP  # ignore Ctrl-Z requests
```

And, then when you're ready to receive them again:

```
trap - TSTP  # allow Ctrl-Z requests
```

Experimentation will show that there are some weird terminal buffering issues associated with SIGTSTP, however, so don't be surprised if you have a signal handler that has output. In this particular instance, it won't show up until the script quits.

Experimentation will show that there are some weird terminal buffering issues associated with SIGTSTP, however, so don't be surprised if you have a signal handler that has output.

Reading a Configuration File

Let's look at a more practical example. Say you have an admin script that is always supposed to be running as a dæmon, but occasionally you want to tweak its configuration file and have it reread its setup (a lot faster than killing and restarting it).

Further, let's use SIGUSR1 for this task, as that is its intended use, so we're using the kernel's signal handling subsystem in the manner it was intended.

Reading a configuration file might be something as simple as:

```
. $config
```

(Recall that using . means that any variables set in the secondary file affect the current shell, not a subshell. The source command does the same thing as the . command.)

Here's our script to experiment with this feature:

```
#!/bin/bash

config="our.config.file"

sigusr1()
{
  echo "(SIGUSR1: re-reading config file)"
  . $config
}

trap sigusr1 USR1        # catch -USR1 signal

echo "Daemon started. Assigned PID is $$"

. $config                # read it first time

while /usr/bin/true; do
  echo "Target number = $number"
  sleep 5
done

trap - USR1              # reset to be elegant

exit 0
```

We'll start with the configuration file containing `number=5`, then after 10–15 seconds, change it to `number=1`. Until we send the actual USR1 signal, however, the script just plugs along without a clue that it has changed:

```
$ ./test2.sh
Daemon started. Assigned PID is 25843
Target number = 5
Target number = 5
Target number = 5
```

Meanwhile, in another window, I've already edited the file, so I type in this command:

```
$ kill -USR1 25843
```

And, here's what happens in the main script window:

```
(SIGUSR1: re-reading config file)
Target number = 1
Target number = 1
```

Cool, eh?

I hope this exploration of signal handling in shell scripts is useful. I actually learned quite a bit about advanced handling as I researched the code here. I'm still a bit stymied about how to reset the output stream after catching a SIGTSTP, but I bet that some sharp *Linux Journal* reader will have an answer.■

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

September 11-13 2010
Columbus Convention Center
OhioLinux.org

Ohio Linux Fest

How will **FREE** change the world?

MICK BAUER

Building a Transparent Firewall with Linux, Part I

Yes, you still need a firewall. How about a transparent one?

When I started writing this column in autumn of 2000, I had the day job of firewall engineer. I enjoyed that type of work, but I was enough of a big-picture kind of guy to be aware that every year, firewalls were becoming less important in the overall scheme of things. In fact, I was convinced that within a decade or so, firewalls would be nearly if not completely obsolete.

But, I was wrong! Firewalls aren't dying. They're *evolving*, and even though traditional firewall technologies probably achieve less than they did ten years ago, the threats we'd face if we did without them still justify the effort and expense of keeping them around.

So, I think the time is ripe for me to return to my roots, so to speak. This month, I kick off a series of articles I've been meaning to tackle for some time: how to build a transparent firewall using Linux. To begin, I set the stage by explaining why firewalls are still relevant in the first place and the difference between "routing" and "bridging" (transparent) firewalls.

What Firewalls Do and Don't Do

I was tempted to begin with a primer on firewall architecture and design, discussing the difference between the multihomed and "sandwich" topologies, rule design methodology and so forth. But my April 2007 article "Linux Firewalls for Everyone" does that just fine, and you still can read it on-line (see Resources). Instead, I'm going to talk about why firewalls are still useful.

An IP firewall, as opposed to an application firewall or XML gateway, inspects network traffic at the IP and TCP/UDP layers, possibly with some very limited amount of application intelligence (for example, being able to distinguish between FTP "get" and "put" commands, HTTP "post" vs. "get" and so forth). In actual practice, most packet-filtering and "stateful" IP-filtering firewalls evaluate traffic primarily based on source and destination IP addresses, and source and destination UDP or TCP ports.

When I started out in network engineering, just filtering traffic based on these few criteria seemed plenty powerful enough for most practical firewall applications, especially if the firewall was smart enough to track network traffic by session rather than by individual packet. (In olden times, it wasn't enough to program your IP-filtering firewall to allow outbound DNS queries on UDP port 53; you also had to put in a corresponding rule to allow DNS replies *originating* from UDP port 53. State-tracking, or "stateful", firewalls automatically correlate packets to already-allowed sessions.)

Because in those days of yore different network applications all used different TCP and UDP ports—TCP port 23 for telnet, TCP and UDP ports 53 for DNS, TCP ports 20 and 21 for FTP and so forth—filtering by port number equated to filtering by application type.

This didn't mean firewalls could detect or prevent evil that might occur over an *allowed* source/destination/address/port combination. We had no illusions that the firewall could stop, for example, Apache buffer overflow attacks against a public Web server reachable from the entire world on TCP port 80. The firewall *could* (and can), however, prevent attempts to connect to that Web server via Secure Shell on TCP port 22, except perhaps from some internal, authorized access point.

The problem is that every year, we're less able to rely on the assumption that the things we should be worried most about will happen on ports that the firewall can block altogether. This is because so much of what people use networks for happens on only two TCP ports: TCP 80 (HTTP) and TCP 443 (HTTPS).

Even ten years ago, developers were racing to migrate from client-server application architectures, in which every network application used its own communication protocol, to the Web services model, in which there are really only two types of network transactions, Web sessions and database transactions. Well before then, people started figuring out ways to do practically everything that can be done over networks—from browsing a filesystem to running a remote desktop session—over HTTP using a Web browser.

But does this really mean that firewalls are obsolete? Definitely not, not even in contexts where Web servers are involved. Let's suppose it were true that *all* network traffic between two security zones happened over TCP port 443. By restricting traffic by source IP address and destination IP address, you still could make decisions about which hosts could initiate *any* transactions to any other given host.

If one filters traffic strictly based on source and destination IP addresses and, in practical terms, not by TCP/UDP port (service type), you may think that you haven't achieved much. All an attacker has to do in order to attack a protected system is gain access to some other system that the firewall allows to initiate transactions with your actual target. But, what if none of those "secondary" systems that the firewall considers trusted is externally reachable? If that's the case, your "crude" firewall rules may, in fact, have effectively mitigated the risk of remote compromise for that system.

This scenario is illustrated in Figure 1, which shows a firewall that sits in between two different networks, Zone A and Zone B, and the Internet. The firewall blocks all inbound traffic from the Internet to either zone, but allows hosts in Zone B to initiate transactions with hosts in Zone A. In this case, the firewall is highly effective in making it unfeasible for Internet-based attackers to exploit the trust relationship between Zones A and B, even if the firewall filters only on source IP address.

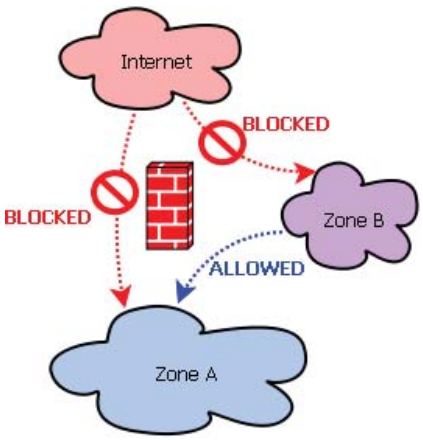


Figure 1. Filtering Only by Source IP Address

But, I'm not really advocating address-only filtering. The fact is there *are* still plenty of important network services that use ports and protocols besides TCP 80/443 and HTTP/HTTPS. Domain Name Services still use TCP and UDP 53; Microsoft Remote Desktop Protocol (Terminal Services) still uses TCP port 3389; Oracle still uses TCP port 1521; and so on. Modern firewalls still make plenty of meaningful decisions about what traffic to allow, based on service type.

Another argument against the usefulness of firewalls is the fact that malware (viruses, trojans, worms) has evolved from being mainly a nuisance to becoming a sophisticated means of infiltrating even well-secured networks. Ten years ago, the most likely impact of a virus or worm outbreak in one's organization was a disruption of service. Malware tended to strain system and network resources, but by its indiscriminate nature, it wasn't very useful for stealing data or breaching sensitive systems.

Nowadays, however, malware often is targeted at specific organizations by attackers who first go to great lengths to learn what data and systems to have their malware seek out, based on known (or highly probable) vulnerabilities on those systems. In other words, nowadays malicious hackers often deploy worms as "avatars" of themselves!

Such targeted malware is extremely difficult to detect, remediate against or trace back. Often, it's placed directly on target networks or systems by co-opted insiders. Therefore, firewalls frequently have little useful role in blocking the activity of targeted malware. It doesn't matter how thick your fortress walls are if your mailroom guys have been bribed to ignore the fact that the large package addressed to your king is ticking.

But I ask you, does the fact that bad guys may simply *mail* your king a bomb, mean that you can safely replace those expensive castle walls—the tuckpointing-bill for which in any given year is probably astronomical—with cardboard? I, for one, don't think you can.

Just because attackers are developing ever-more sophisticated tools doesn't mean they'll forget how to use the old ones. Remember the LAND attack from the late 1990s? It involved sending spoofed TCP packets bearing the same source IP address as the target to which you're sending them, causing a massive "reply to myself" sort of loop, which impairs system performance (potentially cripplingly). LAND was made obsolete by system patches and firewall protections—or so we thought. But in 2005, a security researcher named Dejan Levaja discovered that Windows Server 2003 and Windows XP SP2 were vulnerable

to the LAND attack.

The fact that LAND attacks are (still) trivially blockable by firewalls is actually beside the point. What I'm really trying to illustrate is that no attack is truly obsolete for as long as it's still feasible and as long as systems are vulnerable to it. In theory, if your organization rigorously hardened every single computer connected to the network, if you took down your firewalls you might not get infected or attacked right away. But I guarantee you would, sooner rather than later, and not strictly by completely cutting-edge attack techniques.

So, to summarize, even if you think all your firewall does is block traffic from unexpected sources, it still provides meaningful protection. Modern network traffic does not, in fact, consist solely of HTTP and HTTPS; it still plays out over a surprisingly wide range of different ports and protocols. And, the rise in use of targeted malware and attacks against Web applications aren't arguments against firewalls; they're simply reasons that firewalls *alone* aren't sufficient to protect critical systems.

It doesn't matter how thick your fortress walls are if your mailroom guys have been bribed to ignore the fact that the large package addressed to your king is ticking.

Can we agree, then, that firewalls are still an important tool in the network security toolkit? I hope so, because I'm about to spend several months showing you how to build a particularly clever type of Linux firewall: a transparent firewall.

Routing vs. Bridging Firewalls

Normally, a firewall acts like a router. It receives traffic from two or more network interfaces and makes decisions about whether and how to route that traffic. Any host that needs to send traffic through the firewall must either use the IP address of the firewall interface that faces it as its default gateway, or a router between that host and the firewall must be configured to use the firewall as a route to whatever networks are on the other side of the firewall.

Figure 2 shows a routing firewall. As you can see, each firewall interface has its own IP address that is valid on the network to which that interface connects, and that IP address serves as the route to the other side of the firewall. In this example, hosts in Network A have to know (or send packets to some router that knows) that 10.20.30.254 is the gateway to reach Network B. Hosts in Network B have to know (or speak to a router that knows) that 44.55.66.254 is the gateway to reach Network A.

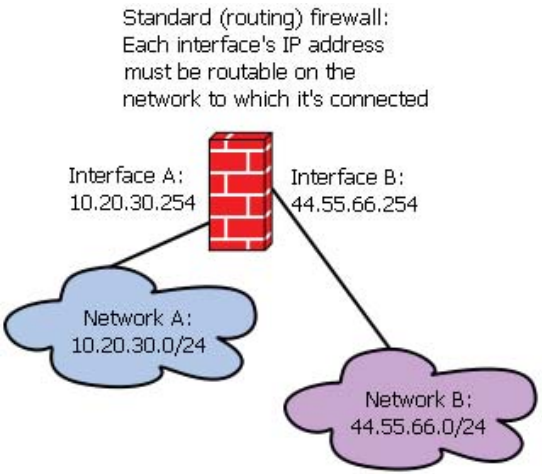


Figure 2. A Standard (Routing) Firewall

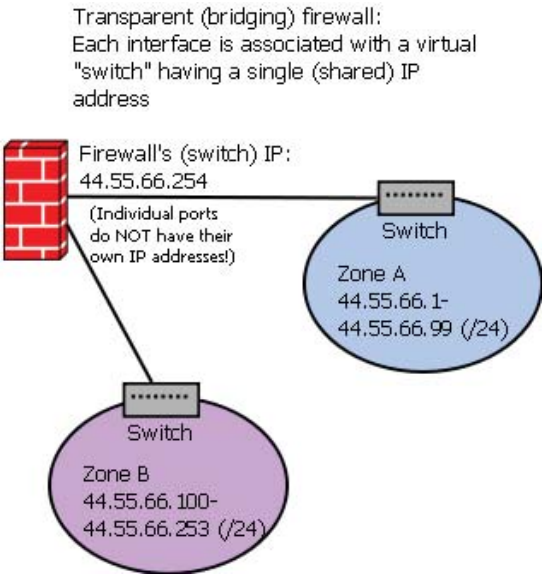


Figure 3. A Transparent (Bridging) Firewall

One ramification of the "firewall as router" approach is that normally, if you have a big bunch of existing systems you want to divide into two security zones, one "trusted" and the other "non-trusted", you'll probably need to re-IP-address the hosts in one or both zones (or re-mask the subnet they're on, which may not be possible) and insert a firewall configured as a gateway between those zones. In other words, inserting a routing firewall into an existing network usually means reconfiguring both the network and the systems connected to it.

But, what if you wanted to insert a firewall between two parts of the same network, without re-addressing *anything*? Is it possible to configure a firewall to act more like a bridge than a router?

Indeed, it is. And best of all, the firewall's rules will

What Is Bridging, Exactly?

This month's column assumes you know something about networking. IP firewalls, whether standard/routing or transparent/bridging, are at least an intermediate-level topic, and you really can't expect to configure either kind unless you have a working understanding of how TCP/IP protocols work. But even if you do, you might be a bit rusty on the difference between bridging and routing. Here's a quick, simplified primer.

Routers operate at Layer 3, the IP layer. They evaluate network packets based on their IP headers. In contrast, bridges operate at Layer 2, the Data Link (most commonly Ethernet nowadays) layer. Bridges evaluate Ethernet headers. A "switch" is simply a bridge with lots of ports.

To "route" is to make decisions on what to do with a packet based on its destination IP address—specifically, to determine the most efficient "route" (series of forwarding routers) by which to deliver the packet to its destination. These decisions can be complicated. There may be many possible routes, and because individual "hops" between gateways in a given route may become congested or otherwise slow down, the "best" route for a stream of packets may change during the course of a single network transaction.

For this reason, routers use a combination of routing algorithms, routing tables and forwarding tables to compute routes "on the fly". Note, however, that although firewalls act *like* routers, they seldom have to make nearly so complicated routing decisions as true routers typically do. Typically, a firewall uses a single gateway for all routes to "internal" networks and another for all other ("outside") networks, where both gateways are actual routers.

Bridging, in contrast, is much simpler. A bridge, or switch, looks at each incoming Ethernet frame entering each bridge/switch port and determines its destination Ethernet address (also known as a MAC address). It then matches this against the entries in its local MAC table (which is simply a list of the MAC addresses of all devices currently attached to active ports) and delivers the frame to the corresponding local port.

For more complete discussions of how routers and bridges/switches work, Wikipedia is a good place to start.

look and behave in much the same way as if it were a standard routing firewall! All the trickery is in the firewall's network configuration.

Figure 3 shows a transparent (bridging) firewall. In this example, the firewall has been configured to treat both of its network interfaces as switch ports. Neither interface has an IP address bound to it! Instead, the virtual switch that they comprise has a *shared* IP address of 44.55.66.254. Although the firewall might be reachable by this IP address (actually there are good reasons for it *not* to be), none of the hosts in Zone A need to use that IP as a gateway in order to reach the hosts in Zone B, or vice versa. Just as with any other switch, the firewall will propagate packets automatically without needing to be explicitly routed through.

However, the firewall will propagate packets only after first matching them against its firewall rule set and determining whether it even *should* propagate them. If you want to evaluate packets based on Ethernet header attributes, you can do so using ebtables. However, in this series of articles on building your very own transparent Linux firewall, we'll use plain-old iptables to evaluate packets in the same way that routing firewalls do, using IP/TCP/UDP header information.

Conclusion

But, that will have to wait until next time. Hopefully, you now understand the difference between a standard, routing firewall and a transparent, bridging firewall. In my next column, I'll sketch out an example usage scenario (conceptually very similar to the network in Figure 3), describe a couple different approaches to selecting Linux firewall hardware and begin showing how to configure a transparent firewall, starting with bridge/switch setup. Until then, be safe!■

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

Resources

"Linux Firewalls for Everyone" by Mick Bauer, *LJ*, April 2007: **www.linuxjournal.com/article/9569**

Internet News' report on Dejan Levaja's latter-day LAND attack against Windows Server 2003 and Windows XP SP2: **www.internetnews.com/security/article.php/3488171**

"Ethernet Bridge" (Wikipedia): **en.wikipedia.org/wiki/Ethernet_bridge**

Temper Temper

A $15 USB thermometer and some spare parts are all I needed to control my refrigerator with Linux.

KYLE RANKIN

If loving Linux ever became a crime and I were hauled into court, I think the prosecution's argument would go something like this:

> Your honor, I need to submit only two pieces of evidence to make my case. First, I present Exhibit A: a stack of *Linux Journal* magazines of which the defendant is a columnist.
>
> And your second piece of evidence?
>
> Your honor, the defendant's refrigerator is powered by Linux.
>
> *(The audience gasps.)* Order! I've heard enough! Guilty!

I can't help it. I mean, why *wouldn't* you power your fridge with Linux if you had the chance? In my case, we recently purchased a new fridge for our house, which meant our spare fridge was headed for the garage where I would use it for beer fermentation. Fridges are well insulated, and it seemed ideal for the task at hand, but the problem I ran into was that the built-in thermostat for the fridge would go up to only around 45–50°F at its warmest. To ferment ales, I needed to maintain temperatures between 60–72°F.

I can't help it. I mean, why *wouldn't* you power your fridge with Linux if you had the chance?

When most people convert fridges to ferment beer, they purchase a purpose-built device from their local brew shop. Essentially, you plug your fridge in to the device, plug the device in to the wall, and then set the analog thermostat on the device to your desired temperature. A temperature probe goes into your fridge, and when it gets too warm, the fridge is powered on. These devices range from around $70 to more than $100, depending on whether they are analog or digital, and I almost bought one until I realized I could do the same thing with an old Linux laptop, a couple pieces of hardware and a few scripts.

If you are into home automation at all, you are familiar with the X10 suite of home automation gadgets. Essentially, you can connect lamps and appliances to different X10 gadgets and then power them on with a remote control. There's even a remote control that connects to a serial port, so you can control everything from a computer. Linux has a program called bottlerocket that works great with X10 serial port controllers, and I had used one to control my DSL modem for many years, but that's something for another column.

Figure 1. TEMPer USB Thermometer (photo from the Amazon product page)

The TEMPer USB Thermometer

So, I had a laptop and could control the fridge power, but I still needed a thermometer that worked under Linux and was relatively cheap. I discovered a great little USB-powered thermometer made by a company named TEMPer. It's small, cheap (less than $15 shipped), supports temperatures between –40°C and +120°C, and with a little effort, it works under Linux. It turns out many Linux administrators are using these devices to monitor temperatures in their data centers.

Apparently, the older versions of this thermometer showed up as a USB-to-serial interface; however, the newer models, including the one I bought, show up as a USB Human Interface Device when you plug it in:

```
Apr 16 14:44:33 muriel kernel: [11601.992205] usb 1-1:
 ➥new low speed USB device using uhci_hcd and address 2
Apr 16 14:44:33 muriel kernel: [11602.182910] usb 1-1:
 ➥configuration #1 chosen from 1 choice
Apr 16 14:44:33 muriel kernel: [11602.188481] usb 1-1:
 ➥New USB device found, idVendor=1130, idProduct=660c
Apr 16 14:44:33 muriel kernel: [11602.188529] usb 1-1:
 ➥New USB device strings: Mfr=0, Product=2, SerialNumber=0
Apr 16 14:44:33 muriel kernel: [11602.188563] usb 1-1:
 ➥Product:  PCsensor Temper
Apr 16 14:44:35 muriel kernel: [11604.090148] usbcore:
 ➥registered new interface driver hiddev
Apr 16 14:44:35 muriel kernel: [11604.119323] input:
 ➥PCsensor Temper as /class/input/input7
Apr 16 14:44:35 muriel kernel: [11604.140885] input,hidraw0:
```

```
 ➥USB HID v1.10 Keyboard [ PCsensor Temper]
 ➥on usb-0000:00:07.2-1
Apr 16 14:44:35 muriel kernel: [11604.170151] input:
 ➥PCsensor Temper as /class/input/input8
Apr 16 14:44:35 muriel kernel: [11604.188677] input,hidraw1:
 ➥USB HID v1.10 Device [ PCsensor Temper]
 ➥on usb-0000:00:07.2-1
Apr 16 14:44:35 muriel kernel: [11604.188931] usbcore:
 ➥registered new interface driver usbhid
Apr 16 14:44:35 muriel kernel: [11604.188980] usbhid:
 ➥v2.6:USB HID core driver
```

At first I thought I could get the temperature from this thermometer through some /proc or /sys interface, but unfortunately, the thermometer is more proprietary than that. The Linux community is resourceful though, and a quick search turned up a number of guides on how to pull the temperature from Linux (see Resources for the most helpful guide I found). Essentially, you need to install a few custom Perl modules, including a special one created just for this device that depends on Perl 5.10, so you need a relatively new distribution for this to work (I used the latest stable Debian release).

Install Libraries and Perl Modules

I've always considered Debian as the distribution with the most packaged CPAN modules, but even it didn't have many of the modules I needed for the TEMPer thermometer, so I had to install them from scratch. I'll warn you in advance, this process is a bit tedious, and it reminded me of what it was like to install programs on Linux more than a decade ago. It's amazing how much we take the hard work of package maintainers for granted. At least the first dependencies I had (headers for libusb and a build environment to compile the Perl modules) were available with packages:

```
$ sudo apt-get install libusb-dev build-essentials
```

Next I needed to install a few Perl modules with the cpan program. What you'll find is that many of these modules have their own set of dependencies, so when you are prompted to install dependencies, just tell the cpan program "yes". Also, the first time you run cpan, you might have to go through the initial setup program. If so, just accept the defaults, and you should be fine. Here are the different cpan commands you need to run in order to install the various modules:

```
$ sudo cpan Bundle::CPAN
$ sudo cpan ExtUtils::MakeMaker
$ sudo cpan Inline::MakeMaker
$ sudo cpan Device::USB
$ sudo cpan Device::USB::PCSensor::HidTEMPer
```

Next I downloaded a Perl script from **www.cs.unc.edu/~hays/dev/bash/temper/temper_mon.pl** and made it executable. When run, the script will print the temperature from the thermometer. In my case, I modified it so it output in Fahrenheit instead of Celsius:

