Player Project | Mesh Potato | LTSP | Beagle Board | Boot Times

LINUX JOURNAL™

Since 1994: The Original Magazine of the Linux Community

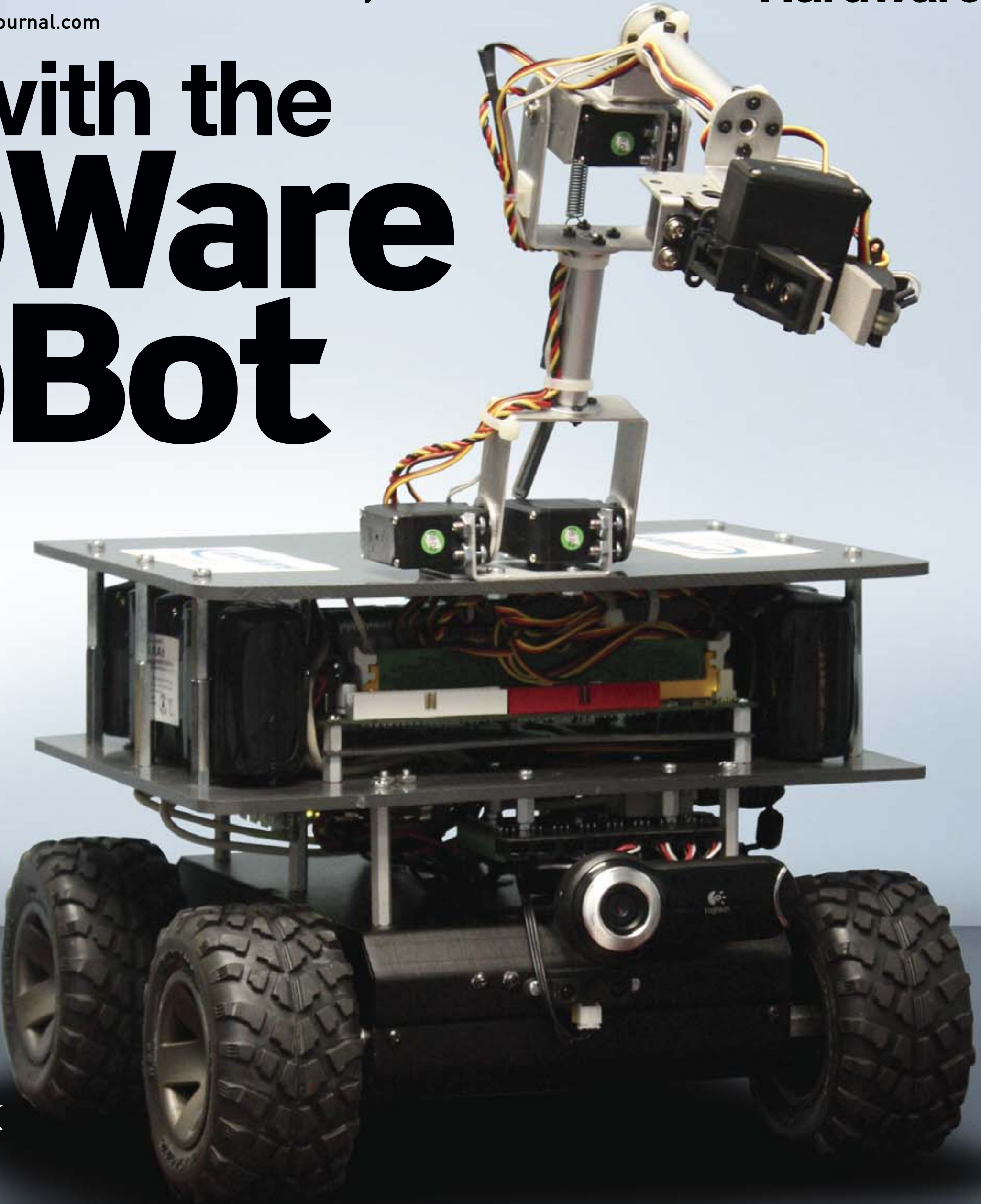
DECEMBER 2009 | ISSUE 188 | www.linuxjournal.com

Intro to Embedded Linux

Multiple OS Desktops

Mesh Potato—Custom Wi-Fi Hardware

Playing with the CoroWare CoroBot



Build a Humidity Controller

Reduce Boot Time on Embedded Systems

NEW COLUMN: Economy Size Geek

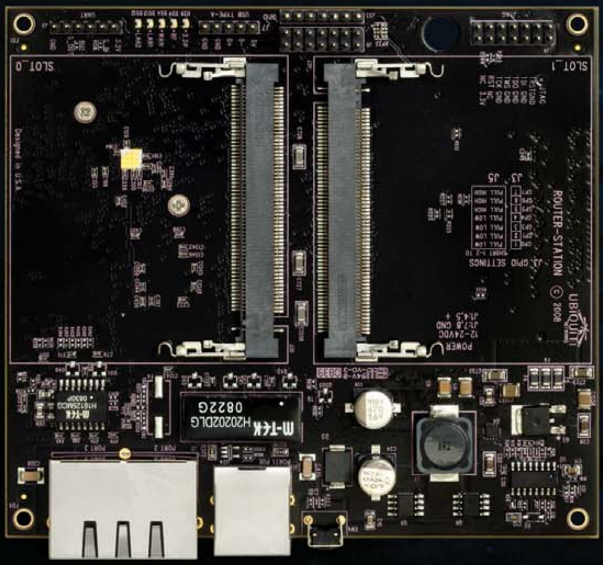
REVIEWED:

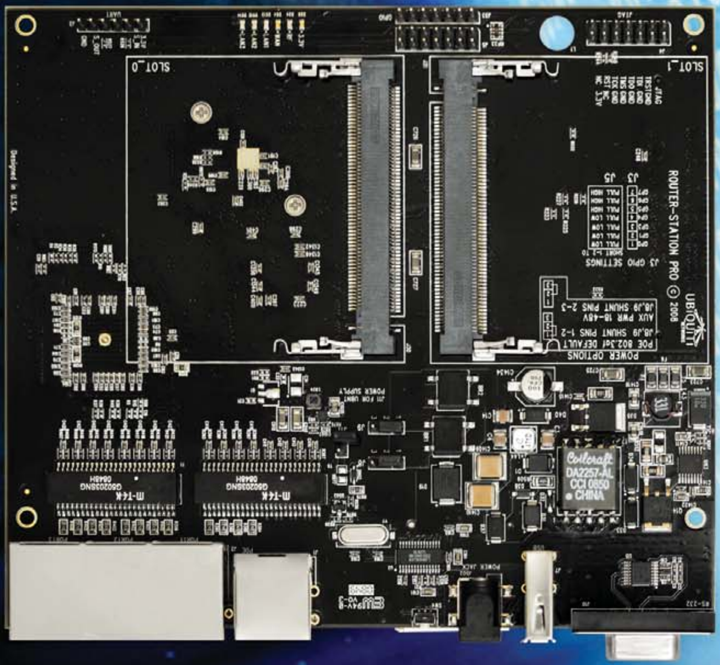
VUZIX VR920 VIDEO GOGGLES

$5.99US $5.99CAN

UBIQUITI NETWORKS

Embedded Wireless Dream Machines.





RouterStation

Featuring a fast 680MHz MIPS 24K CPU, 64MB RAM, and 16MB Flash; RouterStation provides a excellent horsepower for a variety of processor intensive multi-radio system applications.

Up to 3 mini-PCI radios, 3 10/100 