```
#!/usr/bin/perl

use 5.010;
use strict;
use warnings;
use Carp;
use Device::USB;
use Device::USB::PCSensor::HidTEMPer::Device;
use Device::USB::PCSensor::HidTEMPer::NTC;
use Device::USB::PCSensor::HidTEMPer::TEMPer;
use lib;
use Device::USB::PCSensor::HidTEMPer;

my $pcsensor  = Device::USB::PCSensor::HidTEMPer->new();
my @devices   = $pcsensor->list_devices();

foreach my $device ( @devices ){
   say $device->internal()->fahrenheit();
}
```

I stored the script in /usr/local/sbin/temper_mon.pl and ran it a few times to make sure it output the correct temperature. Then I connected the thermometer to a USB extension cord that was long enough to reach inside the fridge.

My Custom Fridge Script

The final step in the process was to write a script that would pull the temperature and control the power to my fridge based on whether it was within the proper maximum and minimum temperature ranges I had set. I decided to separate the max and min by two degrees so that the compressor wouldn't kick on too much. I also wanted to write the results to a log so I could monitor how well the fridge maintained the temperature. Plus, I thought it would be cool to `ssh` in to my laptop from anywhere in the world and check on the temperature.

When I first set this up, the weather was cool in the evenings, so I discovered that my fridge would dive down way below the minimum! My solution was to buy a $15 electric heating pad from the drug-store, connect it to another X10 outlet, and put it in the bottom of the fridge. I figured the heat would be gentle enough to maintain the temperature at night without the risks that a proper space heater would have. I set up the script so that it would turn on the heater only if the temperature dipped down one degree below the minimum. I saved

It's amazing how much we take the hard work of package maintainers for granted.

my script in /usr/local/sbin/temper.pl:

```
#!/usr/bin/perl

use 5.010;
use strict;
use warnings;
use Carp;
use Device::USB;
use Device::USB::PCSensor::HidTEMPer::Device;
use Device::USB::PCSensor::HidTEMPer::NTC;
use Device::USB::PCSensor::HidTEMPer::TEMPer;
use lib;
use Device::USB::PCSensor::HidTEMPer;

my $pcsensor   = Device::USB::PCSensor::HidTEMPer->new();
my @devices    = $pcsensor->list_devices();
my $temp_min   = 71;
my $temp_max   = 73;
my $logfile    = '/var/log/temper.log';
my $time       = localtime();
my $temperature;

foreach my $device ( @devices ){
   $temperature = $device->internal()->fahrenheit();
}

open LOG, ">> $logfile" or die "Can't open $logfile: $!\n";

# B4 = Fridge power, B5 = Heater power

# turn on heater if I'm 1F below the low temp
if($temperature < ($temp_min - 1)){
   system('br --port /dev/ttyS0 B5 ON');
   print LOG "$time\t$temperature\tHON\n";
}
elsif($temperature < $temp_min){
   system('br --port /dev/ttyS0 B4 OFF');
   system('br --port /dev/ttyS0 B5 OFF');
   print LOG "$time\t$temperature\tOFF\n";
}
elsif($temperature > $temp_max){
   system('br --port /dev/ttyS0 B4 ON');
   print LOG "$time\t$temperature\tCON\n";
}
else{
   print LOG "$time\t$temperature\t\n";
}

close LOG;
```

Figure 2. A West Coast-Style Red Ale in My Linux-Powered Fridge

The way the logic of the script works, it allows the temperature to drop or rise naturally while the compressor or heater is on, respectively. It changes the power state of my X10 devices with the br command only when the temperature is outside the preset ranges. I set this script to run every minute with cron, and because I log all of the power states, it's easy to watch the temperature float between extremes. I did a bit of tuning at the beginning with the various ranges, and with the current script, the temperature floats between 1°F below the minimum temperature and 1°F above the maximum temperature, which is good enough for me. If I wanted more accuracy, I always could set $min and $max to be closer to each other.

Since the system has been in place, I've been able to maintain the temperature successfully for the first batch of beer I put in the fridge. If you look closely at Figure 2, you can see the little thermometer on the right-hand shelf. Even though my laptop is old, it has plenty of horsepower to spare, so eventually I will graph all of the temperature data and serve it out over Apache. If Bill were here, I'm sure he'd tell me to tweet the temperature.■

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

Resources

Guide to TEMPer Support under Linux: **https://wwwx.cs.unc.edu/~hays/archives/2010/03/entry_25.php**

One of Many Places to Buy the Thermometer On-line: **www.amazon.com/TEMPer-USB-Thermometer-w-Alerts/dp/B002VA813U**

TEMPer Thermometer CPAN Module: **search.cpan.org/dist/Device-USB-PCSensor-HidTEMPer**

DIRK ELMENDORF

Cool Project Potpourri

A look at Caffeine, Homesick and Mockingbird.

I admit that when I signed on to do this column, I was really excited to see Cool Projects on the editorial calendar. It was months away, and I figured by the time the issue rolled around, I would have something big to share with you. Things did not quite work out that way. Because every article I write involves failure in one way or another, I'm going to start off telling you a little about the projects that sat on my bench.

Tiny and Shiny

A very, very tiny handheld computer showed up on my doorstep. It was the completely copyleft Ben NanoNote. The manufacturers are not really sure what to do with it, so they have made it as open as possible to give people a chance to figure it out. I originally got it because I thought it might make a neat portable Linux device for my son. I hoped to get basic video replay on it (if nothing else, to give him something to play with instead of my phone).

Figure 1. Ben NanoNote

In some ways, I had flashbacks to playing with the BeagleBoard, because I quickly realized I was looking at a very neat device, but I didn't have the foggiest idea of how to get started. There is a wiki and several mailing lists, and recently, I found a nice blog post showing how someone ported gnugo over to the device (see Resources). After spending some time researching, I was forced to admit that it was too early for me to jump in. The device has potential, but the project as a whole just is not far enough along for me to participate. Maybe I should run some kind of contest and give it away to a deserving hacker instead of letting it sit on my desk mocking me.

That Other Smartphone

I have been following Android for some time. It seemed like an interesting platform, but in order to take advantage of it, I would have to switch carriers, which for a variety of reasons was not on my to-do list. Then, Google released the Nexus One for AT&T, and I was excited. I have grown increasingly frustrated with my iPhone, and I thought this would be a great opportunity to try Android on great hardware.

So far, it has been great. There have been problems, but I don't end up as frustrated, because most of the problems I have encountered are bugs, which means they can be fixed. On the iPhone, several of my problems were related to running Linux instead of Windows or Mac OS X, which is not a bug in my book. I have a couple Android books sitting on my bookshelf to read (which I realize now are probably out of date, as I got them long before I got the phone). I had hoped to walk through the process of getting an application up and running. Instead, I have settled into the role of end user rather than power programmer, which has been great for my phone experience even if it does not help my writing.

What's Cool?

Now that I've cleared my guilt backlog, the good news is that I have found some cool projects out there—they're just being done by other people, and I get to use them. In all three cases, these are tools I have picked up recently and use them to get stuff done.

Caffeine

Caffeine was one of those wonderful discoveries, because it brought to light a problem I did not even know I had, and it originally was demonstrated on a Mac. Let's say you're watching a video. After a while, you get so engrossed in the clip that you don't move your mouse or touch your keyboard, because you actually are watching it with your full attention. Then, your screensaver kicks in (as a bonus, you have a password on it). Or, say you're giving a presentation and you stay on a slide a while so you can answer a bunch of questions, and the screen blacks out to save power. In the

past, there were two ways to handle this. You could just remember to flick your mouse every so often (which is annoying and error-prone), or you turn off the screensaver, which means you have to remember to turn it back on later.

Figure 2. Caffeine

Enter Caffeine. This handy app puts a small coffee cup in your notification bar. When you want your computer to stay awake, just click the coffee cup. If you're okay with normal behavior, click on the cup again. This is nice because it is easy to access, and it gives you a visual cue to remind you what you have done. There even are options to tell it to kick in for a certain amount of time. If clicking is too much trouble, you also can tell it to kick in automatically when certain applications are running (Flash videos, OpenOffice.org Presentation or Skype, for example).

On Ubuntu, you can add in the PPA (Personal Package Archive). Then, installation is easy:

```
sudo add-apt-repository ppa:caffeine-developers/ppa
sudo apt-get update
sudo apt-get install caffeine
```

It ends up being very handy. As I mentioned, versions of Caffeine exist for other platforms, so I end up recommending it whenever I see people give presentations and their screens go dark. Now, it's a standard part of my installation for my workstations.

Homesick

Homesick proves that sometimes it pays to wait and spend time on Google. When I started writing this column, I had an idea to write about my struggle to keep my home directory synced across a lot of different workstations. I figured it was a common problem, but I couldn't find an elegant solution. I originally hoped to drag in cloud storage, so I didn't have to access my home computer directly to stay in sync.

I spent a lot of time searching and didn't find what I was looking for, so I started to sketch up some ideas of how to solve the problem. That's when I realized something very important about writing a column: there always is a deadline looming. I ended up shelving the idea of starting a brand-new project and focused on writing about something I actually could finish in time to talk about. In that case, it was a BeagleBoard.

As I was working on a project last week, I stumbled into Homesick. It was designed to solve this exact problem. It introduced the idea of "castles", which are different repositories of dot files stored in git repositories. You then can choose to pull down a variety of different castles on each machine and choose which ones you actually want to use.

In order to use it, you need to have Ruby, RubyGems and git installed. Once those are installed, simply install the homesick gem:

```
sudo gem install homesick
```

A castle can be pulled in by referencing a local folder, a remote git repository or by using the short form if the repository is on GitHub. The example provided here is from Homesick's author's GitHub site:

```
homesick clone technicalpickles/pickled-vim
```

This pulls down the vim files from technicalpickles. It does not replace your existing vim dot files (yet).

In my case, I'd like to build my own vim castle (you do not have to keep the files in a git repository, although it doesn't hurt):

```
cd ~/Desktop
mkdir -p delmendo-vim/home
cd delmendo-vim
cp -R ~/.vimrc ~/.viminfo ~/.vim home/
homesick clone ~/Desktop/delmendo-vim
```

Now you can ask which castles are installed. In my case, it shows technicalpickles and delmendo-vim. It should list technicalpickles/pickled-vim (a bug that should be fixed by the time this goes to print).

To activate it, execute the command:

```
homesick symlink technicalpickles/pickled-vim
```

or:

```
homesick clone delmendo-vim
```

This starts the process of linking in the files. It also warns you of conflicts and allows you to look at differences before you make the swap. The system does not handle synchronizing the files across different machines, but you easily can handle that either by putting them in a network-accessible git repository or host the files remotely and mount them. There are a bunch of ways to do both, but GitHub and Ubuntu One come to mind.

The first commit on the project was March 3, 2010, so it's still early. I hope the features continue to evolve, because it saves me from having to cobble it together.

Mockingbird

To be honest, I have absolutely no artistic talent. This is generally not a problem, because I partner with a great designer (who happens to be my brother, so he doesn't take it personally when I yell at him). I'm in the process of launching a Web application. As you probably can tell

from my writing, I'm very comfortable describing things with words, but sometimes words just don't cut it.

I tried a lot of different software tools, but none of them worked. Some failed because they required too much artistic talent on my part. I always end up feeling like they're designed for designers, which I'm not. In other cases, I hit a wall with adoption, because I work with people who do not use Linux, which means I have to select from tools that are completely cross-platform. In the past, I resorted to scribbling things on paper so I had some kind of diagram to discuss. Paper was a great medium when everyone on my team sat in one room, but now that I work with a distributed team, I end up losing the immediacy of the paper drawing. It also became a pain to manage scanning and distributing paper documents.

That's when Mockingbird got dropped in my lap (meaning my brother sent me a link). Mockingbird is a Web-based wire framing tool. Wire framing is where you just sketch the outline of a design so that you get the idea of what is happening. It does not provide every last detail. Mockingbird allows you to design a basic interface and then easily share the URL of your design with others, or you can export it as a PNG or PDF. The tool ends up being awesome for me, as it provides a bunch of template items that make it easy to grab the interface pieces I need. If I need to change something, I simply drag it around and make quick changes on the fly. Because it is Web-based, my teammates can use it on their OS of choice.

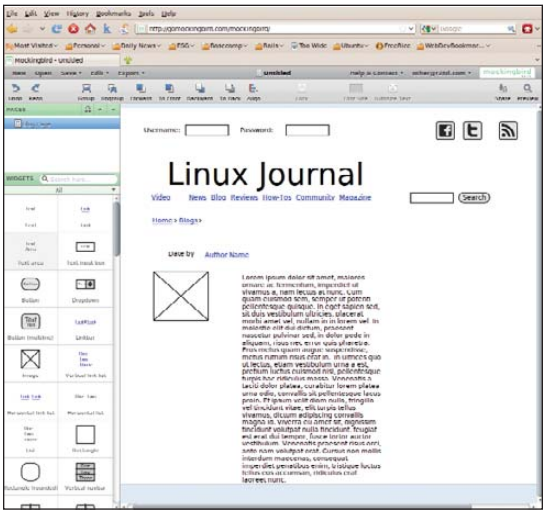


Figure 3. Mockingbird

The first time I used it, I was hooked, but I was a little nervous. It seemed so interactive, I dreaded to find out what custom plugin was required to make it all work. I quickly learned it was built using Objective J. I was really worried this was some new form of Java, but it turns out to be JavaScript.

Mockingbird is built using a framework called Cappuccino. This LGPL'd framework is built in JavaScript, but modeled after GNUstep and Apple's Cocoa framework (hence the name Objective J inspired by the Objective C of those frameworks). Cappuccino makes a distinction between itself and libraries like jQuery by saying that those are about making a dynamic Web site. Cappuccino is useful only if you are building a full-blown application. The other toolkit it compares itself to is SproutCore. Cappuccino is different because it is written entirely in JavaScript and does not require knowledge of HTML or CSS.

A word of warning: Mockingbird is not open source. It currently is free during the "beta". At the time of this writing, I don't know when the beta will end or what the pricing will be like. I admit that I have spent more time drawing with Mockingbird than looking at Cappuccino (which is open source). I was so impressed with the way you can interact with the drawings (like being able to select interface items, copy, go to a different page in the app and so on) that I realized how much I need to look at some of the "application" frameworks for JavaScript. The last time I looked at them was several years ago when I played with ExtJS. It provided a lot of whiz-bang GUI elements, but it proved to be too difficult to work with. Maybe that has gotten better in the interim.

Conclusion

I've realized that for me, "cool" has a lot more to do with usage than fashion. Sure, I thought it was an impressive proof of concept that someone got Android onto the iPhone, but I'm much more impressed when I discover something that changes how I work every day. These tools certainly have done that for me.■

Dirk Elmendorf is cofounder of Rackspace, some-time home-brewer, longtime Linux advocate and even longer-time programmer.

Resources

Copyleft Ben NanoNote: **en.qi-hardware.com/wiki/Main_Page**

"My first port to the Ben NanoNote: gnugo": **www.mostlymaths.net/2010/04/my-first-port-to-ben-nanonote-gnugo.html**

Caffeine: **https://launchpad.net/caffeine**

Homesick: **github.com/technicalpickles/homesick**

Mockingbird: **gomockingbird.com**

ExtJS: **www.extjs.com**

Linux on the iPhone: **linuxoniphone.blogspot.com/2010/04/ive-been-working-on-this-quietly-in.html**

19th USENIX SECURITY SYMPOSIUM

Washington, DC • August 11–13, 2010

The USENIX Security Symposium brings together researchers, practitioners, system administrators, system programmers, and others interested in the latest advances in the security of computer systems and networks.

The 3-day program includes:

Keynote Address by:

- Roger G. Johnston, *Vulnerability Assessment Team, Argonne National Laboratory*

Refereed Papers:

- Refereed paper presentations showcasing new research in a variety of subject areas including cryptography, Internet security, dissecting bugs, and more

Invited Talks by Industry Experts such as:

- "Understanding Scam Victims: Seven Principles for Systems Security," by Frank Stajano, *Senior Lecturer at the University of Cambridge, UK,* and Paul Wilson, *The Real Hustle*
- "End-to-End Arguments: The Internet and Beyond," by David P. Reed, *MIT Media Laboratory*
- Plus a Poster Session, BoFs, and more

Stay Connected...

http://www.usenix.org/facebook/sec10

http://twitter.com/USENIXSecurity

Co-Located Workshops:

EVT/WOTE '10: 2010 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections
August 9–10, 2010

CSET '10: 3rd Workshop on Cyber Security Experimentation and Test
August 9, 2010

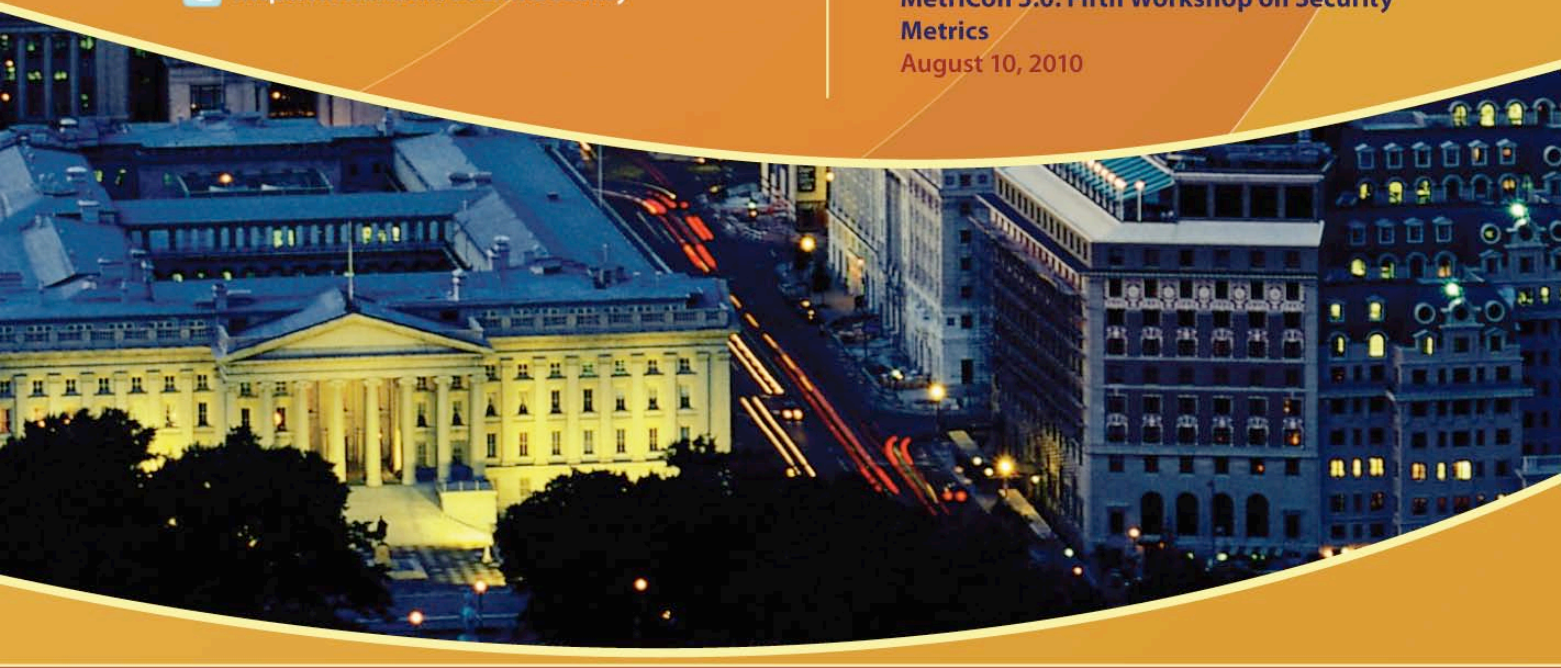
WOOT '10: 4th USENIX Workshop on Offensive Technologies
August 9, 2010

CollSec '10: 2010 Workshop on Collaborative Methods for Security and Privacy
August 10, 2010

HealthSec '10: 1st USENIX Workshop on Health Security and Privacy
August 10, 2010

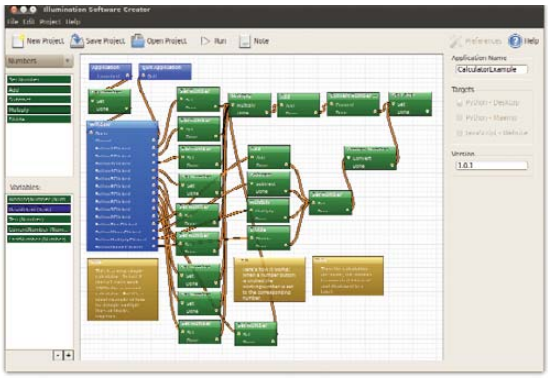
HotSec '10: 5th USENIX Workshop on Hot Topics in Security
August 10, 2010

MetriCon 5.0: Fifth Workshop on Security Metrics
August 10, 2010



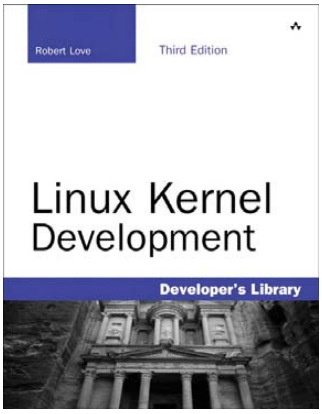
Register by July 19, 2010, and save! **www.usenix.org/sec10/lj**

Radical Breeze's Illumination Software Creator



Were you a LEGO-lover as a kid? If so, Radical Breeze thinks you'll find its new Illumination Software Creator 1.0, a visual software application builder, extremely intuitive. A user—even a non-programmer—can easily experiment with software building blocks, in a completely visual and understandable way, and create software that can be run on Linux, Mac OS X and Windows computers. The application converts visual concepts into source code. No virtual machine is involved. In addition, software built with Illumination Software Creator can be run as a rich Internet Web page or an Internet tablet application (for Nokia Internet Tablets). The languages utilized are Python and GTK for desktop and mobile applications, and JavaScript for Web applications.

www.radicalbreeze.com



Robert Love's *Linux Kernel Development*, 3rd Ed. (Addison-Wesley)

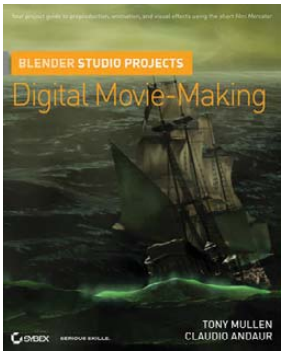
Our old friend Robert Love, whose popular Kernel Korner column long graced these pages, has updated his classic book *Linux Kernel Development* to a new 3rd edition. Published by Addison-Wesley, this work details the design and implementation of the Linux kernel, presenting the content in a manner that is beneficial for those writing and developing kernel code, as well as for programmers seeking to better understand the operating system and become more efficient and productive in their coding. The book details the kernel's major subsystems and features, including its design, implementation and interfaces. It covers the Linux kernel with both a practical and theoretical eye, which is intended to appeal to readers with a variety of interests and needs. Updated content includes an all-new chapter on kernel data structures, details on interrupt handlers and bottom halves, extended coverage of virtual memory and memory allocation and more.

www.informit.com

CloudLinux's LVE Wrappers

Hosting service providers may want to read on about the new LVE Wrappers from CloudLinux, tools that let the server owner control the exact CPU usage for individual users and applications. LVE Wrappers are based on CloudLinux's Lightweight Virtual Environment technology that allows administrators to control CPU usage on a server at the tenant or application level. It does so by isolating specific hardware resources in a lightweight container and prevents one tenant on a shared server from affecting others. LVE Wrappers, which start individual applications and dæmons inside LVE environments, allow for control of resources for each application, which allows for greater flexibility and stability on the overall server infrastructure. The software is available to all CloudLinux subscribers.

www.cloudlinux.com



Tony Mullen and Claudio Andaur's *Blender Studio Projects: Digital Movie-Making* (Sybex)

Authors Tony Mullen and Claudio Andaur have set out to prove that the amazing open-source 3-D animation software Blender isn't just for hobbying anymore. Their new book *Blender Studio Projects: Digital Movie-Making*, now out from Sybex, shows readers how to do on Blender what one typically has done with high-end (and high-priced) apps Maya and 3ds Max. Sybex bills the book as "a real-world, roll-up-your-sleeves guide that plunges straight into step-by-step instructions designed to help you build skills and create solid assets for film, video and games." The companion DVD includes starter, intermediate and final files, as well as movie files to help you every step of the way.

www.sybex.com

CAST Software's Analysis and Measurement Software for SAP, Siebel and PeopleSoft

Managing ERP apps can require the patience of Job, which is why CAST Software released its new Analysis and Measurement Software for SAP, Siebel and PeopleSoft. The product's mission is to bring consistent and objective measurement of these platforms and eliminate costly customization errors in critical business applications. CAST's own studies of ERP installations showed a large percentage of issues that normally go undetected in functional testing and result in serious performance, stability and maintainability problems. For instance, 80% of database interactions in ERP installs are handled improperly, leading to significant business disruption. Environments like SAP, Siebel and PeopleSoft, says CAST, require a deceptively high level of customization. Because manual measurement of quality is expensive and time consuming, automation is useful for exposing flaws that cause applications to perform erratically.

www.castsoftware.com

Bricsys' Bricscad

The "team" of Linux-based CAD programs just added a new star player to its roster, namely Bricscad V10 from Bricsys NV, a high-end DWG-based CAD platform that previously was only for Windows. Bricsys calls Bricscad V10 "the most application-friendly CAD platform in the industry", thanks in part to DCL and LISP APIs that allow existing applications and customizations written for Windows-based Bricscad and/or AutoCAD to run without modification. Besides essential CAD functions for users in GIS, AEC, mechanical CAD and civil engineering, additional core product benefits include a recognizable interface, comprehensive support and reasonable price points. Initially, Bricscad will support Red Hat and Ubuntu Linux.

www.bricsys.com

RiverMuse Pro

The developers at RiverMuse have given their IT operations management platform RiverMuse Pro its vitamins for a stronger, more robust offering. Now in version 2.0, RiverMuse Pro combines the power of a robust Manager of Managers (MoM) functionality—centralizing event collection from other management systems as well as from the infrastructure—with real-time event correlation and analysis (ECA) capabilities to detect and alert on business-impacting incidents. RiverMuse says that the platform's architecture "offers a number of disruptive innovations that are critical to managing dynamic and virtualized environments that are common in IT environments today." The new version 2 focuses on mid-market service providers and enterprises, enabling them to assure delivery of dynamic IT services through advanced event capture, correlation and alerting. A free and open-source RiverMuse Core also is available.

www.rivermuse.com

CodeWeavers' CrossOver Games

Our zany friends over at CrossOver were on the verge of mutiny since the grand poobah boss, Jeremy White, scheduled the release of CrossOver Games 9.0 just when *Iron Man 2* was hitting the theaters. We're told that in order to ensure programmers continued working weekends without break, White created a makeshift "electric whip" and paced around the office screaming "full Steam ahead, minions!" in a mediocre Russian accent. The intimidation apparently worked, because the team pushed out both Linux and Mac versions of CrossOver Games, which allows one to play Windows-based games on these platforms. Version 9.0 supports the new Steam UI, *StarCraft 2* (beta) and *StarTrek Online*, as well as enables users to install games from a single screen and a single click on CodeWeavers' compatibility center. Furthermore, users who figure out how to use CrossOver to install a Windows-compatible game can upload the installation recipe to the company's database.

www.codeweavers.com

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

Fresh from the Labs

Tiny Ear Trainer—Color-Based Interval Training

29a.ch/tinyeartrainer

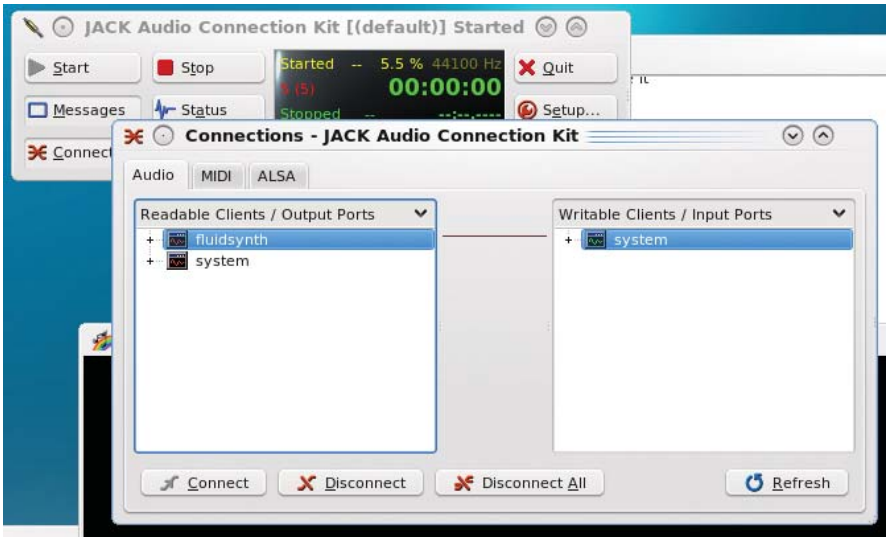
An ongoing fascination of mine has been with Synaesthesia: the crossing of senses, often leading to new and unique perceptions in the mind of the "Synaesthete". This is particularly common among musicians (myself included) where individuals will "see" sound as color, possibly along with shapes, numbers, textures and so on. It seems that Jonas Wagner has the concept of linked sound and color quite firmly embedded in his own thoughts when it comes to his program Tiny Ear Trainer—an ear training program designed to teach you musical intervals with color association.

To quote the Web site: "Tiny Ear Trainer is a tiny piece of software that helps you recognize musical intervals. Tiny Ear Trainer is free/open-source software written for the GNU/Linux operating system." According to the Web site, its features include:

- Associates colors to intervals.
- Features learning mode, which plays intervals together with color and name.
- Plays harmonic or melodic intervals.
- Uses fluidsynth/soundfonts for playback.
- Uses Python/GTK.
- Is tiny!

Installation i386-based Debian and Ubuntu users are in luck as a .deb file is

Tiny Ear Trainer takes the unique approach of teaching musical intervals through color association.



Making Tiny Ear Trainer work with JACK can be tricky for a first-timer, but all you need to do is connect "fluidsynth" with "system", and you're away.

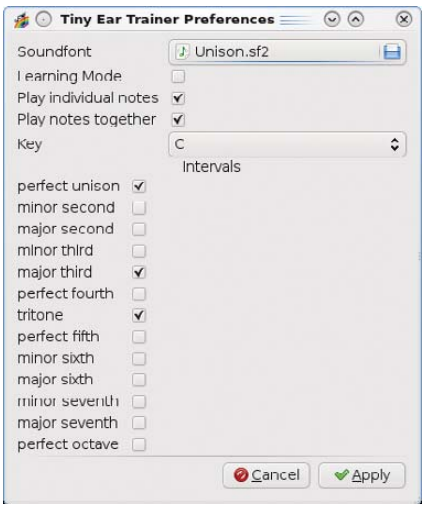
available at the Web site. For those using other systems, the obligatory source tarball has been provided, but thankfully, the library requirements are fairly modest and don't require anything particularly obscure for installation. The documentation says you need the following:

- Python 2.6 (2.5 *might* work too and will require simplejson).
- PyGTK >= 2.10.
- fluidsynth.
- A soundfont (fluid-soundfont-gm).

Once you have the needed libraries, download the latest tarball, extract it, and open either a terminal or a file manager inside the new folder. For those with a modern file manager like Nautlius or Dolphin, you can install Tiny Ear Trainer simply by clicking on the file install.sh, where you'll be prompted to enter a root password. For those with a minimalist system, or those who prefer the manual method, you can install it by entering the following command either as root or by using sudo:

```
# python setup.py install
```

Once the installation has finished, Tiny Ear Trainer should be in your system menu,



Tiny Ear Trainer gives you a number of useful options from which to choose (including a learning mode), all in one concise package.

or you can run it by entering:

```
$ tinyeartrainer
```

Usage Before going any further, I must warn you that Tiny Ear Trainer uses JACK. That's not a bad thing, but it can be tricky getting it to work for the uninitiated (experienced JACK users will have no trouble at all and can skip the next few paragraphs).

First, start JACK. Most people will be using the JACK Audio Connection Kit,

which usually can be found in your menu as JACK Control, or you can run it from a terminal with the command `qjackctl`.

Now, click Start inside the GUI, and if you're lucky, it will start straightaway. But, I wasn't lucky. This was the first time I'd used JACK with this distro, and it came up with an error that I didn't understand until I looked further into it. You may encounter this too. JACK usually is set up by default to run in real time, generally coupled with a real-time kernel. Real-time kernels crash constantly on my hardware, so that wasn't an option for me. In this case, my only real choice was to disable real time in the Settings tab in the Setup option.

Now that JACK is running, start Tiny Ear Trainer, and in the JACK Audio Connection Kit, click Connect on the bottom left, and open the Audio tab. In the left pane under Readable Clients/Output Ports is the client fluidsynth, and in the right pane under Writeable Clients/Input Ports is the client system. Click on fluidsynth on the left and system on the right, and then click the Connect button on the bottom left, and you should be good to go.

Back over to Tiny Ear Trainer—the program will be sitting idly with a blank screen (presumably so you can connect to JACK in the meantime) until you click the Play button. When you do, several notes will play in a randomly chosen interval (for instance, a minor third), and when the notes are over, it will tell you what was played along with a colored background. This builds your association between sound and color, as well as trains your ear to intervals.

Now, let's take a look at the Preferences screen. Two default soundfonts usually are provided for you under /usr/share/sounds/sf2, but if you don't like these soundfonts, don't have any or want to change the instrument (a guitar springs to mind), you can change the soundfont with the Soundfont option at the top. The next option, Learning Mode, is equally important. Learning Mode plays the sound with the color at the same time (instead of a black screen), so you can learn it now and test yourself later.

From here onward, the options start getting musical. You can choose whether notes are played together, which intervals to play from a long list, as well as the all important key in which they'll be played (with the default being the obvious C). I recommend starting off with Learning Mode with only several intervals selected, testing yourself on these, and then gradually adding more to the list.

I'm not sure if Synaesthesia is what Jonas really had in mind, or if he just happened to strike upon the same theme, but this is the first time I've come across what Synaesthetes always have seen in their minds being used practically. Of course, it all could backfire as some other Synaesthete shouts, "No, no, a major third is light green, damn you!", but personally, I look forward to the inevitable color arguments with glee.

Sunflower—Twin-Pane File Management

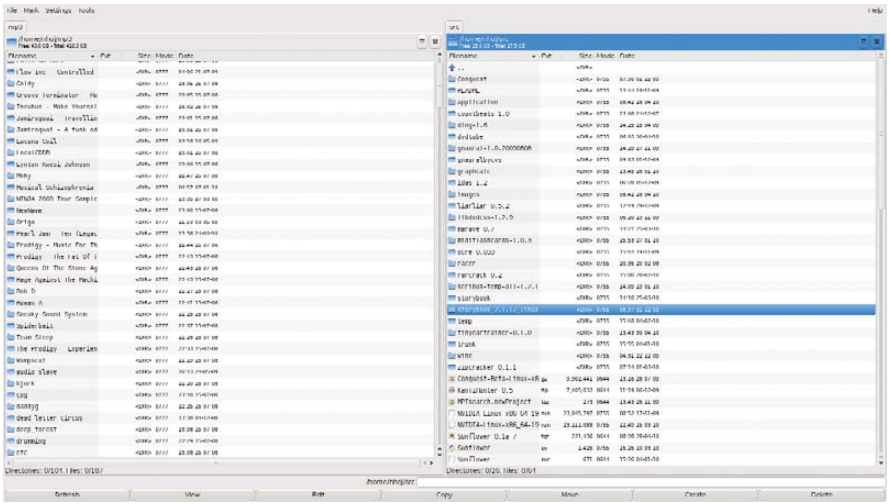
rcf-group.com

Anyone chasing a lightweight file manager is pretty spoiled for choice when it comes to Linux. However, lightweight usually brings along with it some nasty compromises—perhaps an awkward interface, hideous grayness, bad design or a total lack of aesthetics. Not so with Sunflower. To quote the Web site:

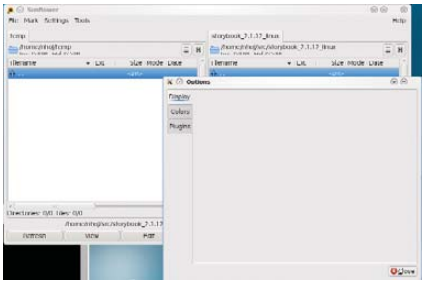
> Sunflower is an open-source, small and highly customizable twin-panel file manager for Linux. It supports plugins. It is possible to run this application on other systems (Mac OS, Windows), but you will have to install necessary libraries (Python, GTK 2.0+ and PyGTK).
>
> Currently, this program is still in the heavy development phase and is open for testing. I plan on releasing versions often during development.

Installation Installing Sunflower actually went off without a hitch. I just downloaded it and it worked. The hardest part was finding it on the Web site, but that shouldn't be hard for smart *LJ* readers! Head to the Download Section, click on Sunflower, and at the bottom is a download link where you can grab the latest tarball. Download and extract the tarball, and open a terminal where you extracted the contents.

As far as library requirements go, I didn't have to install anything, but anyone with a minimalist system may have to. As



Keyboard-driven twin-pane file management in glorious full screen—it's lovely!



As these blank options show, Sunflower is at a rather early stage in life.

stated above, you'll need Python, GTK 2.0+ and PyGTK. Once you're ready to go, run Sunflower by entering:

```
$ ./Sunflower.py
```

Usage Inside the Sunflower window, things feel like a cross between Norton Commander (hereby referred to as NC, along with its subsequent clones) and newer file managers, such as Nautilus, Konqueror and so on. Now I must say that this definitely is early alpha stuff, and as such, a great deal of features that most people take for granted are still missing, which is evident when you click on Tools, only to be greeted with nothing.

This current sparseness extends to the Options list where every tab also is empty (at least it was at the time of this writing). As a result of this, I recommend clicking on Settings→Show command bar, so basic functions, such as copying, moving, editing and others, at least will be provided for you at the bottom of the screen.

Nevertheless, don't let what I've just said put you off, because Sunflower has one supreme strength: it's delightfully keyboard-driven! It takes me back to the days of lazily browsing around inside the NC with only the arrow and Enter keys, without all of the comparatively headache-inducing GUI elements that were about to come, competing for your attention and getting in the way. As the keyboard is so essential to Sunflower, Here's a guide for your reference:

- Ctrl-T: duplicate tab.
- Ctrl-W: close tab.
- Ctrl-Tab: next tab.
- Ctrl-Shift-Tab: previous tab.
- Ctrl-Z: open terminal tab.
- Ctrl-R: reload list.
- Backspace: go to parent directory.
- F4: open text editor.
- Alt-Letter: quick search.
- Menu: open file/directory menu.
- Ctrl-Menu: open with menu.
- F7: create directory.
- F8/Delete: delete selection.
- F11: full screen.
- Ctrl-F7: create empty file.

One of the first things you should try is pressing F11, which makes Sunflower go full screen, giving you that old NC feel but with modern GUI elements. You can open files and folders by double-clicking, but I recommend skipping the mouse entirely and just using the keyboard, with the well-known combo of the arrow keys for navigating up and down and Enter for opening or executing something.

Pressing Tab changes between the left and right panes, but unlike NC and its clones, each pane can have another tab added onto it, bringing this style of design ethic firmly into the 21st century. Something that seems to be missing in the Midnight Commander that I took for granted in NC is the ability to jump between the starting letters of filenames with two keystrokes (this saved untold amounts of time). Thankfully, this is featured early on and works well. Just press and hold Alt followed by whichever letter you're searching for, and you quickly can navigate inside huge lists of filenames with ease.

Sunflower may be a very early alpha, but it already feels pretty solid in parts and looks very promising. My apologies go out to developer MeanEYE for my constant Norton Commander references, but this old design ethic features here so prominently, and it's refreshing not only to see and feel it again, but also to have it re-interpreted in a way that fits 21st-century computing habits so well. Sunflower seems bent on keeping only what is necessary. And for those who keep stripping away, often to the extent where features and functionality are laughable, Sunflower retains what is genuinely useful over the last decade of computing, minus the bulk. Although this early Sunflower currently is far from what most people will be expecting from a file manager, as time goes by, expected elements (especially more mouse functionality) will make their way into the interface, and the beautiful design ethic behind Sunflower will really shine.■

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

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

A Simple Approach to CHARACTER DRIVERS in USER SPACE

The BaseBoard4 from Demand Peripherals can contain different combinations of 25 different character devices, all multiplexed on to a single USB-serial link. Its drivers, described here, show how writing drivers in user space can get a complex device up and running quickly.

BOB SMITH

Demand Peripherals, Inc., makes an FPGA-based robot controller that gives a robot or other industrial control systems the high I/O pin count and precise timing that a Linux laptop or single-board computer alone cannot offer. The company has built more than 25 different FPGA-defined peripherals for the controller, and it wanted to offer Linux device drivers for all of them.

Doing 25 drivers in the kernel, although possible, would have required time and effort far beyond what the company could afford. The process of building kernel device drivers would have been even more complicated because the FPGA card connects to the Linux host over a USB-serial link. The solution, illustrated in Figure 1, is to have a dæmon manage the USB-serial port and demultiplex the various FPGA-based peripherals out to their own device nodes. The device nodes are little more than shims that let the high-level application deal with separate device entries for each peripheral.

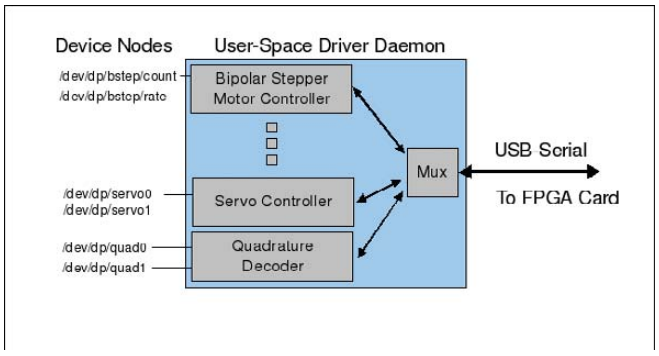


Figure 1. Example of a User-Space Device Driver

The customer selects the mix of peripherals to be loaded into the FPGA. Figure 2 shows a BaseBoard4 with some cards that demonstrate what might be a fairly common peripheral mix. The system pictured has eight peripherals, including a four-channel servo controller, a dual H-bridge controller, a quad interface for the Parallax Ping))) range sensor, a RAM-based pattern generator (driving the data and clock lines going to a 48-bit shift register that connects directly to the LCD), a unipolar stepper motor controller, a bipolar stepper motor controller, a quad event or frequency counter (connected to a single Parallax light-to-frequency sensor), and a dual quadrature decoder. Schematics for all of these demo cards

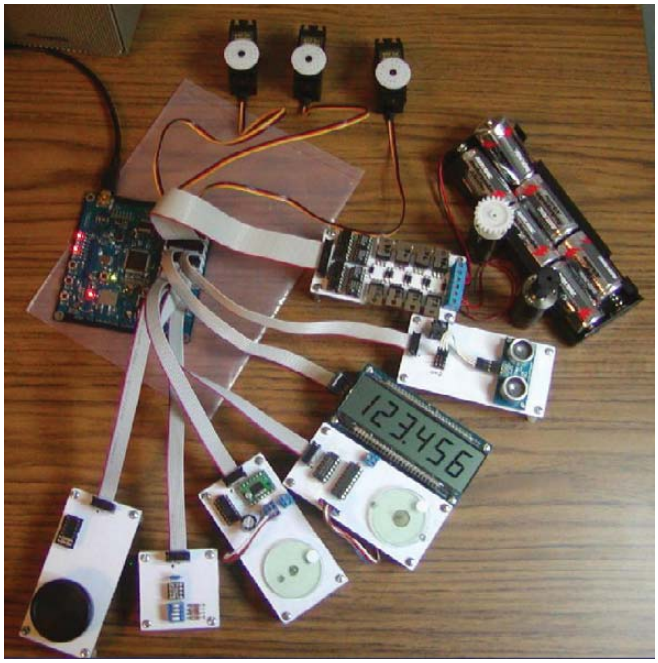


Figure 2. Robot Peripherals, All with Linux Drivers

are on the Demand Peripherals Web site.

All of the peripherals shown in Figure 2 can be configured and controlled using device nodes in the /dev directory. The following Bash commands, for example, might be part of the higher-level control software for the system pictured:

```
# Feed wheel quadrature counts to a motor control program
cat /dev/dp/quad0 | my_motor_pgm &
# Feed the same quadrature counts to a navigation program
cat /dev/dp/quad0 | my_navi_pgm &
# Set a stepper motor step rate to 1000
echo "1000" > /dev/dp/bstep1/rate
# Now step 300 steps
echo "300" > /dev/dp/bstep1/count
# Monitor distance reported by a Parallax Ping)))
cat /dev/dp/ping0/dist &
# Set a servo pulse width to 1.5 ms (1500000 ns)
echo "1500000" > /dev/servo/servo4
```

USE CASES

The above commands illustrate two of three important use cases for the user-space drivers: sensor broadcast and driver configuration. The third use case is bidirectional transfer.

The first use case is sensor broadcast, and in the example above, it's actually multicast of sensor data. Did you know that the /dev/input drivers implement a multicast mechanism? Multiple readers get identical copies of the events that come from the input devices. There is a simple experiment you can do to demonstrate this. Press Ctrl-Alt-F2 (to go to a different console), log in, and run the command `sudo cat /dev/input/mice | od -b`. Do the same for another console (for example, Ctrl-Alt-F3). Now, move the mouse a little and switch between the F2 and F3 consoles. They both display the same thing, don't they? What a shame that Linux does not

FOR ROBOTICS, THE ABILITY TO FAN A SENSOR READING OUT TO SEVERAL PROCESSES IS PARTICULARLY IMPORTANT.

have some generic way to do multicast like that of the /dev/input subsystem.

For robotics, the ability to fan a sensor reading out to several processes is particularly important. For example, a quadrature encoder attached to a wheel needs to be seen by both the motor controller software and by the navigation software. The motor controller might need to know if the wheel is turning to know whether the motor is stalled, and the navigation software might count the wheel revolutions to compute the robot's current location.

The second use case is peripheral or driver configuration. DC motor controllers need to know the frequency of the PWM pulses. Stepper motors need to know the step rate, and the SPI (Serial Peripheral Interface) ports need to be told the clock frequency and the mode of operation. Either an ioctl() call or a sysfs-style interface can be used for driver configuration.

Configuration interfaces can be a little tricky, in that the information is often not a simple stream of bytes—it may encompass several different pieces of information. An ioctl() interface typically passes a data structure for complex configurations, while a sysfs interface might use a space-separated list of ASCII-encoded values. Demand Peripherals uses the ASCII-encoded numbers approach, because the overhead of decoding and parsing a line of text is not too onerous given the relative infrequency of driver configuration. Also, being able to `cat` a sysfs type file to see the driver configuration is kind of handy.

The third use case, bidirectional transfer, is really the most common use case. You probably are already familiar with serial ports, the most common example of bidirectional I/O. Although none are included in the examples above, the FPGA-based robot controller needs bidirectional I/O for peripherals that transparently pass data from one end to the other. These include both FPGA-defined serial ports and SPI ports. You may prefer, as we did, to be able to do block reads and writes until both sides of the interface are open.

REQUIREMENTS FOR USER-SPACE DRIVERS

Our number one requirement for this project was to spend as little programmer time as possible on it. This meant minimizing the number of lines of code to be written and avoiding modifying someone else's poorly or completely undocumented code. This requirement also implied that we not try to hide our interfaces in an application library. Because a library is part of the higher-level control application, you still would need a dæmon, still need some common IPC mechanism, and still need to document the internal and the external interfaces. The other problem with a library approach is that it is usually not just one library; you may need to write a library, or binding, for every programming language you want to support. Using a real character device instead of a library means your customers can program in any language they want, not just the ones for which you've written a binding.

The second requirement was that the driver security model be based on file permissions. This implied that all of the device data and configuration interfaces should be visible in the filesystem. That is, you should to be able to do a `chmod 644` on something like /dev/dp/bstep1/rate. Using named pipes and FUSE (Filesystem in USErspace) could have fulfilled this requirement. Doing this using pseudo-terminals would have been tricky.

Another requirement is that select() works both in the higher-level control application and in the user-space driver itself. This requirement comes about because select() is so much faster than threads in most applications. Embedded systems, such as robotic or other industrial control systems, often run on the cheapest, lowest cost hardware possible, and, in the case of robots, often on battery power. These constraints lead embedded Linux programmers to prefer select()-based systems.

FUSE often is suggested as a way to implement character drivers, but I was unable to get select() to work on both sides of a FUSE interface. I like FUSE; it can solve a lot of user-space driver problems, but it seems unfair to me to ask FUSE, a filesystem, to double as a character driver. After all, who would expect ext3 or other kernel filesystems to have built-in character drivers?

The last requirement was that writers block until a reader is present. Both named pipes and pseudo-ttys allow the writer to write 4KB before blocking. It was important to us that the driver not fill a buffer with stale data that a higher-level robotic application must discard to get to the current data.

A SIMPLE APPROACH TO USER-SPACE DRIVERS

In the end, we didn't find any existing Linux facilities that satisfied all of our requirements and use cases. However, we were able to find or create two relatively simple device drivers that could. Figure 3 illustrates the basic idea.

The idea is to have two very thin drivers that sit between the higher-level applications and the user-space driver. These are real drivers and appear as such to the higher-level software. The data exchanged between the application and the user-space driver passes as transparently as possible through the kernel. Even flow control passes transparently between the application and the user-space driver.

The first use case, that of multicasting sensor data, is solved by the "fanout" driver described in detail at **www.linuxtoys.org**. Demand Peripherals uses fanout devices for quadrature decoders,

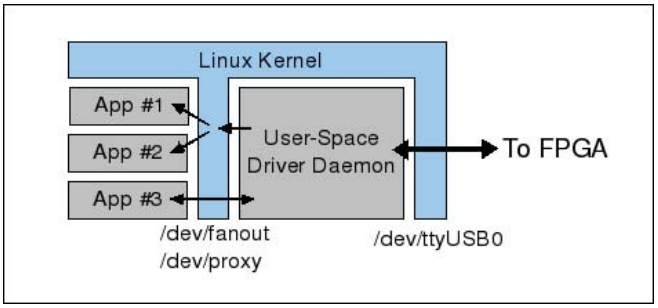


Figure 3. Two New Drivers Link Applications to Driver Dæmons

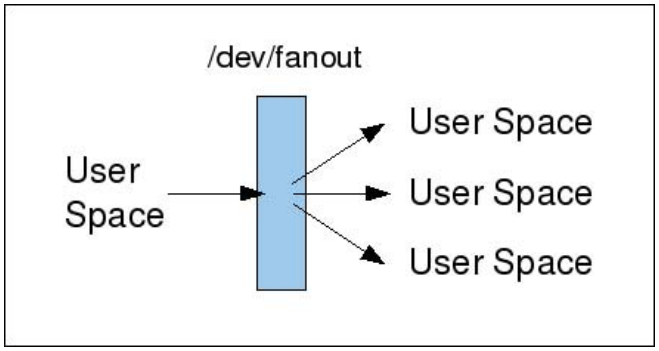


Figure 4. A Simple Multicast Device

IR receivers, ultrasonic range sensors, PlayStation controller interfaces, event counters and all other continuously sampled sensors. Figure 4 shows the basic data flow in a fanout device.

You can skip further down in this article to get and install fanout, or you can continue reading and come back to try the examples. Once you've installed fanout and created the device nodes for it, you can test it with a few simple commands:

```
cat /dev/fanout &
cat /dev/fanout &
cat /dev/fanout &
echo "Hello World" > /dev/fanout
```

The message appears three times, as you'd expect. Fanout is like /dev/input in that it protects the writer, not the reader. If a reader does not keep up, the reader gets the error, allowing the writer and other readers to continue unimpeded.

For data flowing in the opposite direction, you need something like a "fan-in" device—that is, something that protects the reader. A named pipe works reasonably well for this.

The low-speed nature of driver configuration, the second use case, makes possible several approaches. The approach we took was to write a driver, called proxy, that solved both the configuration use case as well as the bidirectional transfer use case. The two defining features of proxy are that one side cannot write until the other side is open for reading, and that a write of zero bytes is passed through the driver and seen as a read of zero bytes at the other end. The usefulness of the second feature is best shown by an example. Consider the case of a user reading the current value of a configuration parameter:

```
cat /dev/dp/bstep/rate
```

/dev/dp/bstep/rate is a proxy device, and the user-space driver dæmon on the other side of it would see that a write is possible when `cat` opens the device. The dæmon writes the

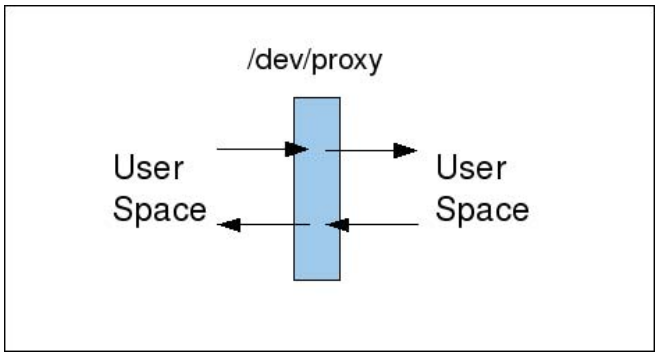


Figure 5. A Device to Pass Data between Two Applications

current value and then does a write of zero bytes (both of which are read/seen by cat). It is the write of zero bytes that tells cat it is done and can exit.

One drawback to the proxy driver is that it required the customer to build and install a kernel module. Although Dynamic Kernel Module Support can help, a lot of customers are intimidated by this even if it's not that difficult [see "Exploring Dynamic Kernel Module Support (DKMS)", *LJ*, September 2003: **www.linuxjournal.com/article/6896**]. FUSE is a good approach for customers who do not want to deal with building modules and who don't need select() support.

Another approach for driver configuration is to use a regular filesystem file and the inotify facility to alert the user-space driver that a new configuration is being requested. You get two nice features by keeping the configuration files on a volume with persistent storage. The first is that you don't need to provide any other configuration storage—the files themselves are the persistent storage. Another nice feature is that the driver starts immediately with the correct configuration and does not need to be initialized from an external source. Inotify works well with select() and is ideal in many situations, but be aware of a couple issues. You may have a race condition if your driver modifies a configuration parameter before making it available. Sound drivers often have this behavior—you can set the sample rate to 45KHz, but the driver probably will round it to closest standard value, 44.1KHz. If you do something like this with inotify, you may have a window where a reader would get the wrong value for the configuration parameter. Also, be aware that you may need to rebuild the kernel to include inotify support.

You also can use UNIX sockets for driver configuration if you split configurations reads from configurations writes. The problem is that when the user-space driver dæmon accepts a socket open request, the new socket is both readable and writable. The dæmon cannot tell if the user is trying to write a new configuration value or trying to read the existing one. One way around this problem is to have two sockets for every

I LIKE FUSE; IT CAN SOLVE A LOT OF USER-SPACE DRIVER PROBLEMS, BUT IT SEEMS UNFAIR TO ME TO ASK FUSE, A FILESYSTEM, TO DOUBLE AS A CHARACTER DRIVER.

driver configuration parameter, one socket for reading the current value and another socket for setting the value.

There are a couple good ways to add bidirectional data streams to your user-space drivers. The proxy driver provides this immediately and was our choice for the robotics project. The other approach is to use UNIX sockets. Sockets work well with select(), and their permissions map into a filesystem, but they don't work easily with echo, cat and command-line pipes. Also, if you use UNIX sockets for bidirectional transfers, you really shouldn't call them "device drivers" when you describe your system.

INSTALLING THE FANOUT AND PROXY DRIVERS

The fanout and proxy modules are fairly straightforward to build and install. Be sure the kernel header files for your kernel are available. Both drivers are in the tarball at **www.linuxtoys.org/usd/usd.tar.gz**. Download the driver tarball, then untar, build and install the drivers:

```
tar -xzf proxy.tar.gz
cd proxy
make
sudo make install
```

How and where you install your modules and create device nodes is a matter of personal preference. You can, for example, add the following to your rc.local startup script or put the equivalent commands in a udev rules file:

```
modprobe fanout
FANOUTMAJOR=`grep fanout /proc/devices | awk '{print $1}'`
mknod /dev/fanout  c $FANOUTMAJOR 0
mknod /dev/fanout1 c $FANOUTMAJOR 1
mknod /dev/fanout2 c $FANOUTMAJOR 2
mknod /dev/fanout3 c $FANOUTMAJOR 3
mknod /dev/fanout4 c $FANOUTMAJOR 4
chmod 666 /dev/fanout*

modprobe proxy
PROXYMAJOR=`grep proxy /proc/devices | awk '{print $1}'`
mknod /dev/proxy  c $PROXYMAJOR 0
mknod /dev/proxy1 c $PROXYMAJOR 1
mknod /dev/proxy2 c $PROXYMAJOR 2
mknod /dev/proxy3 c $PROXYMAJOR 3
mknod /dev/proxy4 c $PROXYMAJOR 4
chmod 666 /dev/proxy*
```

The robot's bootup scripts are slightly different because we wanted the device node names to reflect the device it serves. For example, the dual quadrature decoder might create fanout device nodes with the following:

```
mknod /dev/dp/quad0 c $FANOUTMAJOR 0
mknod /dev/dp/quad1 c $FANOUTMAJOR 1
```

The source tarball contains some simple demonstration programs in the demo directory. The program pxtest2.c shows how to use the proxy device to configure a user-visible string, and pxtest2 works by accepting a short string and echoing it back on request. As mentioned above, drivers often have to limit or otherwise modify a configuration value set by the user. The pxtest2 program demonstrates this kind of processing by adding one to each (non-newline) character in the input. You can run pxtest2 with the following commands:

```
gcc -o pxtest2 pxtest2.c
./pxtest2 /dev/proxy &
echo 111aaa222 > /dev/proxy
cat /dev/proxy
# output of the cat command should be 222bbb333
```

SUMMARY AND NEXT STEPS

Our ad hoc approach to building device drivers in user space has some nice features. It does not add a lot of kernel code and does not require any user-space libraries. It supports select() everywhere you might want it, and it has good flow control for streamed data.

Fanout and proxy have some shortcomings too. The data stream from a fanout device is byte-aligned, which makes it inappropriate for an application that needs to send blocks of binary data. Fanout could not, for example, be used to simulate a new /dev/input device. Demand Peripherals gets around this problem by sending lines of ASCII text that are terminated by a newline. If you need multibyte transfers, you could add an ioctl() to fanout that sets the byte count for atomic reads from the data source.

If you like the simplicity of /dev/proxy but really need ioctl() support, you can add it to the proxy driver by allocating two minor numbers for each proxy device. Use the even-numbered minor numbers for the data interface and the odd-numbered minor numbers for the ioctl() interface. Your configuration might look like this:

```
mknod /dev/proxy_data c $PROXYMAJOR 0
mknod /dev/proxy_ctrl c $PROXYMAJOR 1
```

Your additions to the proxy driver would have to serialize the data passed to and from the ioctl() request, and your user-space driver dæmon would have to open both devices to handle the ioctl() requests separately from the data stream requests.

We used fanout and proxy to add device drivers for an FPGA-based robotic controller, but they are actually fairly generic. What Linux problems can you solve using fanout and proxy?■

Bob Smith (bob@demandperipherals.com) is a consultant specializing in embedded Linux.

USING wview

If knowing what the weather is gets your blood pumping and your heart racing, you need to get wview and hook it up to your other great passion, which is, of course, Linux.

MARK TEEL

Are you fascinated with weather? Do you often find yourself checking local weather conditions? Is the weather your favorite part of the news broadcast? If so, you may be a weather geek, and wview may be the application for you.

wview is an open-source weather application that retrieves sensor readings from a weather station. The sensor data is stored in SQLite3 databases. Aggregate data, such as minimums, maximums and averages, are computed and stored in the database back end. Optional uses of the stored data include weather Web site generation; generic file generation for external applications; data submission to third-party organizations, including Citizen Weather Observer Program (CWOP) and Weather Underground; and store-forward to remote data collection centers. A user-friendly HTML interface is provided for configuring your weather station as well as for optional features.

To set up your weather station and publish your data with wview, you need a weather station. Supported stations include Davis Vantage Pro/Pro2 (Figure 1) or Vantage Vue, Texas Weather Instruments, Vaisala WXT510/520, Oregon Scientific WMR9X8 and La Crosse WS-23XX. Next, you need a platform to host the wview application. Desktop computers of any vintage work well, but it often is desirable to host wview on a low-power, unattended system. The now discontinued Linksys NSLU2 has been a popular choice. The new SheevaPlug quickly is gaining popularity as a wview host also. Industrious people even have used a Western Digital Worldbook NAS as their wview host. Because wview is modular and designed for embedded applications, it can be hosted on low-horsepower systems.

Next, you need to install a Linux distribution of your choice. The Debian (and derivatives) wview packages provide the most idiot-proof installation path, but source installs also are straightforward for any Linux distribution.

Finally, you need an interface cable. This may be a simple 9-pin serial cable or perhaps a USB-serial adapter if your host has no serial ports.

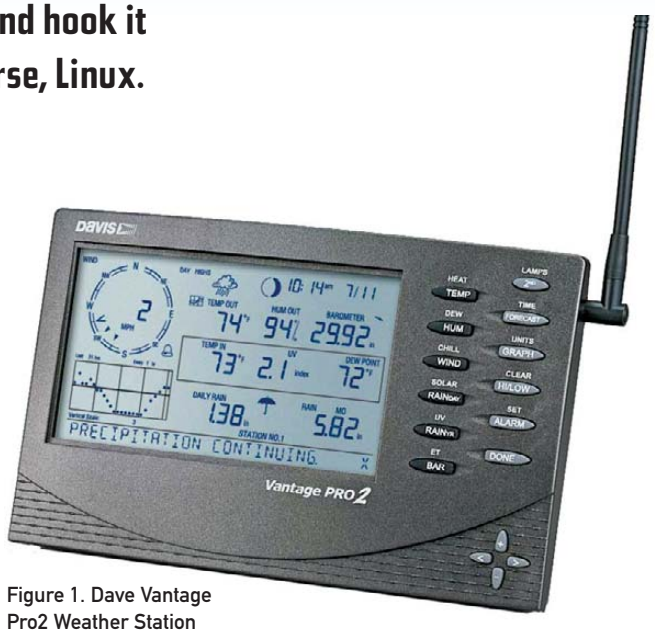


Figure 1. Dave Vantage Pro2 Weather Station

Configuration

To configure wview, open your favorite browser and point it to the wview management Web site, typically http://[your_wview_server]/wviewmgmt/login.php. An HTTP server is required on the wview host (this will be installed automatically if you use the APT packages). Use the default administration password "wview" (you can change this later). After logging in successfully, the System Status page is displayed (Figure 2). The System Status page displays the current state of all wview services as well as other status information.

Configuration is broken up into logical sections with context-sensitive help available by mousing over the configuration items. Click the Station tab to configure the station parameters (Figure 3).

The critical parameters here are the station type and the interface characteristics. Select Save Changes when you are done. Next, click the Services tab (Figure 4).

This page provides the configuration of wview services, log verbosity for the services and e-mail alerts. Services available are File Generation, Alarms, CWOP, HTTP (Weather Underground and

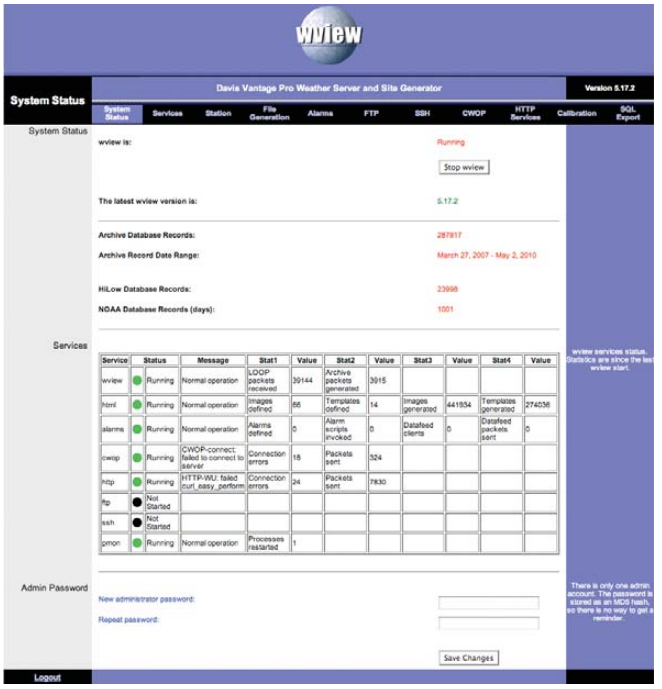


Figure 2. System Status

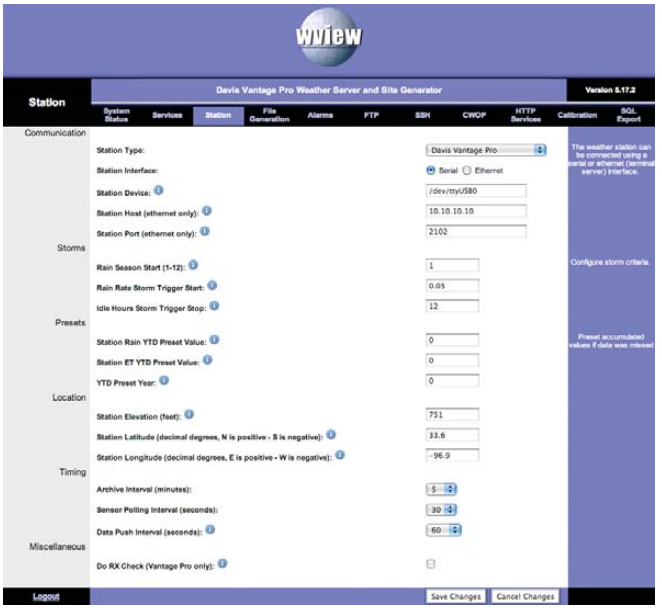


Figure 3. Station Configuration

Weatherforyou), File Export (SSH or FTP) and Process Monitoring. For now, let's not enable any additional services until you have confirmed your station interface.

Station Confirmation

Now, let's proceed to the station interface verification. Open a shell on the system that is running wview, so you can follow updates to the system log. At the prompt, enter the following:

```
$ sudo tail -f /var/log/syslog
```

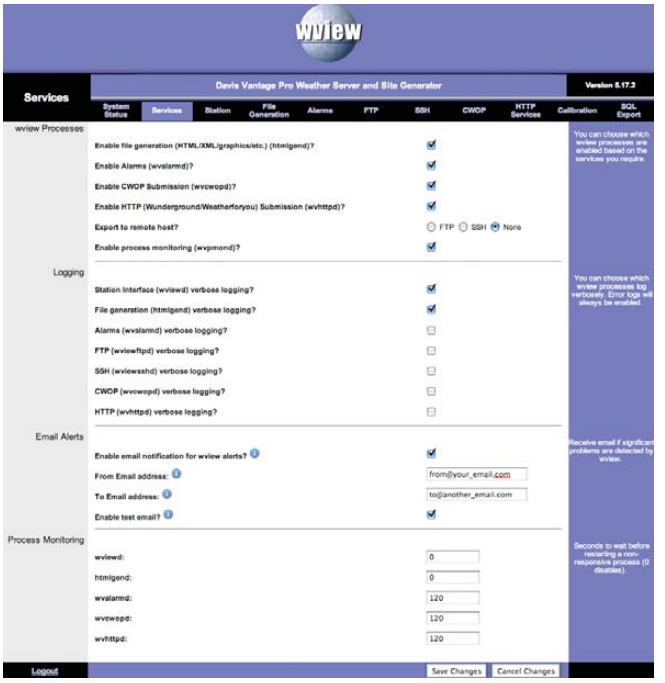


Figure 4. Services Configuration

This displays new system log messages as they are generated. Here, you will monitor wview startup and status messages. Open another shell, and execute the following:

```
$ sudo /etc/init.d/wview stop
$ sudo /etc/init.d/wview start
```

You will see a flurry of activity in the system log from the wview processes as they start up. It is a good idea to familiarize yourself with these wview log messages, as a wealth of detail is included that can be very helpful.

Return to the System Status page and observe the status of the station interface and the file generation. If both are not status "green" and "Running", further investigation in the system log file will be required to find any configuration or station interface issues.

Default Web Site

If all is well, you now can view the default wview weather site. This is typically found at http://[your_server_url]/weather/index.html (Figure 5).

Current conditions are given in the table on the left and by the dials in the center and on the right. These values are updated every time station data is polled (default is 30 seconds). The weather site pages are regenerated every 60 seconds (configurable). Observing changes in the current conditions is an easy way to confirm proper station interface operation.

Historical data for the last 24 hours are presented as graphs. Graphs of the last 24 hours, the last 7 days, the last 28 days and the last 365 days are available on other site pages.

Citizen Weather Observer Program (CWOP)

CWOP is a system by which individuals with weather stations and the proper software can submit their weather data to an APRS-based data storage system, so that others, including NOAA

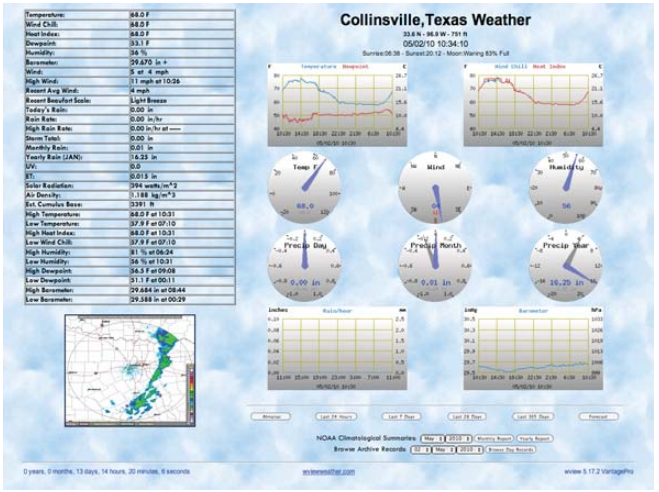


Figure 5. Default Web Site

(National Oceanic and Atmospheric Administration), can use the data however they see fit. There are some really neat station display Web sites, including some Java apps to look up station data, position, maps and so on. See **www.findu.com/cgi-bin/wxpage.cgi?call=CW4097** for an example weather station.

CWOP participation requires registering for an APRS callsign. Once you have configured wview for CWOP properly and confirmed your data on-line, you must contact the maintainers via e-mail to confirm your registration. Then your data will be available for anyone to see and possibly be used in NOAA forecast models and so on.

When CWOP support is enabled and configured properly, wview transmits a new WX packet to the APRS server every ten minutes, based on the last digit of your callsign.

wview supports the APRS-IS Rollover (Automatic Packet Reporting System-Internet Service) functionality by enforcing the definition of three APRS-IS server:port pairs. The goal is to avoid data loss to the CWOP system caused by connection errors. Select three different servers from the list at **www.wxqa.com/activecwd.html**.

Click the Services tab and enable CWOP submission and CWOP verbose logging. Click Save Changes. Next, click the CWOP tab, and enter your callsign, latitude and longitude (see the mouse-over help for format details), the CWOP servers (three should be entered) and port numbers. Go ahead and enable "Log CWOP Packet?"; you can disable it after submission is confirmed. Click Save Changes.

Now, restart wview:

```
$ sudo /etc/init.d/wview restart
```

You can monitor CWOP packet submission in the system log (and on the CWOP status pages).

Weather Underground

The Weather Underground (Wunderground) is a privately held organization that provides many weather services—some free and some not. Among the free services is the ability to register your weather station and submit your data to them, so you can access your data and some nice graphs from the Wunderground site. Weatherforyou.com also is a privately held outfit with similar capabilities to Wunderground.

Register for a Weather Underground Station ID (unless you already have one) at **www.wunderground.com/weatherstation/usersignup.asp**. Determine your accurate latitude and longitude: **www.topozone.com/viewmaps.asp**.

Click the Services tab and enable the HTTP service. Click HTTP Services and configure the Weather Underground settings. Click Save Changes.

Look in the system log for something similar to:

```
"WUNDERGROUND: configured to submit \
                station KTXCOLLI1 data to wunderground.com"
```

Confirm your data at the Wunderground server: http://www.wunderground.com/weatherstation/WXDailyHistory.asp?ID=XXXXXXX, where XXXXXXX is your Wunderground Station ID. This should start displaying your weather data graphically and as a packet list.

Data Archive and External Applications

A number of simple open-source weather station applications are available that do little more than extract the data from the weather station and archive it for later post-processing or retrieval by another server for multisite analysis. Researchers wanting to gather weather data for their own purposes is one example of such an implementation.

It is often (incorrectly) asserted that wview is "more application" than is needed for simple archival purposes. In fact, wview allows for much configuration as to "how much" it does for you. Designed as a series of loosely coupled UNIX processes, wview easily can be configured as an archive-only server. It also is easy to add CWOP and/or Wunderground/Weatherforyou to the archive server—all without any "fancy" HTML or other file generation. If you don't want to generate a Web site, you don't have to have one!

After installation and typical configuration, disable all wview Processes (under the Services tab in wviewmgmt). The station interface process always is enabled and, thus, is not configurable. Start wview as normal. Only the wviewd_<station> dæmon will be running, collecting data from the station and archiving records and HILOW values in the archive databases.

Because wview stores archive data in SQLite3 databases, it is a simple matter to implement scripts or applications that access the data via SQL. Many wview users create their own custom Perl/PHP/Java/WordPress applications for their weather data.

Weather Site Customization

The default generation model for wview is to generate a weather Web site based on a series of HTML file templates and images. For any template file named example.[ext]x and listed in html-templates.conf, wview will generate a file named example.[ext]. Thus, myscript.phpx listed in html-templates.conf and found in $prefix/var/wview/html will have all wview tags replaced, and the resulting file will be named myscript.php. For any template file named example.htmx and listed in html-templates.conf, wview will generate a file named example.htm. The resulting files are stored at the location specified on the wviewmgmt File Generation page: "Generation Target Path".

Changing HTML templates in $prefix/etc/wview/html does not require you to restart wview. The changes you make will take effect at the next htmlgend (HTML generation dæmon) generation cycle. Changing the config files images.conf, html-templates.conf and (if supported) forecast.conf does not require restarting wview, but it does require an HUP signal to be sent to htmlgend to cause these files to be reread. Do this as follows (this also will toggle log verbosity):

```
$ sudo kill -s HUP `cat $prefix/var/wview/htmlgend.pid`
```

wview supports template macro file inclusion in template files. The meta-tag is <!--include filename.xxx-->. Any template macro file that is to be included in one or more template files should be listed *before* any templates including it in the $prefix/etc/wview/html-templates.conf configuration file. There is no restriction on the levels of inclusion, just be sure you specify macro templates early in the html-templates.conf file. The wview default Web site templates utilize several header macro files.

HTML template files (in $prefix/etc/wview/html) can be customized to your language and design preferences. The configuration file html-templates.conf specifies the template files to be used for generation. You may add or remove from this list as needed. Weather image captions can be edited in the $prefix/etc/wview/images.conf file for your language preferences. The configuration parameter on the File Generation Page "Enable Metric Units For Generation?" allows for configuration of metric units. If set to "yes", it causes wview to output all images (buckets and charts) as well as all values for HTML tags in metric units. The file images.conf can be edited to translate the English labels, titles and units to any language. By editing this file and the HTML template files, any language can be supported by wview. In fact, you easily can switch back and forth between US and metric units by toggling this configuration parameter and restarting wview.

Advanced Features

wview provides a number of features that allow advanced use of the weather data collected from your station. Alarms may be defined such that if an upper or lower bound is exceeded, a user-defined script will be executed. These scripts may send a notification e-mail or trigger an external application. It's also possible to connect to the wview server via TCP/IP socket and receive an unsolicited, periodic data feed of weather data. By using the "Virtual" station type, you can connect to another wview server remotely and receive the station data as if it were connected directly to the station hardware.■

Mark Teel is the Software Engineering Manager for a major supplier of display and control systems for mass transit and commercial airline systems. He is also an advocate of open-source software development and has contributed to several projects, including CodeAnalyzer (a Java-based source code analyzer), radlib (Rapid Application Development Library) and wview.

Resources

wview Home Page and On-line User Manual:
www.wviewweather.com

wview Google User Group: **groups.google.com/group/wview**

Citizen Weather Observer Program (CWOP): **www.wxqa.com**

The Weather Underground (Wunderground):
www.wunderground.com

Building Custom Firmware with OpenWrt

Using an inexpensive wireless router, you can build a file server, a print server and even a media server. Plus, you can put all of those together and build the network device that does what you want.

MIKE PETULLO

OpenWrt provides an environment for building custom, Linux kernel-based firmware for a variety of embedded devices. Originally targeting the Linksys WRT54G series of routers, OpenWrt now provides support for a wide range of devices, including Openmoko mobile phones and routers from Linksys, NETGEAR and D-Link. This article focuses on the Linksys WRT160NL router, which I chose because of its support for 802.11n wireless networking, its USB connectivity and its reasonable price. Within this device's 8MB of Flash and 32MB of RAM, I explain how to fit a Kerberos authentication server, LDAP directory server, NFS file server, print server and iTunes-compatible media server.

In addition to the Linksys WRT160NL router, you'll need an external USB hard drive, a USB printer, a USB hub, a build workstation and a custom-built console cable. To help build the console cable, You'll also need a continuity tester (often available as a function of a multimeter). The first two items are available for a total of less than $200. Any USB printer should suffice, as long as your network clients support it. The build workstation should be running an up-to-date Linux distribution. Finally, the console cable is a modified Nokia DKU-5 connectivity cable; this type of cable has a USB connection and embedded USB-to-serial adapter. In this article, I use the example.com domain, someuser user account and server.local router hostname. You should replace these three items with your own names.

This project has seven steps: preparing the build workstation, downloading the OpenWrt build environment, configuring and building a firmware image, installing a firmware image onto the WRT160NL, configuring the external USB disk, performing a post-installation configuration of the WRT160NL and configuring a network client. In addition, this article describes how to rescue a misconfigured WRT160NL using the console cable. If you find your router will not boot at any time while following these steps, skip ahead to that final section.

The first step to building a firmware image using OpenWrt is preparing your build workstation. OpenWrt requires that gcc, g++, binutils, patch, bzip2, flex, bison, make, gettext, pkg-config, unzip, the glibc headers, the libz headers, the ncurses headers and the perl-XML-parser are installed on the build workstation. All major distributions provide packages for these items, although the package names may be slightly different from the upstream titles.

Once you have installed all of the requisite packages on your build workstation, download OpenWrt using the command:

```
svn co -r 20526 svn://svn.openwrt.org/openwrt/trunk
```

(Before downloading OpenWrt, you may want to check if a newer version is available and substitute its revision number in the -r option.) This creates a directory named trunk in the current working directory. Enter this directory with `cd trunk`. Inside, you'll see the core OpenWrt build system. Additional packages are provided by what OpenWrt calls feeds. Pull in the feeds provided by the default configuration by executing:

```
./scripts/feeds update
```

Finally, complete the installation of the files required to build your optional packages with the command:

```
./scripts/feeds install krb5-server \
                        openldap-server \
                        nfs-kernel-server \
                        p910nd \
                        mt-daapd \
                        ntpd
```

The third step is to configure your firmware and build an image containing it. The OpenWrt build environment is similar to many BSD ports systems, MacPorts or Gentoo Linux in that it allows users to define a list of packages that the system will download and compile. Unlike these systems' general use, OpenWrt often must cross-compile its packages. For example, although my build workstation has an Intel Core 2 Duo processor, the WRT160NL router has an MIPS32 processor. As a result, before downloading and compiling general-purpose packages from source, OpenWrt downloads and builds a cross compiler and other build tools from source.

OpenWrt provides a configuration system that is very similar to the Linux kernel's and can be invoked with the command `make menuconfig`. You can navigate the menu-based configuration tool using the arrow keys and select submenus by pressing Enter. Activate an item with the Y key, or in the case of a series of choices, with Enter. Conversely, pressing N deactivates an item. Press the Esc key to return to a previous menu screen.

Within the configuration menu provided by the `make menuconfig` command, first select Target System and choose the Atheros AR71xx/AR7240/AR913x option. Set the Target Profile to Linksys WRT160NL. In the Target Images submenu, ensure that only squashfs is activated.

Activate Advanced configuration options (for developers), and press Enter while this option is highlighted. Select Toolchain options, and press Enter again. Deactivate Build/install c++ compiler and libstdc++, because none of the packages for this example build require a C++ environment.

Return to the main menu using the Esc key. Select Base System, and then block-mount and block-hotplug.

Return to the main menu and select Network. Activate Filesystem, nfs-kernel-server; Time Synchronization, ntpd; Printing, p910nd; howl-mdnsresponder and openldap-server.

Return to the main menu and select Kernel Modules. Activate Filesystems, kmod-fs-ext3 and USB Support, kmod-usb-printer and kmod-usb-storage.

Return to the main menu, select Extra Packages, and activate krb5-server.

Return to the main menu, select Sound, and activate mt-daapd. Finally, Esc out of the configuration tool, ensuring that you save your configuration when prompted.

Now that your OpenWrt build is configured, execute the command `make V=99` to build the firmware image. This process takes a long time to complete because OpenWrt is compiling the build environment itself, followed by a Linux kernel and all of the firmware's programs. More than likely, this will take several hours.

The result of the build process is a firmware image stored at bin/ar71xx/openwrt-ar71xx-wrt160nl-squashfs.bin. You may install the OpenWrt firmware onto your router using Linksys' firmware upgrade tool, assuming you have not already replaced Linksys' default firmware. The Linksys firmware provides a Web-based configuration (Figure 1). The Linksys firmware has a default IP address of 192.168.1.1, a default user name of "admin" and a default password of "admin". After configuring your build workstation by adding it to the 192.168.1.0 network, point your browser to 192.168.1.1. Click Administration and then Firmware Upgrade. Finally, click Choose File, and select the firmware image you just built, namely openwrt-ar71xx-wrt160nl-squashfs.bin. Follow the directions on the screen to replace Linksys' firmware with the OpenWrt firmware. Restart the router once the upgrade process is complete.

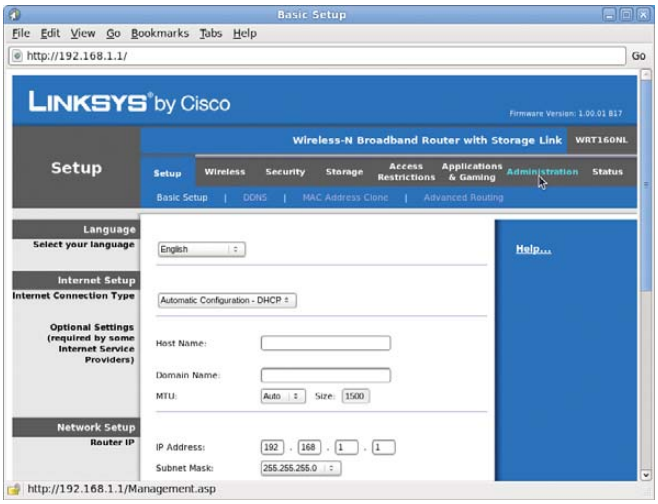


Figure 1. Firmware Upgrade Utility

Now that you have installed your firmware on the router, it's an appropriate time to focus your attention on the USB disk that will serve as the router's data store. Connect the disk to your build workstation. You're going to create two partitions on the disk, one 64MB swap partition and one filesystem partition spanning the rest of the disk. You can do this with the following parted commands:

```
$ parted /dev/sdX
 (parted) mklabel msdos
 (parted) mkpart primary 0 64
 (parted) mkpart primary ext3 64 -0
```

```
 (parted) quit
$ mkswap /dev/sdX1
$ mkfs.ext3 /dev/sdX2
```

Remember to replace the X in /dev/sdX with the correct letter for the newly connected drive on your system. Finally, create a directory skeleton as follows (assuming /mnt is an unused mountpoint):

```
$ mount /dev/sdX2 /mnt
$ mkdir -p /mnt/Storage/Music /mnt/home /mnt/mt-daapd
```

Now, copy your music library to /mnt/Storage/Music.

One final step remains before you turn your attention to your clients. The final step on the router is the post-install configuration. Connect the newly initialized disk, the hub and the printer to your router, and restart the router with the router connected to your build workstation but not yet connected to a public network. Ensure that your workstation is configured to obtain an IP address using DHCP. Once the router has finished starting, connect to it using `telnet 192.168.1.1`. Change the root password using `passwd`. Once this is complete, the router will no longer accept telnet connections. Instead, you will connect using secure shell: `ssh root@192.168.1.1`.

Now that you've logged in to the router, let's pause and address an issue you may be wondering about: the next firmware upgrade. Once the OpenWrt firmware has been installed, you can no longer use the Linksys firmware upgrade tool that you used previously. You will perform future firmware installations using an OpenWrt-provided command-line tool. First, use the `scp` command to copy a firmware image to the router's /tmp directory. Then, log in to the router, and execute `mtd -r write <path-to-image> firmware`.

Continuing work at the router's command prompt, now let's create nine configuration files. The first file configures the router's hostname and timezone, /etc/config/system:

```
config system
     option hostname server.local
     option timezone  UTC
```

The next three files configure the router's network parameters. First, you'll configure the router's Ethernet devices. The following configuration will cause all five Ethernet interfaces to be bridged to perform switching on one network, 192.168.1.0. The configuration sets the router's IP address to 192.168.1.2 (this is a change from the default, 192.168.1.1). This configuration assumes that routing between the 192.168.1.0 and upstream networks and Internet DNS resolution will be performed by another device (for example, a DSL device) with an IP address of 192.168.1.1:

```
/etc/config/network:

config interface loopback
     option ifname  lo
     option proto   static
     option ipaddr  127.0.0.1
     option netmask 255.0.0.0

config interface lan
     option ifname "eth0 eth1"
     option type     bridge
     option proto    static
     option ipaddr   192.168.1.2
     option netmask  255.255.255.0
     option dns      192.168.1.1
     option gateway  192.168.1.1
```

The WRT160NL has a total of five Ethernet ports, but the configuration option above shows two:

```
option ifname "eth0 eth1"
```

This is correct. The WRT160NL's four LAN ports perform switching functions and are collectively referred to by the Linux kernel as eth0. The kernel refers to the single WAN port as eth1. This configuration bridges all five Ethernet ports together to perform switching functions on one subnet. It also configures the switch to act as a host, assigned the IP address 192.168.1.2. After applying this configuration, all five Ethernet ports are equivalent switch ports.

Next, configure the router's wireless interface:

```
/etc/config/wireless:

config wifi-device radio0
     option type     mac80211
     option channel  5
     option macaddr  <MAC-ADDRESS>
     option hwmode   11ng
     list ht_capab   HT40-
     list ht_capab   SHORT-GI-40
     list ht_capab   DSSS_CCK-40

config wifi-iface
     option device   radio0
     option network  lan
     option mode     ap
     option ssid     <SSID>
     option encryption psk2
     option key      <WPA2 KEY>
```

Configure the DHCP service by writing /etc/config/dhcp:

```
config dhcp lan
     option interface    lan
     option start        100
     option limit        150
     option leasetime    24h
     # GW, DNS:
     option dhcp_option "3,192.168.1.1 6,192.168.1.1"

config dhcp wan
     option interface    wan
     option ignore       1

config dnsmasq
     option leasefile   '/tmp/dhcp.leases'
     option resolvfile  '/tmp/resolv.conf.auto'
```

Next, remove the default fstab using `rm /etc/config/fstab`,

because all mounts and swap space will be set up dynamically by the hotplug system.

Configure the disk's two shared directories using /etc/exports:

```
/mnt/sda2/Storage *(fsid=0,rw,insecure,no_subtree_check,sync)
/mnt/sda2/home *(rw,insecure,no_subtree_check,sync)
```

Configure Kerberos by creating /etc/krb5.conf:

```
[libdefaults]
    default_realm = EXAMPLE.COM
    dns_lookup_realm = false
    dns_lookup_kdc = false
    ticket_lifetime = 24h
    forwardable = yes

[realms]
    EXAMPLE.COM = {
        kdc = localhost:88
        admin_server = localhost:749
        default_domain = local
    }

[domain_realm]
    .local = EXAMPLE.COM
    local = EXAMPLE.COM
```

OpenLDAP's configuration file is /etc/openldap/slapd.conf:

```
include     /etc/openldap/schema/core.schema
include     /etc/openldap/schema/cosine.schema
include     /etc/openldap/schema/inetorgperson.schema
include     /etc/openldap/schema/nis.schema
include     /etc/openldap/schema/autofs.schema

allow bind_v2

pidfile     /var/run/openldap/slapd.pid
argsfile    /var/run/openldap/slapd.args

database    ldif
directory   "/etc/openldap/ldif"
suffix      "dc=example,dc=com"
rootdn      "cn=Manager,dc=example,dc=com"
rootpw      "<PASSWORD>"
```

Configure the media server, mt-daapd, by writing to /etc/mt-daapd.conf:

```
web_root        /usr/share/mt-daapd/admin-root
port            3689
admin_pw        <PASSWORD>
db_dir          /mnt/sda2/mt-daapd
mp3_dir         /mnt/sda2/Storage/Music
servername      server.local
runas           nobody
playlist        /etc/mt-daapd.playlist
extensions      .mp3,.m4a,.m4p
```

Finally, use /etc/config/p910nd to configure printer sharing:

```
config p910nd
    option device         /dev/lp0
    option port           0
    option bidirectional  1
    option enabled        1
```

Because OpenWRT sometimes starts services before the kernel initializes the USB subsystem, you need to make one last modification. Edit /etc/rc.d/S50mt-daapd and add `sleep 5` as the first line in the start() function. This ensures that mt-daapd does not start until the kernel is aware of the USB disk containing your media files.

Reboot the router to ensure all configuration changes take effect.

The next step is to initialize the router's Kerberos database. Log back in to the router using `ssh root@192.168.1.2`. Initialize the account database using the command `kadmin.local`, which provides an interactive interface:

```
$ kadmin.local
  > add_principal someuser
  > exit
```

Returning to the build workstation, let's initialize the LDAP database. First, create a file named example.com.ldif with the following contents, a database defining a user account and automount configuration in LDIF format:

```
dn: dc=example,dc=com
objectClass: organization
objectClass: dcObject
o: Example Organization
dc: example

dn: ou=people,dc=example,dc=com
objectClass: organizationalUnit
ou: people

dn: ou=group,dc=example,dc=com
objectClass: organizationalUnit
ou: group

dn: cn=ldapusers,ou=group,dc=example,dc=com
objectClass: posixGroup
objectClass: top
cn: ldapusers
userPassword:: WFhYWA==
gidNumber: 1002

dn: uid=someuser,ou=people,dc=example,dc=com
uid: someuser
cn: Some User
objectClass: account
objectClass: posixAccount
objectClass: top
userPassword:: WFhYWA==
loginShell: /bin/bash
uidNumber: 1100
gidNumber: 1002
homeDirectory: /home/someuser
gecos: Some User
```

```
dn: automountMapName=auto_master,dc=example,dc=com
objectClass: top
objectClass: automountMap
automountMapName: auto_master

dn: automountKey=/home,automountMapName=auto_master,dc=example,dc=com
objectClass: top
objectClass: automount
automountKey: /home
automountInformation: auto.home

dn: automountMapName=auto.home,dc=example,dc=com
objectClass: top
objectClass: automountMap
automountMapName: auto.home

dn: automountKey=*,automountMapName=auto.home,dc=example,dc=com
objectClass: top
objectClass: automount
automountKey: *
automountInformation: server.local:/mnt/sda2/home/&
```

Next, load the file into the LDAP database on the router using:

```
ldapadd -x -D "cn=Manager,dc=example,dc=com" -W -f example.com.ldif
```

Finally, return to the router and create someuser's home directory, `mkdir /var/sda2/home/someuser`, and then do `chown 1100:1002`. Notice that the UID:GID arguments to the chown command match those assigned to someuser in the LDIF file above. Now you can connect the router to your real network.

To add additional users, review the fifth block in the LDIF file above, tailor it for each user and add them using `ldapadd`. Also, create each user's home directory as before.

Your router now should be fully functional, so it is time to configure the clients. Several existing articles document how to configure a client to operate in LDAP and Kerberos environments (see Resources). You may configure your clients by editing configuration files, or you can investigate your distribution's administrative tools.

One difficult point I've encountered has to do with the LDAP schema used by the automounter. Different UNIX flavors use different schemas. My instructions use the schema employed by Mac OS X, because it also is supported on Fedora. In order to instruct the Fedora automounter to use this schema, write the following to /etc/sysconfig/autofs:

```
MASTER_MAP_NAME="auto_master"
TIMEOUT=300
BROWSE_MODE="no"
USE_MISC_DEVICE="yes"

# Mac OS X 10.5-compatible schema:
MAP_OBJECT_CLASS="automountMap"
ENTRY_OBJECT_CLASS="automount"
MAP_ATTRIBUTE="automountMapName"
ENTRY_ATTRIBUTE="automountKey"
VALUE_ATTRIBUTE="automountInformation"
```

Other distributions may configure the automounter differently. If the automounter is configured to use the appropriate schema, it will learn of the NFS-provided home share from the LDAP entry you created earlier and mount users' home directories on-demand.

Refer to your client system's documentation for instructions covering how to connect to the network printer. Your OpenWrt firmware will share an attached USB printer using the HP Jetdirect protocol. This protocol is supported by Linux, Mac OS X, Windows and many other operating systems.

Your OpenWrt device's music share is accessible by iTunes, Rhythmbox and any other media player that supports the DAAP protocol. Compatible players usually will display the share as an option alongside their local music library list.

What if you install a defective firmware, and the router is left in a state that will not boot? This is where your console cable comes in. Figure 2 shows a completed console cable next to an unmodified DKU-5. There are three points on the WRT160NL to which you can connect a console cable. The first are two identical sets of leads within the WAN and LAN 4 RJ45 sockets, opposite the Ethernet leads. The second is a five-post connector on the WRT160NL's motherboard. In order to use the latter interface, you have to open the WRT160NL's case, which voids the device's warranty.

Figure 2. Console Cable

Figure 3 provides a picture of a Nokia DKU-5 cable connector. The other side of the cable has a USB connector. Cut the cable in half so that most of its length is on the USB connector side. Now, expose the six leads on the Nokia connector side. Use a continuity tester to identify which four of the six leads correspond to the functional pins noted in Figure 3. Note the insulator color of the lead connected to each pin. Now focus your attention to the other half of the cable, the one with the USB connector. Expose the four leads matching the colors noted earlier. Connect each of these leads to the four of five binding posts labeled in Figure 4, ensuring that the lead and post functions match. You may connect them using a PCB connector, hook to pico hook jumpers or a more primitive technique.

Connect the USB connector of the console cable to your build workstation. Install a serial terminal emulator, such as minicom, on your build workstation and run the program. Configure the emulator to use eight data bits, no parity bit and one stop bit. Select 115200 baud. Boot the router and observe the emulator console. You should see the router print

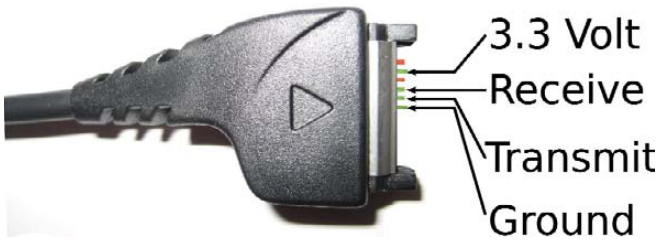


Figure 3. DKU-5 Connector

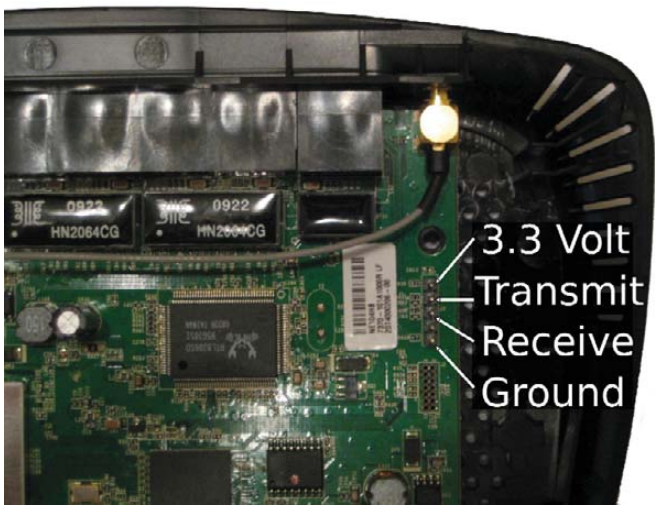


Figure 4. Router's Console Interface

diagnostic information through the console interface. Pay close attention to the messages being printed through the console, and press a key when the U-Boot bootloader prompts you to "Hit any key to stop autoboot." You should see the U-Boot command prompt, `ar7100>`. Enter the command `upgrade code.bin` to instruct U-Boot to initiate a tftp server. Return to the build workstation command line, enter the command `tftp 192.168.1.1`, and then:

```
tftp> binary
tftp> put <firmware filename>
```

Observe the data transfer being displayed over the console interface. Return to the terminal emulator, and enter `go` at the U-Boot prompt.

The firmware you load using this technique may be an official firmware obtained from Linksys or an OpenWrt firmware that you built.

Conclusion

This article demonstrates a technique for providing Kerberos, LDAP, network filesystem, print and media services using a Linksys WRT160NL wireless router. The result is a low-cost network server for the home or small office. OpenWrt has a wide range of packages available, so there is potential for many other solutions to be developed around this capable platform. For example, Samba could allow tight integration with Windows clients. Another option is Netatalk, which provides native Mac OS X file sharing, including integration with Apple's Time Machine backup software. Linux, open-source applications and popular network hardware like the Linksys WRT160NL provide a solid basis for developing innovative devices.■

Mike Petullo serves in the US Army and enjoys solving problems with innovative open-source software. He has been working with Linux since 1997 and welcomes your comments at mike@flyn.org.

Resources

"OpenLDAP Everywhere" by Craig Swanson and Matt Lung, *LJ*, December 2002 (see the section titled "Configure the Linux LDAP Client"): **www.linuxjournal.com/article/6266**

"Centralized Authentication with Kerberos 5, Part I" by Alf Wachsmann, *LJ*, January 2005 (see the section titled "Configuring the Clients"): **www.linuxjournal.com/article/7336**

"Serving Apples: Integrating Mac OS X clients into a Fedora network" by Mike Petullo, *Red Hat Magazine*, January 2008 (demonstrates how to configure Mac OS X clients): **magazine.redhat.com/2008/01/17/serving-apples-integrating-mac-os-x-c lients-into-a-fedora-network**

REAL-TIME PLOTS *with* kst *and a* Microcontroller

Follow along as we build a real-time data graphing system using kst, an Arduino microcontroller and a Linux notebook.

Rob Reilly

Lots of programs take data from a file and create an X-Y graph under Linux. Desktop applications like xplot, gnuplot or even PHPlot do a great job. But, what if you want to see how a physical process changes and use a real-time plot on your Linux machine?

I couldn't find this capability for a long time. Then, I discovered kst. kst is a fast, large-data set, real-time viewing and plotting program, and it's part of the KDE suite.

You need to have some way to get real-time sensor data into the kst program. I've used Arduino microcontrollers to automate different things, so it seemed only natural to combine one of these boards with kst to build an easy-to-use and very capable real-time data-gathering system. Because it's open-source-based, expansion and customization are possible.

In this article, I show how to link all the parts together to produce a real-time plot of live data and explain how to install and test kst. I also cover Arduino programming environment installation, so you can get the board programmed and stream data right into a Linux notebook.

Installing and Testing kst

kst can read text-based data from a file and has basic data analysis capabilities. As part of the KDE suite of applications, it is available on virtually all modern Linux distributions.

The easiest way to put kst on your machine is with your distribution's package manager. I used Synaptic under Xubuntu for the installation on my ASUS 64-bit Core Duo X83-VM notebook.

Once installed, kst appears under the Applications and Accessories pull-down tabs on the desktop taskbar.

Below is a small segment of some temperature and light-level data that I captured. The data snapshot will be used to test kst. Later, this same format will be used to stream real-time data from the Arduino into our Linux machine. Copy the data into a text file named testdata.txt:

```
74.64|444
74.64|448
74.64|452
74.64|450
74.64|447
74.64|439
74.64|435
```

Then start kst. The main kst window will show the task bar across the top and the kst QuickStart window in the middle.

Click on the Data Wizard button at the bottom of the Kst QuickStart pop-up pane. Figure 1 shows the kst toolbar, data source and configure data source windows. The pop-up Data Source pane will appear. Enter the data filename, testdata.txt. Press the Configure button. The Configure Data Source pane appears. Enter the custom delimiter character to separate the values in the data set. I used the vertical bar as a delimiter between the temperature and light-level values.

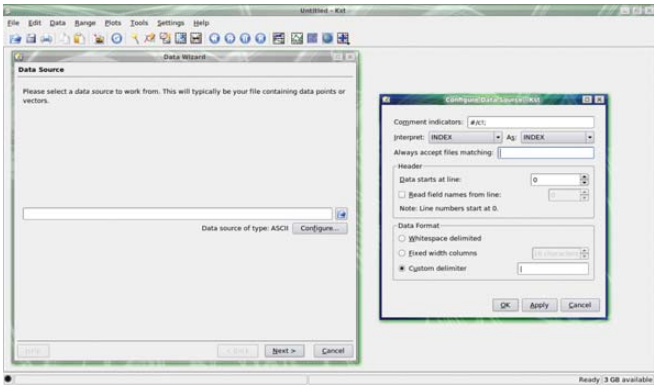


Figure 1. kst Toolbar, Data Source and Configure Data Source Window

Once the delimiter character is set, click the Apply then the OK buttons to save the settings and close the window. Click Next on the Data Source pane to bring up the Select Data pop-up window.

In the Data Source pane, hold down the Ctrl key and select numbers 1 and 2 in the left-hand pane. These correspond to the temperature (left) and light-level (right) values in the data file. Once selected, click the right-pointing arrow to copy the data streams to the right-hand pane. Using two data streams will give two separate graphs, one for temperature and one for light levels, referenced by a common line number. Temperature and light levels

will appear on the y-axis, and the line numbers will appear across the x-axis on each respective plot.

Plot customization is done with several pop-up windows. Click anywhere on the top (temperature) plot label to bring up the Edit Plot pop-up window. Select the Appearance tab to edit the labels. kst assigns its own default labels. In my case, I changed the x and y labels to reflect the data that the plot was showing, namely the temperature, light levels and time interval. Modify the label fonts, font sizes, justification and other assorted options to your tastes. Other tabs under this window control how data is plotted on the x and y axes and the range of numbers displayed. Whenever you make a change on one of these tabs, be sure to click the Apply button then the OK buttons to save the changes.

This sets up a template for future runs with that data file. It doesn't matter if the file is static or grows over time. kst will start plotting what's in the file the next time the template is selected. Assign an appropriate name to the template file.

Now that you've installed and tested kst with a static data file, it's time to program the Arduino to sense the environment (temperature and light level), then stream the data out over the USB line to the notebook.

Arduino Open Hardware Primer

A simplified description of a microcontroller is that it's a small-footprint computer that can read and interpret input pin values, make a few calculations or decisions, and then control output pin signals based on the embedded program.

Arduino microcontrollers are known as open hardware. In the spirit of the open-source software tradition, the board designs, schematics and code are freely available to download, modify and enhance. The Arduino home page has active forums for information exchange, and there is a comprehensive set of reference documents.

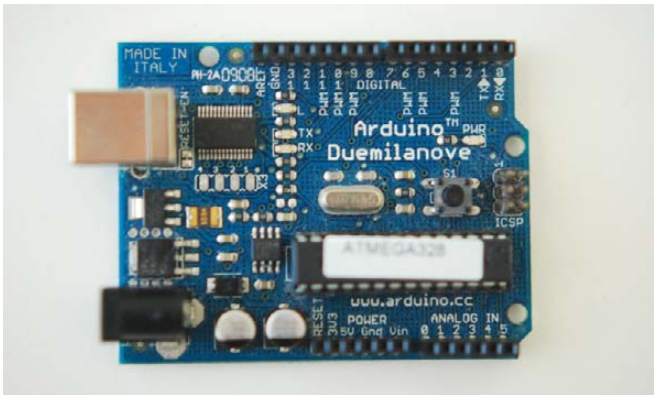


Figure 2. Arduino Board

Naturally, the boards and raw components cost money. A number of Arduino clones are available, offering board configurations for a variety of specialized applications. Prices for solder-it-yourself boards start at around $20. Complete ready-to-run boards with built-in USB interfaces cost about $30. You'll also need a breadboard, a few resistors, some jumper wires, a wall wart or battery and some sensors (suppliers are listed in the Resources for this article).

Inputs are either analog or digital. An analog device might be a potentiometer or photocell, while a digital device might be a magnetic reed switch or a push button.

Outputs control things. You could turn on a light with a digital output operating through a relay or transistor. In this article, we won't control anything with output pins. Instead, the Arduino will communicate sensor data to a Linux notebook, over the USB line.

Modern microcontroller modules, like the Arduino, take advantage of what's called in-circuit programming. The processor chip uses Flash memory for program storage and is accessed via the USB connection. Flash is a type of Electrically Erasable PRogrammable Read-Only Memory (EEPROM), which means it can be erased and rewritten multiple times using the proper electrical signals. These days, the term EEPROM normally is used only to refer to the more traditional type of EEPROM (which is still used in smaller sizes due to some of its other advantages). In-circuit programming is great because it minimizes equipment costs and prototyping turnaround time.

The flagship Arduino module is known as the Duemilanove. It is a 2.7" x 2.1" circuit board that has 14 digital I/O (input/output) pins and 6 analog input pins. It also has a built-in USB port, uses an Atmel Atmega 328 processor and screams along at 16MHz. Modules are powered by batteries or from a wall wart, with a recommended range of 7–12 volts DC.

Setting Up the Arduino

Arduinos are programmed in a language called Processing. The Arduino integrated development environment (IDE) manages compilation of the Processing source code into machine code that is then uploaded to the Arduino board. Veteran programmers quickly will note Processing's remarkable similarity in format and syntax to the C language.

The Arduino IDE runs on 32-bit or 64-bit Linux notebooks and Netbooks. Obviously, Netbooks are cool because they are tremendously portable. Windows versions of the IDE are available. So, you'll always have the capability to program an Arduino, even if you get in a spot and don't have your Linux notebook close at hand.

Download the latest Arduino software from the Web site. Various Java packages, gcc-avr and avr-libc need to be installed, along with the Arduino integrated development environment. See Resources for a good tutorial on getting everything working in a 64-bit Ubuntu environment.

Open a terminal, and move to the directory where you installed the Arduino IDE. On the command line, start the IDE:

```
rreilly>  ./arduino
```

The main Arduino code editor screen will appear. From the

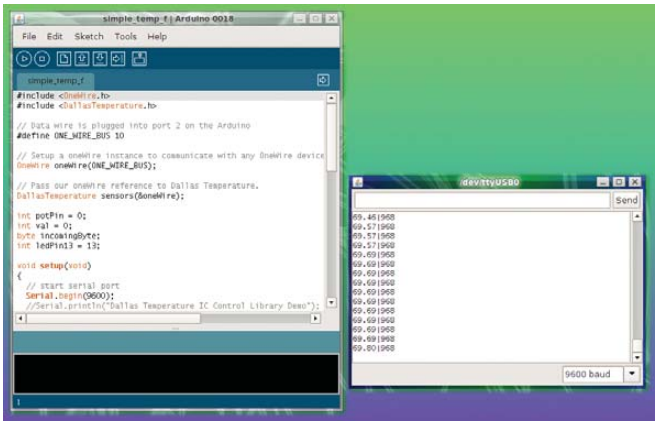


Figure 3. Arduino IDE and Editor Window

Listing 1. Source Code for This Project

```
#include <OneWire.h>
#include <DallasTemperature.h>

// Data wire is plugged into digital port 10 on the Arduino
#define ONE_WIRE_BUS 10

// Setup oneWire instance to communicate with OneWire temp device
OneWire oneWire(ONE_WIRE_BUS);

// Pass oneWire reference to Dallas Temperature
DallasTemperature sensors(&oneWire);

// Photocell input pin number
int potPin = 0;

// Declaration for photocell value
int val = 0;

// Arduino init functions
void setup(void)
{
  // Start serial port
  Serial.begin(9600);

  // Start up the library
  sensors.begin();
}

// Celsius to Fahrenheit conversion function
float c2f(float val){
  float aux = (val * 9 / 5);
  return (aux + 32);
}

// Main Arduino program loop
void loop(void)
{
  // Read photocell for light value
  val = analogRead(potPin);

  // Send command to get temperature from Dallas device
  sensors.requestTemperatures();

  // Convert returned C temp to F, print value
  Serial.print(c2f(sensors.getTempCByIndex(0)));

  // Print delimiter character in serial stream
  Serial.print("|");

  // Print (w/line feed) light-level value
  Serial.println(val);

  delay(1000);
}
```

Advertiser Index

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

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

ATTENTION ADVERTISERS

November 2010 Issue #199 Deadlines

Space Close: August 23; **Material Close:** August 31

Theme: Hacking

BONUS DISTRIBUTIONS: USENIX OSDI

Print: contact **Joseph Krack, +1-713-344-1956 ext. 118, joseph@linuxjournal.com**

On-line: contact **Michael Beasley, +1-713-344-1956 ext. 119, michael@linuxjournal.com**

drop-down File menu, select New to get a blank code window. Type in your program. You also can copy code from another source, such as Web examples or from the sample code bundled with the IDE in the Examples directory.

The examples offer standard routines to read various input sensors and control output pins. The Arduino Web forums and reference pages contain all kinds of code snippets that you can use instead of having to write everything from scratch. Just like open source in the Linux world, Arduino users are encouraged to develop and share their code.

After entering the code in a new file, select Save As under the File drop-down tab. Give your file a name that makes sense (in my case, simple_temp_f). The file will be saved in the Sketchbook directory with a .pde extension. Arduino source code files are called sketches, so, of course, that's where they are stored.

Once a program is entered and saved, you need to compile it. Under the Sketch tab, select Verify/Compile to produce the machine code. After a short period, a message noting the program size will appear in the status window at the bottom of the main editor window. Errors will show up highlighted in red. Most of my errors are usually typos or forgetting a variable declaration. As in C, don't leave out any semicolons.

Make sure the Arduino module is connected to the Linux notebook by the USB cable, and click the little upload button with the right-facing arrow on the toolbar. Some messages may appear in the status window at the bottom of the editor screen. Again, errors again will show up in red.

If you happen to be using an older version of the Arduino, such as the NG, you'll have to push the onboard reset button right before pressing the upload button to get the upload to start. There is a short upload window before the Arduino bootloader starts that is used to upload the program via the USB connection. Late-model Arduinos run a reset without the need for a manual button push.

In the middle of the Arduino module, the two onboard RX/TX LEDs will show that the machine code has been transferred from the notebook to the board.

The Arduino IDE and related programs are updated frequently, and I'm happy to report that version 0018 is much faster at compilation and uploading than version 0012. The speed increase goes hand in hand with the in-circuit programming capability. These steps minimize the program/compile/upload cycle and increases available prototyping time.

After the machine code is uploaded, the Arduino will perform a reset, and two seconds later, the bootloader will run the program and begin reading inputs and writing outputs.

You'll see the power LED light up, and if data is being sent over the USB (or optional serial line), the RX/TX LEDs will flash as data is moved back and forth.

The toolbar button in the very middle of the editor will open a new screen to view data coming in from the Arduino. It's called the serial monitor and is used to watch data transferred from the Arduino to the notebook. Note that the USB port on the Arduino is a USB-to-serial converter (an FTDI chip), so the Arduino shows up as a serial port on your computer.

Enough about Arduino programs. Let's link things together and make a real-time plot.

Putting kst and Arduino Together

Figure 4 shows the circuit required to read the photocell and hook up the Dallas DS18B20 one-wire temperature sensor. The

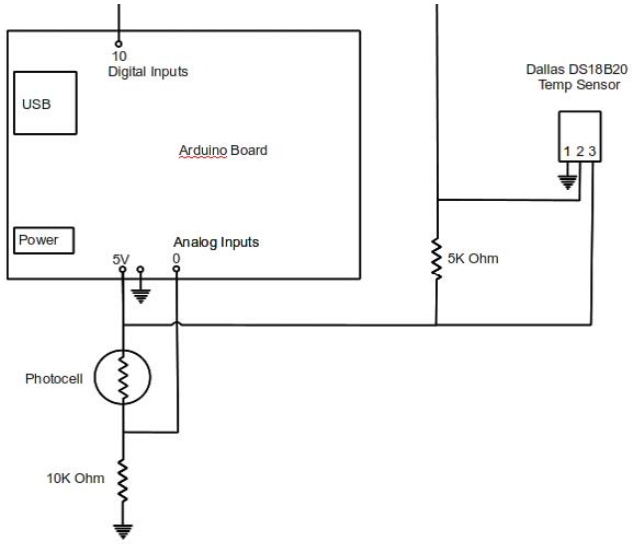


Figure 4. The Circuit Required to Read the Photocell and Hook Up the Dallas DS18B20 One-Wire Temperature Sensor

photocell produces changes in the voltage that is processed by one of the built-in analog-to-digital converters in the Arduino.

The Dallas sensor is a cool piece of technology, because a whole bunch of these sensors will work on a simple three-wire bus. Each sensor has a unique 64-bit device number. The Arduino code pings the Dallas sensors and receives a coded data stream from each one containing the device number and temperature reading. The Dallas sensor and one-wire libraries need to be added to the Libraries directory. Miles Burton built some awesome libraries and code; download them from his Web site (see Resources).

Code particulars are a little beyond the scope of this article. In a nutshell, the Arduino reads the photocell and temperature sensor values and converts them into a data stream, one line of data per program loop that is fed out over the USB port. Again, we don't change any output pins in this particular project.

Make sure the USB-serial port is configured to accept the data from the microcontroller. Open a terminal and use the stty command to set the baud rate for the port. If you have the wrong baud rate, you'll get funny characters that you can't read or import into kst:

```
rreilly>  stty -F /dev/ttyUSB0 9600 clocal
```

Plug the USB cable in to the port, wait a couple seconds, and the Arduino will start sending data to your notebook. Use the cat command, in a terminal, to record the data to the testdata.txt input file:

```
rreilly> cat /dev/ttyUSB0 > testdata.txt
```

Stop the data stream with Ctrl-C.

Once you have the data coming in from the USB port, start kst to view it. Remember, you set up the kst template file earlier. Select your template file from the menu when kst starts.

The two graphs should appear, and the plot will change as data streams in. Scaling is automatic by default and will work for many situations (Figure 5).

That's pretty much the rundown on plotting real-time data

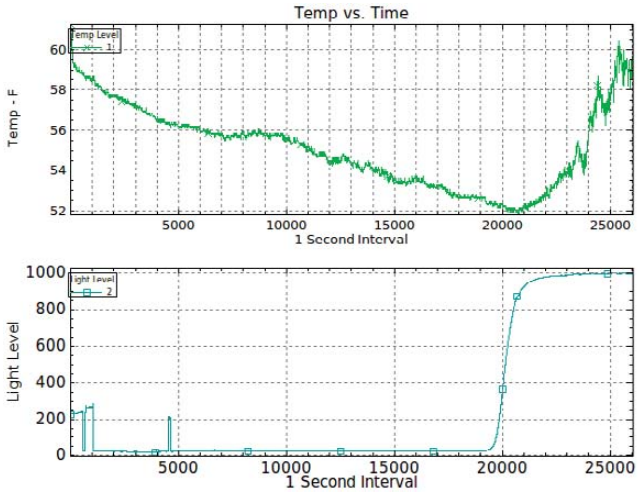


Figure 5. A Couple Live-Data Plots

with kst with an Arduino microcontroller and a Linux notebook. Explore the kst program for more display options.

Wrap Up

This article looks at real-time data plotting with an Arduino microcontroller and the kst viewer program on a Linux machine. Use this system as the basis of a more advanced setup with multiple inputs and sensors. The system also might expand to include wireless or battery operation. I didn't talk about sending data to the Arduino from the Linux notebook, so perhaps that will be a topic for a follow-up article.■

Rob Reilly is a technology consultant, writer and portable computing expert. Early adopter tech trends, seminars and media projects are his stock in trade. Links to many of his published articles appear on his Web site at home.earthlink.net/~robreilly. Contact Rob at robreilly@earthlink.net.

Resources

Arduino Home Page: **www.arduino.cc**

Arduino IDE Download Page:
www.arduino.cc/en/Main/Software

The Arduino IDE on Ubuntu Tutorial:
www.codetorment.com/2009/11/02/tutorial-getting-started-with-arduino-ide-on-linux-ubuntu-9-10

Miles Burton Dallas Temperature Sensor Libraries:
www.milesburton.com/index.php/Dallas_Temperature_Control_Library

Sparkfun: **www.sparkfun.com/commerce/product_info.php?products_id=666**

Adafruit: **www.adafruit.com/index.php?main_page=index&cPath=17**

Maker Shed: **www.makershed.com/ProductDetails.asp?ProductCode=MSGSA**

The OSWALD Project

Open source and education: a match made in computer heaven.

VICTOR KUECHLER AND CARLOS JENSEN

Computer Science (CS) education, for some time now, has been accused of being outdated and failing to capture the hearts and minds of the next generation of leaders and innovators. Rekindling the pioneer spirit and excitement of the hacking culture of the 1970s and 1980s requires flexible and accessible open hardware and software in an environment that promotes experimentation and collaboration. Although the software community has a long history of building and sharing such platforms, open and flexible hardware platforms have been few and far between. Oregon State University developed the Oregon State Wireless Active Learning Device (OSWALD) to help close this loop. The OSWALD is an open, Linux-based, ultra-mobile personal computer incorporating many of today's new technologies and gives students and enthusiasts full access to both its hardware and software. Based on discussions with the OSWALD development team, this article provides a brief overview of the platform and explores the lessons learned in re-inventing computer science through open source.

The OSWALD in Memorial Union Square at Oregon State University. Holding the OSWALD is Patrick Fancher, a New Media Communications student at OSU. (Photographer: Victor Kuechler)

Open Source and the State of Computer Science Education

One of the most frustrating challenges CS educators face today is convincing the next generation of students that room still exists for innovation and making a difference in computing. With the advancement of computing and its proliferation into most aspects of our lives, many prospective students feel computing is "a solved problem", that the days of innovation and chances to make a significant contribution have passed. We are victims of our own success. The rapid spread of cheap hardware and polished applications had removed one of the most compelling reasons many have had for studying computer science: the need to scratch our own itch. As a consequence, CS enrollment has dropped for more than a decade.

It doesn't help that the CS curriculum has, in many ways, failed to keep up with changes. Here I refer to the core curriculum, not the electives or advanced courses, where we often see a much richer and up-to-date set of topics. Little has changed in the core curriculum during the last 15 years short of programming language choices. In all fairness, the body of CS has been growing tremendously, and the fundamentals are still fundamental, but academia has failed to make sure this core stays relevant and visible to current and future students.

Open source provides a unique opportunity to help us bridge this gap. In an article titled "Computer Science Education in the 21st Century", David A. Patterson, former president of ACM and Pardee Professor of Computer Science at UC Berkeley, explained that open source provided unique opportunities for injecting realism and relevance into the classroom. One of Patterson's laments is that students can easily go through their undergraduate education without real exposure to computing. Students rarely read other people's code, and sharing of code is often the same as cheating. The largest codebase students might see is a few thousand lines of their code written over a period of 6–12 months with, at most, a half-dozen other students. This code will not evolve or be maintained. In all likelihood, it will be discarded as soon as the class ends. Open source allows educators to change this dynamic, opening a window into the chaotic and complex, yet realistic world of software development. A place where code is designed and written by many, and the context, metadata, discussion and collaboration required is available to all. It's a unique resource for educators to present an invaluable opportunity to students to learn real skills. Oregon State University has been busy developing curriculum centered on open source, and it is aware of how to navigate its sometimes discouraging waters.

Dr Carlos Jensen, an assistant professor at Oregon State University and driving force behind this project, explains:

> Students do not want to re-invent the wheel, and can be intimidated by the quantity and quality of easily available software. Many of us learned how to program by writing small programs designed to meet some need other software couldn't. Some of us attained glory by writing something our fellow students also found useful and would circulate. Today, chances are, there is already an app to meet most needs, and students often see joining large open-source communities as their only choice to make a difference in computing. It can be a daunting and intimidating prospect for someone learning how to code, and therefore, open source must be introduced in the right context.

The OSWALD

To be true to this open-source model, the OSWALD development team decided to take this model and apply it to both hardware and software. For students to learn through open source, it's important that this extend all the way down to the hardware platforms they use, which is still surprisingly difficult to do.

The OSWALD is a handheld ultra-mobile PC designed to be accessible and affordable to students, while incorporating many new technologies seen in the mobile device market. The OSWALD is an open platform meant to open up the creative space for student projects. With its relatively novel form factor, students are given the opportunity to explore and push the boundaries of computing again, as well as explore contemporary interaction techniques through hardware, such as a touchscreen and accelerometer. The device was carefully designed to balance power and performance with cost, equivalent to a college textbook.

We could have encouraged the use of Netbooks running open-source software. The price point and capabilities of these machines make them a tempting alternative to full laptops. The form factor was important to us for this reason. The device is an auxiliary computing device rather than a desktop or laptop replacement. Students treat their primary computing devices with care; this is where they store their media and do their work. The primary computing platform, therefore, is not a place where we would expect to see a lot of experimentation. The relatively little software customized for this form factor is a benefit to the OSWALD. A student might create or port an interesting piece of software and share it with another student who will appreciate it. With this in mind, students will have a long list of desired features to bring to the platform and are motivated to create better software.

The biggest challenges to building an open platform, such as the OSWALD, are the costs associated with manufacturing it, especially on a small scale. Because this is a university project, the devices basically are manufactured for students' own consumption at this point. For instance, though there is no shortage of companies that can assemble and test these devices, the devices are built by undergraduates. This means there will be positive and negative outcomes. There will be quality issues, and the devices take longer to assemble, but there is a relatively large number of freshman and sophomores working with the devices and, in the process, learning and gaining the confidence to work with computer hardware. Ben Goska, an undergraduate in Electrical Engineering and Computer Science and chief hardware developer elaborates: "We only make 300 OSWALDs in a build run. Talking to developers is tricky because most manufacturers are used to numbers in the thousands. We are building these by hand."

The hardware is also not as optimized as one would expect with commercial support, but there is intentionally a fair bit of redundancy. As Dr Jensen explains:

> I'm a user interface researcher. What I want to do is use this device to help students understand the trade-offs between different interface techniques. When you pick a phone, or a mobile device, typically you will have an interaction method chosen for you. For the iPhone, everything is touch. There is no way of doing anything other than by touching the screen. This is a very intuitive, slick interface, but there are instances where it's just a dumb idea. For a certain application, having a separate touch area makes more sense, or a rocker switch, or a joystick. We provide all of these because part of what I want to do is have a device that lets students experiment with all aspects of computing, including the interface. That's the reason for its flexibility. This is just one example of how, for a very specific type of course or curriculum, you can take advantage of that type of flexibility. Even if we didn't have all three capabilities built in to the device, I could plug in a mouse or a joystick, essentially anything, through a USB port that would give me that flexibility, and students can then learn their pros and cons.

The OSWALD at Oregon State University, Jefferson Street, Corvallis, Oregon (Photographer: Victor Kuechler)

At the university level, a cheap and open embedded computing platform can be valuable in multiple ways. At Oregon State University, the Physics Department has taken OSWALDs and ripped them apart to control scientific instruments. The robotics club uses the OSWALD board to control their Mars rover and aerial

OSWALD Specifications

SOFTWARE LAYER

- **RADIX**
  A custom distribution of Linux created using OpenEmbedded.
- **Matchbox**
  A lightweight, themeable, window manager designed for handhelds that provides the OSWALD's GUI framework and environment (**matchbox-project.org/overview.html**).
- **JamVM**
  A small and specification-compliant Java virtual machine (**jamvm.sourceforge.net**).
- **Leafpad**
  A simple GTK+-based text editor (**tarot.freeshell.org/leafpad**).
- **ePDFView**
  A lightweight PDF viewer (**trac.emma-soft.com/epdfview**).
- **GPicView**
  A lightweight GTK+-based image viewer with low memory usage (http://lxde.sourceforge.net/gpicview"/>).
- **PCManFM**
  A lightweight tabbed file manager used for basic graphical file manipulations (**pcmanfm.sourceforge.net**).
- **Xournal**
  An application for note taking and sketching using the OSWALD's touchscreen (**xournal.sourceforge.net**).
- **MPlayer**
  A media player for Linux that plays most movies and music (**www.mplayerhq.hu/design7/news.html**).

HARDWARE SPECIFICATIONS AND FEATURES

Size and Weight:

- Height: 80mm (3.2").
- Width: 150mm (5.9").
- Depth: 25mm (10").
- Weight: 210g (7.4oz).

Core System:

- Texas Instruments OMAP3530 applications processor 500MHz ARM Cortex-A8 Core, NEON SIMD Coprocessor, 430MHz TMS320C64x+ DSP Core, SGX530 2D/3D graphics processor (OpenGL ES 2.0 compatible).
- 128MB DDR-SDRAM (266MHz).
- 256MB NAND Flash memory.

Display:

- 3.5" resistive touchscreen LCD.
- QVGA resolution (320x240) 24-bit color.
- DVI-out at HD resolution (1024:768).

Audio:

- Texas Instruments TLV320AIC33 stereo audio codec, 24-bit resolution, 96kHz sampling rate, 3-D/bass/treble/EQ/de-emphasis effects.
- Speaker.

Wireless:

- IEEE 802.15.4 ZigBee wireless.

Buttons and Sensors:

- Touchpad.
- Three-axis accelerometer.
- Five-way rocker switch (up, down, left, right and center).
- Six general-purpose buttons.
- Microphone.

Connectors and IO:

- 3.5mm stereo headphone port.
- HDMI (not fully complaint, just as a small DVI port).
- Secure Digital card slot (SDIO- and SDHC-compatible).
- Two full-speed USB host ports (USB-type A).

Power:

- 1300mAh Polymer Li-ion battery (~7-hour battery life).
- Charging via USB or power adapter.

For more information, go to **beaversource.oregonstate.edu/projects/cspfl**.

The OSWALD and the *OSWALD User's Guide* (Photographer: Victor Kuechler)

vehicles. The OSWALD is more robust than a laptop; it has a solid-state drive, which makes it durable. Dr Jensen explains:

> We're envisioning, as we go forward, people will be taking some of these devices, taking them apart and refashioning them into whatever they need them to be, whether it's a mini Web server, file server, media server or whether it's embedded in other things. It has enough processing power to do that, and it has very low energy requirements, which make it ideal for embedding in many different applications. It's a general-purpose computing solution for your home market.

Why Linux?

On the software side, the choice of operating system was much clearer. Linux is a real operating system; it's not a toy. Students take it seriously and see the long-term benefits of learning Linux in depth. Students also benefit from joining Linux's rich open community, where newbies can find guidance and plenty of resources. From a development standpoint, Linux supports many types of software and devices, so it has simplified development. On the downside, little development has occurred on Linux for handheld devices, but this also means students still can make meaningful contributions with relatively modest effort.

Embracing Linux also differentiates the OSWALD from the many media player devices out here, like as the iPod Touch. Although many of these platforms offer a sleeker design and an attractive price point, development for these devices is difficult, as vendors either lock their platforms or apply stringent IP restrictions. In this consideration, the OSWALD has been designed with the flexibility needed to support a wide range of classes. Two USB ports were critical in meeting this design requirement. This component extends the platform in any number of directions, something that no current media player platform allows you to do. With Linux as a foundation, if someone has written a driver for it, the OSWALD can support it. This gives it great flexibility in terms of curriculum design. In freshman courses, students use the OSWALD to control small robots and sensors. In graphics classes, USB cameras enable simple image recognition, and the built-in networking allows students to design P2P protocols as part of their networking classes.

The Future of the OSWALD

A university designing a UMPC like the OSWALD relies on the benevolence and support of both hardware and software partners. To stay on the upside of technological change, the OSWALD developers need to work with the Linux community to make sure their changes and needs are met. Tim Harder, a graduate student in Electrical and Computer Engineering and lead software developer for the OSWALD, puts it like this:

> In open source, you have a lot of updates because a lot of people are working on different things. Currently, we use OpenEmbedded to create the BitBake files needed to create our custom Linux distribution, but an important fix to help us move forward is to find a development environment with rich contributions. We want to have flexibility as well as the high configurability that something like Gentoo provides. Gentoo packages more software for us, and we can take advantage of that.

Moving to a bigger development community, such as Gentoo, will minimize risks in the future.

On the hardware side, fortune smiles with positive support from a number of chip companies who have provided their chips free of charge, at a reduced rate or subsidized the build process. These include Texas Instruments, Tektronix, Intel Corporation and the National Science Foundation. Their support for this device comes from OSU's belief that the OSWALD will help train better engineers and programmers, but also because these partners believe in a substantial future demand for people trained in UMPC and embedded platform development.

Currently, the OSWALD's biggest challenge is broadening the program. Manufacturing hardware for a small community is expensive and risky, and the long-term costs may outweigh the benefits. Therefore, scaling up demand to other universities and even enthusiasts is the key to this device's success. Right now, the OSWALD is available to students and a handful of friends. OSU, like many universities, is not set up for large-scale manufacturing or distribution. It will have to think up ways, as an Open Source community, to support these types of educational efforts in the future.

Acknowledgements

Thanks to Ben Goska, Tim Harder, Techtronix, Texas Instruments, Intel and the National Science Foundation.■

Victor Kuechler is an undergraduate student studying English literature and writing at Oregon State University. He works as technical writer for the department of Electrical Engineering and Computer Science.

Dr Carlos Jensen is an assistant professor in the school of Electrical Engineering and Computer Science at Oregon State University where he leads the Human-computer interaction research group. His research focuses on usable privacy and security, open-source systems and open-source development.

Video Production 101: Making a Movie with Kdenlive

With an inexpensive camera and a Linux system, you can be producer, director, editor and even the grip of your next blockbuster.

DAVE PHILLIPS

By profession, I'm a music teacher. Four times a year, my students perform a show at a local coffeehouse called Coffee Amici. The room hosts regularly scheduled performers, open-mic nights and various community events. My students love to play there. It's a great opportunity for them to show their stuff for an attentive crowd, and eventually, they asked about the possibility of doing video recordings of the shows. I looked into the affordability of the requisite gear, the extent of Linux support for recommended hardware, and the availability of software that would help me produce a simple DVD from my recordings. I was happy to discover that the project was well within my price range and that my software choices included more than one adequate solution.

This article describes how I recorded a student show with an inexpensive camcorder and how I produced a movie from that footage with Kdenlive, a non-linear editor (NLE) for Linux. Kdenlive is a powerful program, but my project was simple, and the methods I describe here should be usable in other Linux NLEs.

Figure 1. Samsung SC-D382 miniDV Camcorder

Learning about the Hardware

First, I needed a video camera. My research and my budget led me to the Samsung SC-D382 miniDV camcorder (Figure 1). For about $100, I got a camera with a remote control, a DV interface (IEEE 1394, aka FireWire) and advanced settings that can be configured for a relatively good-quality recording. It's lightweight, and I found it easy to operate and control. Obviously, more money gets more camera, but I intended to keep expenses as low as possible without compromising the quality of the basic material.

By default, the camcorder produces video in an NTSC 4:3 display format, commonly associated with the square screen of a television in North America. The default audio setting appears to be with a sample rate of 32kHz with a resolution of 12 bits. The camera's advanced settings allow me to set the preferred display format to NTSC 16:9, also known as widescreen, and to increase the audio bit resolution to 16 bits. Alas, the audio sample rate is fixed, so CD-quality sound is not possible with the Samsung, a limitation I overlooked when purchasing the unit.

Of my two studio machines, only one has an integrated IEEE 1394 connection. I planned to use both boxes for video work, so I bought and installed an inexpensive PCI board that added three IEEE 1394 ports to my main machine. If you plan to make a connection with a FireWire-enabled camera, check for a port on your motherboard and make sure you have an available slot for an extension board if you need one. And by the way, even the cables need attention. FireWire cables have a variety of pin configurations, so be sure you have the right connectors on both ends of the cable. For example, my camcorder wants a 4-pin connector, and the ports on my machines require a 6-pin adapter.

I also bought a carrying case for the cameras and a good tripod with bubble levels and a variety of gizmos to position the camera in just about any direction. I've been especially happy with the tripod, and I advise anyone interested in amateur video recording to invest in a decent stand for your camera. Yes, you can solve a lot of problems in your edit stage, but you'll save time and energy if your basic material is centered correctly from a stable position.

I drew the line for my expense account at lighting. The camera provides autocorrection for lighting and focus, and my project didn't need a dedicated light system. I was ready to consider my software requirements.

The Software Side

I had two primary concerns on the software side. I needed to ensure that the kernel versions included the IEEE 1394 modules

and that I would find a video editor user-friendly enough for a total newbie. The modules issue was non-problematic. My main box runs 64 Studio 2.1, a by-now old Debian-based system running on a 2.6.21 kernel compiled for 64-bit hardware and real-time operation, while my secondary 32-bit machine runs a more-modern Ubuntu Jaunty system with a 2.6.29 kernel optimized for real-time performance. Both systems provide the required modules—raw1394, ohci1394, ieee1394—all of which are autoloaded when the machines go through their hardware discovery during the boot process.

The process of finding the right video editor consisted of trying out the available NLEs for Linux to determine which one offered the most amenable work flow. During the process, I wrote a series of profiles on Linux NLEs that you can read at LinuxJournal.com. I worked on a simple project in each editor, and I found that more than one program could serve my purpose. I finally decided that Kdenlive (Figure 2) provided the tools I needed in a layout that was easy for me to understand and operate. For the project described in this article, I used Kdenlive 0.7.6, but since then, I've been building the program from the latest SVN codebase. By the time you read article, this minitutorial Kdenlive should be available in a shiny new version 0.7.7.

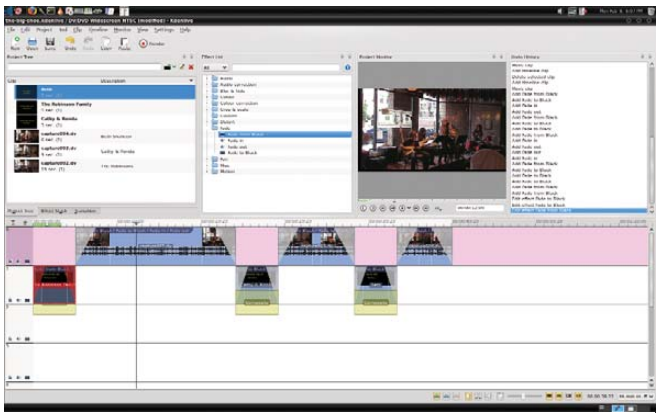


Figure 2. Kdenlive, a Video Editor for Linux

The Shoot

The coffeehouse is a single room with good acoustics and lighting that poses no special problems for a video recording. Coffee Amici also has an excellent in-house sound system manned by co-owner Craig Allen, so I knew the original sound source would be well balanced. I planned to perform with most of the students, so I needed to position the camcorder where I could operate it remotely. I also needed to get a good shot of the performance area and a good audio level. Fortunately, I had no problems with my setup, and the shoot was trouble-free.

Incidentally, I learned that the camera's battery will last just long enough to record a complete 60-minute miniDV tape. The student shows routinely last at least that long and often longer. Wherever possible, I plan to use the camcorder's power supply, but now I know how long I can expect the battery to perform before I need to switch it out for a recharge.

The Transfer

Kdenlive records real-time video over a FireWire connection or from a Video4Linux (V4L)-compliant Webcam. The program also utilizes the RecordMyDesktop software for capturing on-screen actions, such as mouse movements, window placement, program controls and so forth. I used Kdenlive's Record Monitor panel (Figure 3) to transfer the videotape recording to my computer. The panel provides full transport control of my camcorder, and I quickly searched for and found the sections I wanted to transfer to disc. I clicked on the Record control when I wanted to transfer the video, and I clicked it off until the video reached the next interesting point.

Figure 3. Kdenlive's Record Monitor Panel

When I was finished with each transfer, Kdenlive asked if I wanted to import the resulting DV file. I said yes, and my transferred video appeared in the Kdenlive Clip list. When my clip collection was complete, my project was ready for the editing stage.

Into the Editing Room

I dragged and dropped my clips along the timeline in the

multitrack display. I wasn't concerned with their accurate placement yet, because I had three main jobs to do first as an editor:

- Trim excess footage.
- Add audio and video fades to the start and end of each clip.
- Create title clips to introduce each performance clip.

Kdenlive's Scissor tool made quick work of the excess footage. The fade-ins and fade-outs are drag-and-drop effects with user-definable lengths and auto-sync between the audio and video fades, a handy default action. The material had no need for color correction or other repair, so I proceeded to create my title clips.

The name and length of the title clip can be redefined at will, and a titling editor is provided for adding text, images and background colors to the clip. The editor has a number of amenities, including some handy positioning tools and animation effects. However, I intended to use the title clips only for introducing each performer's footage, so I kept things simple again with a yellow text against a basic black background.

After editing my title clips, I could then drag and drop them as needed. Thanks to Kdenlive's Snap function, they aligned themselves instantly when I placed them near their respective footage, yet another friendly default action. I added fades to those clips too, and I was effectively done in the edit room.

Configuring the Output

Kdenlive's Render button opens a dialog for preparing your movie for its eventual output format (Figure 4). I selected the DVD option from the Destination list, named the output VOB file, and defined the format as NTSC. I set the encoder for a 2-pass rendering to a widescreen display, ticked the option to open the DVD Wizard after rendering, and clicked the Render To File command. Next I made some coffee (it seemed the appropriate thing to do) while Kdenlive did the rest.

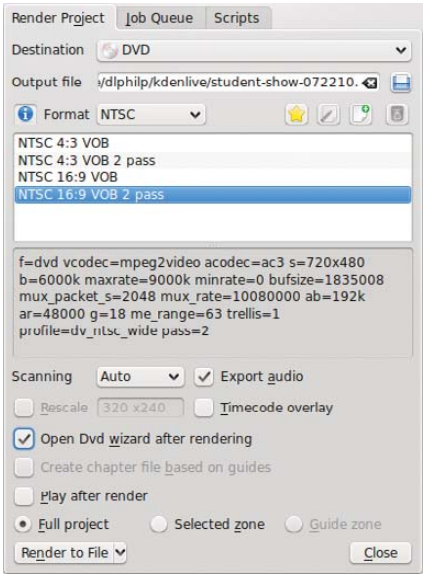


Figure 4. Kdenlive Render Dialog

Making the Disc

The DVD Wizard (Figure 5) stepped me through the process of preparing an ISO image to burn to disc. I selected my files, defined the disc chapters and made a basic menu. The menu builder can

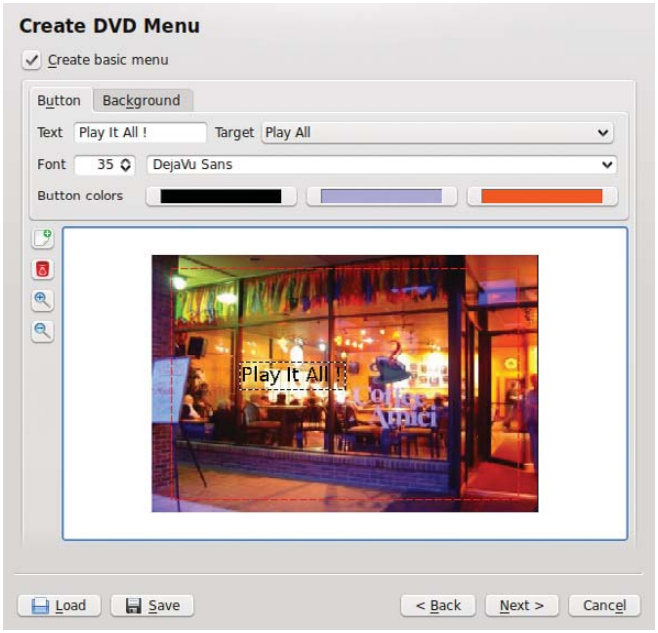


Figure 5. Kdenlive's DVD Wizard

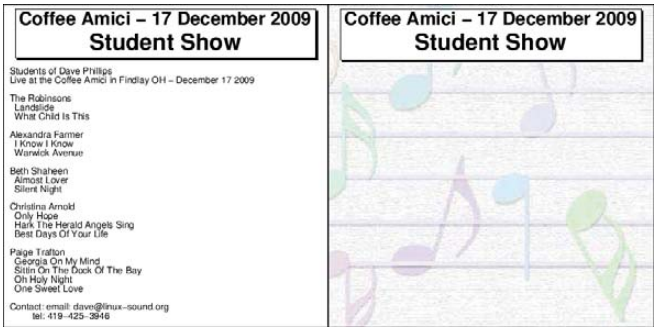


Figure 6. A CD Insert Produced by cdlabelgen

set the background to a color, an image or a video file. I selected an image of the coffeeshop at night and added a text overlay of menu items. Each item is linked to a particular target—that is, one of the chapters defined during the Wizard's configuration. The whole process was uncomplicated enough to bring me quickly to the final stage of creating the DVD image. I clicked on the Create ISO Image and again let Kdenlive do the rest while I finished that coffee I brewed earlier.

The ISO image creation dialog includes an option to start a disc-burning program when the image is ready. I selected the K3B option and followed that program's instructions until I had successfully burned the number of discs I needed. Some discs were specially prepared media for use in my laptop's Lightscribe drive. Templates for making detailed CD/DVD labels are available for various Linux graphics programs, and the Lightscribe technology will burn the label graphics directly to the surface of the special discs, giving them a professional appearance.

Simplicity and ease were my prime directives throughout this

The fade-ins and fade-outs are drag-and-drop effects with user-definable lengths and auto-sync between the audio and video fades, a handy default action.

project, so when I wanted to make inserts for the disc jewel cases, I availed myself of the services of Avinash Chopde's cdlabelgen, an on-line insert creation utility. I selected a background, added titles, songlists and credits, and saved the output as a PDF file to be printed as needed (Figure 6).

Notes for Fellow Novices

I'm definitely an amateur in the video domain, but I'm having a lot of fun learning about it. I've benefited from the advice and suggestions from many people far more knowledgeable, so in keeping with the spirit of sharing helpful information, I've assembled the following notes in the hope that they might benefit other newcomers.

First, take the time to learn about your hardware. You need to know exactly what to expect from it during and after the shoot. Buy an extra battery and keep it charged. Buy a good stand and extra tapes.

On the set, set up your shoot carefully and take extra care for camera position and lighting. Record your audio and video at the highest quality your camera allows. Record your audio at a strong level, but avoid clipping. Normalize audio only as a last resort. Bear in mind that normalization raises the level of the noise floor along with all other sound in the normalized track.

Be generous when transferring your video from camera to computer. Approximately 13GB of storage space is required for 60 minutes of DV-formatted video, but massive storage is cheap these days. Restrict your edits to simple functions. Avoid fancy transitions and don't get distracted by effects. Yes, effects are great fun, but unless they truly add something to the final product they are best left for another project.

Know your target destination (disc, stream, file), and format your rendering options accordingly. Two passes are better than one, but life is short. Do what you can with the time you have, then move on.

The Wrap

I felt that I had done my homework, that I had made my choices as wisely as possible. Nevertheless, I was a newbie at video production, and I expected surprises. As my work progressed, I was surprised indeed, but only at how easily the entire process flowed along. My hardware behaved as expected, Linux provided the necessary low-level support, and the applications software worked without troubles throughout the process.

Kdenlive was a special pleasure. I had no problems with its interface, and its tools and utilities were easy to learn and apply. Since my first project, I've learned more about Kdenlive, and I continue to work with it as my primary video production software. There's far more to the program than the few features presented in this article, so if you're interested in affordable video production with Linux, be sure to check out Kdenlive.

I hope you've enjoyed this little introduction to amateur video production with Linux. I've had a lot of fun making my own movies with my little setup, and I hope to improve its capabilities in the future. It may be some time before I can afford a better camcorder, but in the meanwhile, I can look forward to new releases of Kdenlive to keep me busy.■

Dave Phillips has been using Linux for sound and music since 1995. He is one of the original founders of the Linux Audio Developers/Users groups and has been the maintainer of linux-sound.org for more than ten years. He is the author of *The Book Of Linux Music & Sound* and has written many sound-related articles for various Linux publications. His other activities include playing in a blues band, reading Latin literature, playing with his shar-pei Maximus and spending time with his beloved Ivy. You can hear Dave's music at linux-sound.org/ardour-music.html.

Resources

Kdenlive: **www.kdenlive.org**

K3B: **k3b.plainblack.com**

cdlabelgen: **www.aczoom.com/tools/cdinsert**

Kdenlive Profile: **www.linuxjournal.com/content/kdenlive-meets-studio-dave**

TECH TIPS

Making `ps aux | grep` Work Right

When using `ps aux | grep` to look for processes, I get annoyed when I find the grep process in my search. For example:

```
$ ps aux | grep firefox
user   ...  /usr/lib/firefox-3.5.8/firefox
user   ...  grep --color=auto firefox
```

To avoid this issue, I use a character class regular expression that is only one character long. Simply put, I enclose one of the letters in my search term in brackets, like this:

```
$ ps aux | grep firef[o]x
user   ...  /usr/lib/firefox-3.5.8/firefox
```

This prevents the grep process itself from matching the search term, because the search term is "firefox", but the grep command contains "firef[o]x".

—ROSS LARSON

Extract Audio from the pacpl (Perl Audio Converter) Command-Line Tool

The pacpl command-line tool allows you to extract the audio from any type of video format. The command usage is like so:

```
pacpl -to OUTPUT-FORMAT INPUT-FILE
```

For example, to extract the audio from an .mov video and store it in an .mp3 file, do the following:

```
$ pacpl -to mp3 2010-01-Five_Minutes_SpringRoo.mov

Perl Audio Converter - 4.0.5

Converting: [2010-01-Five_Minutes_SpringRoo.mov] -> [mp3] ..done.

Total files converted: 1, failed: 0
```

The output is stored in the 2010-01-Five_Minutes_SpringRoo.mp3 file in the same directory.

—MAHENDRAN NATARAJAN

Convert PDF Documents to JPEG Images

If you want to convert a PDF document to a JPEG image, first use `pdftoppm` to convert it to a PPM (Portable Pixel Map) file, and then use `ppmtojpeg` to convert it to a JPEG file.

First, convert the PDF:

```
$ pdftoppm input.pdf output
```

This generates one PPM image per PDF page in files named output-N.ppm (where N is the page number). If you want only part of the document, specify the first page to convert with `-f N` and the last page to convert with `-l N`.

Finally, to convert all the PPM files to JPEG images, you can do something like this:

```
$ for file in *.ppm
> do
>   ppmtojpeg $file > ${file/.ppm/.jpg}
>   rm $file
> done
```

—MALICK KHADY DIA

Play Videos Packed in a RAR File without Extracting Them

Most DivX/Xvid movies you download from Torrent sites are packed in multiple RAR archives. It takes some time and space to extract each of them. If you don't want to wait, or use the space, you can use VLC and unrar to play the files without extracting them. You won't be able to rewind and fast-forward within the file, but you'll be able to play and stop the movie without actually unarchiving the video file. Here's how:

```
$ unrar p -inul /path/to/movie_folder/movie.name.r00 | vlc -
```

—MALICK KHADY DIA

Send a tech tip to techtips@linuxjournal.com, and if we publish it in the magazine, we'll send you a free T-shirt.

Solid-State Drives vs. Rotational Media

SSDs are the future, but are they ready for prime time today?

KYLE RANKIN

BILL CHILDERS

BILL: You know, I just got a new 2.5" drive for my laptop.

KYLE: Yeah? What'd you get?

BILL: Picked up a 7200RPM 500GB disk for $100—half a terabyte, man.

KYLE: That's a lot of storage, but really? 7200RPM? That hasn't been a fast rotation speed for about a decade. Why not get a solid-state drive?

BILL: SSDs are too expensive for my taste, and they don't hold nearly enough data. I like carrying my movies, music and docs around with me—despite the fact that I love cloud storage.

KYLE: I suppose they are a little pricey, but you get a huge performance boost with that price. Plus, it's not too expensive to get a 100Gb+ drive, which should be enough for most of the things you'd need on a laptop. Oh, and nice try throwing the cloud in there to distract me. The fact is, many people are using their laptops as their primary machines these days, and for heavy use, 7200RPM laptop drives just don't cut it. That's especially true if you want to play around with any kind of virtualization. Have you ever tried running two VMs at the same time on a 7200RPM laptop drive? I have, and it's painful.

BILL: For heavy use, 7200RPM drives don't cut it? What crack are you on, man? My machines are plenty fast for what I need. I just priced a 128GB SSD, and it's $475! That's nearly 4x the cost for 1/4 the storage. A 256GB SSD is $679. Crazy prices, man. I can buy a whole *laptop* for $679.

KYLE: The prices are a bit high right now, but they keep coming down. Sure, you can buy a whole laptop for $679, or for $1k, you could buy a laptop with twice the performance. With the multicore and other high specs in machines these days, the disk is definitely the bottleneck. If you solve that problem, you can take advantage of the rest of the machine.

BILL: Not to mention there's the whole "not fully baked yet" issue with the technology. Do you remember Intel's boo-boo when it had some firmware glitch that killed performance and ate your data? Rotating media isn't perfect, but its issues are known. I love the smell of baked data in the morning—just like when I tried XFS at your insistence.

KYLE: That's true, rotational media has *never* had a firmware problem nor issues that resulted in data loss. They certainly don't have *any* problems in portable devices like laptops if they get any kind of shock. The fact is, it's common now for laptops to have at least two cores, if not more. Parallel computing is not just the future, it's how we use our computers today, and you need storage that doesn't choke with multiple processes accessing random sections of disk.

BILL: This from a guy who was running a single-core laptop until just a few months ago—with a 1.8" disk, if I recall.

KYLE: Yeah, in fact, it was a 1.8" disk, a 5400RPM one (the fastest they made at the time), until I upgraded it to SSD. The old rotation media in the laptop just didn't cut it.

BILL: I didn't say they haven't had issues. But, SSDs were touted as invulnerable and the greatest thing since sliced bread until that point.

KYLE: I'm sure you still listen to music on vinyl, so no wonder you prefer your storage to use the same technology. At least if you bump your record player, you risk only a scratch and not a head crash.

BILL: Hey, I don't listen to vinyl. Are you not paying attention to our own column? I listen to music on my iPhone, man. And, not all SSDs are super performance-oriented. I have a Netbook with an SSD in it, and the thing is the biggest POS I have ever used.

KYLE: I agree not all SSDs are equal. You have to do a bit of research, just like with video cards or any hardware purchase where performance matters. Oh, and thanks for the lessons on data integrity. I must be remembering some other friend of mine who trashed his rotational laptop drive by putting his magnetic palm pilot case too close to it.

BILL: Yes, I killed my disk. Like you've never smoked a piece of gear. It's funny how recently you got on the performance bandwagon too. I remember you telling me that "a single 1.2GHz core was fast enough", and "I'm

willing to pay the penalty for a slow 1.8" disk for the portability".

KYLE: Yeah, I *was* willing to pay the penalty for portability. Now my main portable computer is an N900, so I can stand to have a slightly larger laptop that can handle a larger drive. Recently, I've tried using it for virtual machines, and the 7200RPM drive it has is a pain. It just chokes when you have too much going on. I'm wishing I had invested in an SSD when I first spec'd it out, but I figured I'd give the 7200RPM drive a chance first.

BILL: Regarding g-forces and rotating media, the new accelerometer-based tech has helped disks survive crazy things, and the smaller, lighter head shave helped too. Anytime you have IO contention, you'll have performance issues. Faster disks move that out. But really, it's not like two VMs are unusable—I have two going on my laptop routinely. Matter of fact, one of them is how I access e-mail (don't ask, I have some totally bizarre corporate antics to put up with).

KYLE: I'm glad they've made the tech equivalent of a seat belt and airbag to save your rotating media, but I'd rather it not be a problem in the first place. I mean, even the iPod figured out SSDs were the way to go years ago. I'd figure if it's good enough for Apple, it'd be good enough for you. The bottom line for me is that my laptop is more than just a portable computer, it's my main computer for anything resource-hungry these days (I use a pocket computer for everything else). So performance matters, especially when I have multiple processes accessing different parts of the disk at once. SSDs are the future, and although they are a bit pricey now, the price keeps coming down and will continue to come down.

BILL: Hey, it's all about price/performance ratio. SSDs appear to have about an 8x premium, and that's not something I'm willing to pay when a 7200RPM disk gets the job done for me and allows me to carry all the data I could possibly want and more. SSDs are the future—yes, I grant you that. But, we live in the *present*, and I'm totally fine with letting early adopters like you find the data-eating bugs and drive the cost down to the point where it does make sense for me to buy. Today, my money's still on good-old rotating media.■

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

Bill Childers is an IT Manager in Silicon Valley, where he lives with his wife and two children. He enjoys Linux far too much, and he probably should get more sun from time to time. In his spare time, he does work with the Gilroy Garlic Festival, but he does not smell like garlic.

EOF

Waving Goodbye to Facebook

We need an alternative. Google Wave is one possibility.

DOC SEARLS

In a blog post last April titled "Building the Social Web Together", Mark Zuckerberg of Facebook wrote, "The power of the open graph is that it helps to create a smarter, personalized Web that gets better with every action taken." The "open graph" of which he speaks is your social graph—your collection of contacts—on Facebook, augmented by "personalized experiences" with the likes of "Microsoft Docs, Yelp and Pandora"—which are Facebook's "three pre-selected partners".

Responding in his own *Newsweek* blog, Barrett Sheridan called Zuckerberg's plans a "Play to Take Over the Entire Internet". In *TechCrunch*, MG Siegler's headline read, "I Think Facebook Just Seized Control Of The Internet". Whether or not Facebook is that ambitious, it won't succeed at anything other than enlarging itself. The limits to that are those of any private architecture. It can get big, but not bigger than the planet. What Facebook has built is the Great Indoors. A lot of people like going there, just like a lot of people like going to shopping malls. But Facebook is a building, not geology.

The Web is geology. It is a wide-open public space on which private and public structures can be built in boundless varieties. Linux is probably the most widely used building material below and within those structures. Calculating its value is pointless, because—as Eric S. Raymond made clear long ago—Linux has use value more than sale value. As useful stuff, its leverage is boundless and, therefore, incalculable. It also will last as long as it remains useful.

The same cannot be said of Facebook, whose value is quite calculable and which will thrive only as long as its revenue model and its investors' patience holds out. Both of those will be shortened by the dissatisfaction of users, which Facebook has been risking increasingly over the years.

To see how this has been going, it helps to check with Facebook's "Eroding Privacy Policy: A Timeline", by the EFF. It shows how, during the five years of its existence, Facebook's privacy policy has ratcheted down from respectful to exploitative. And, why not? Facebook's customers are advertisers, not users. As a user, your influence on Facebook rounds to zero. The company is far more interested in making you into better bait for advertisers and visitors who click on ads. A side benefit is "a smarter, personalized Web" that is not the Web at all, but rather an indoor commercial habitat with some nice conveniences.

We've seen this movie before, many times. The most important and dramatic example is Microsoft's "HailStorm", which arrived and flopped in 2001. As with all heavy weather, it threatened to change (at least superficially) the Web's geology...

> The HailStorm architecture is designed for seamless extensibility and consistency across services. It provides common identity, messaging, naming, navigation, security, role-mapping, data modeling, metering and error handling across all "HailStorm" services. And rather than risk compromising the user-centric model by having advertisers pay for them, the people receiving the value—end users—will be the primary source of revenue. "HailStorm" will help move the Internet to end-user subscriptions, in which users pay for value received.

...by putting the Net itself inside Microsoft's own building. That didn't work. Nor will anything Facebook does for roughly the same purposes.

Of course, Microsoft wasn't talking about the real Internet. It was talking about the commercial activity happening on top of the Net. Likewise, Facebook isn't talking about the Web, but rather the "social networking" that's been all the craze during the past few years, and which seems to be happening mostly within and between commercial entities.

As Facebook seems determined not to learn from the failings of its elders, how about moving past all the commercial interests here? How about making our own social networks—ones that are owned by nobody (or close enough), used by everybody and improved by anybody? How would we do that?

One possibility is Google's Wave. It's a way to meet, collaborate, share files and do other literally social stuff. It's also an open-source project that still needs a lot of shaking down. And, although Google hosts the first incarnation, it's still there for anybody else to run with it, fork it or whatever. Here's how Joel David Palmer summed up the possibilities, in a blog post by Steven Hodson:

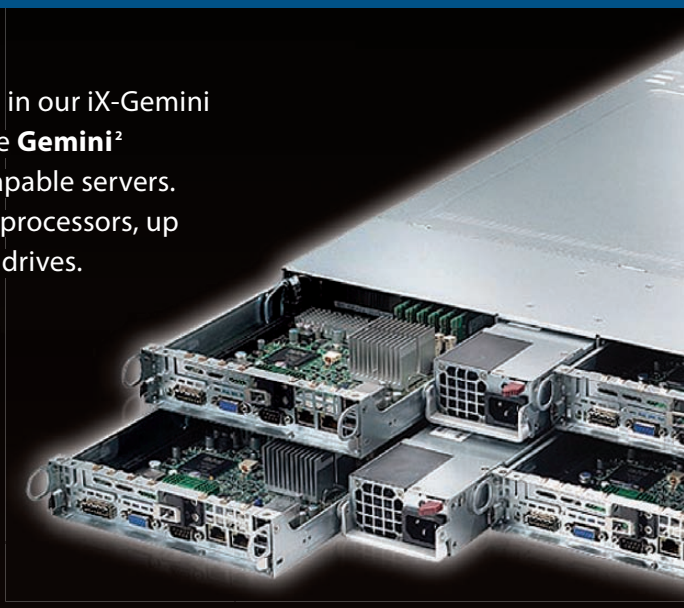
> At the extreme in personal control, we could each configure our own computer as a little wave server and have primary control of our social networking server logs. Less extreme than this, any community or association that's used to serving e-mail could as easily and securely serve waves to their group.
>
> Widespread adoption of waves would automatically reclaim a lot of user privacy and personal responsibility for on-line communications of every kind. The contents cannot be mined by outside interests without committing criminal acts—waves are as private as e-mail.
>
> Waves give you and your connections complete control over your social networking experience, and real opportunities for creative collaboration. A wave can be e-mail, telephone, IM, videophone, collaboration platform, art form, performance venue. Transitioning to waves appears to be a good strategy toward a collaborative social Web that is peer-to-peer rather than server-client.

Sounds good to me. If you've got any better ideas, let's hear them. Maybe we can coax some Facebook occupants—including ourselves, in many cases—to come enjoy the Great Outdoors.■

Doc Searls is Senior Editor of *Linux Journal*. He is also a fellow with the Berkman Center for Internet and Society at Harvard University and the Center for Information Technology and Society at UC Santa Barbara.

Gemini²: The Fantastic Four

IXsystems is proud to introduce the latest offering in our iX-Gemini line, the **Gemini²**. Cleverly disguised as any other 2U server, the **Gemini²** secretly houses 4 highly efficient, extremely powerful RAID 5 capable servers. Each node supports the latest Intel® Xeon® 5600 or 5500 series processors, up to 192GB of DDR3 memory, and three 3.5" hot-swappable hard drives.



This system architecture achieves breakthrough x86 server performance-per-watt (375 GFLOPS/kW) to further satisfy the ever-increasing demands for efficiency, density and low-TCO of today's high performance computing (HPC) clusters and data centers. For more information and pricing, please visit our website at http://www.iXsystems.com/gemini2.

Features

Four hot-pluggable systems (nodes) in a 2U form factor

Each node supports the following:

- Dual 64-Bit Socket 1366 Six-Core, Quad-Core, or Dual-Core, Intel® Xeon® Processor 5600/5500 Series
- 3 x 3.5" SAS/SATA Hot-swappable Drive Bays
- Intel® 5520 Chipset with QuickPath Interconnect (QPI)
- Up to 192GB DDR3 1333/1066/800 SDRAM ECC Registered Memory
- 1 (x16) PCI-E (Low Profile)
- Matrox G200eW 8 MB DDR2 Memory Video
- Integrated Remote Management - IPMI 2.0 + IP-KVM with dedicated LAN
- All four nodes share a Redundant 1200W High-efficiency Power Supply (Gold Level 92%+ power efficiency)

800-820-BSDi
http://www.iXsystems.com
Enterprise Servers for Open Source

Powerful.
Intelligent.

Cool, Fast, Reliable

GPGPU computing for your office and data center

Designed from the ground up for ultimate customer satisfaction, Microway's WhisperStation integrates the latest CPUs with NVIDIA Tesla GPUs. Tesla's massively multi-threaded Fermi architecture, the CUDA™ C and FORTRAN language environments, and OpenCL™ provide the best performance for your application.

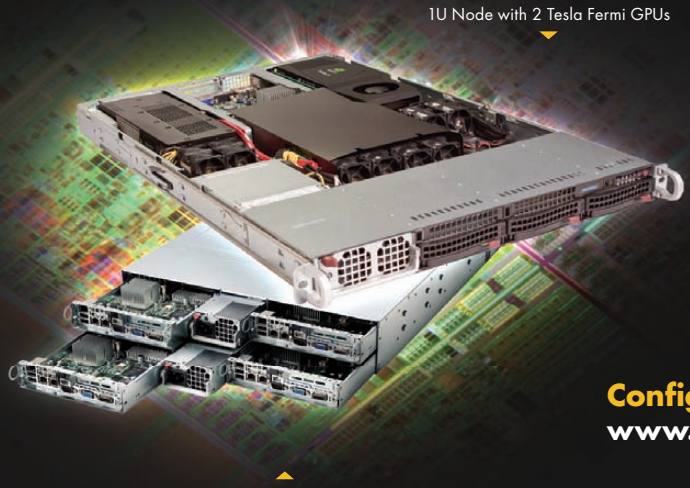
- Up to Four Tesla Fermi GPUs per WhisperStation, with 448 cores and 6 GB GDDR5, each delivering 1 TFLOP single and 515 GFLOP double precision performance
- Up to 24 cores with the newest Intel and AMD Processors, 128 GB memory, 80 PLUS® certified power supply, and eight hard drive
- Nvidia GeForce GTX 480 for state of the art graphics
- Ultra-quiet fans, strategically placed baffles, and internal sound-proofing

WhisperStation with 4 Tesla Fermi GPUs

The Microway Advantage: Custom Integrations and HPC Expertise Since 1982

Put our years of expertise with Linux, Windows, CUDA and OpenCL to work for YOU!

Every Microway system is backed by pre and post sale techs who speak HPC. Whether it's graphics or GPGPU, FORTRAN or MPI, hardware problems or Linux kernel issues; you can talk to Microway's experts to design and support solutions for power hungry applications.



1U Node with 2 Tesla Fermi GPUs

2U Twin² Node with 4 Hot-Swap Motherboards
Each with 2 CPUs and 256 GB

Microway's Latest Servers for Dense Clustering

- 1U nodes with 48 CPU cores, 512 GB and QDR InfiniBand
- 1U nodes with 24 CPU cores, 2 Tesla GPUs and QDR InfiniBand
- 2U Twin² with 4 Hot-Swap MBs, each with 2 Processors + 256 GB
- 1U S2070 servers with 4 Tesla Fermi GPUs

The Fastest CPUs and GPUs Ever

- 12 Core AMD® Opterons with quad channel DDR3 memory
- 8 Core Intel® Xeons with quad channel DDR3 memory
- 448 Core NVIDIA® Tesla™ Fermi GPUs with 6 GB GDDR5 memory

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

NVIDIA Tesla CUDA · Intel Channel Partner Premier 2010 · AMD Fusion Partner Premier · GSA

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

Microway®
Technology you can count on℠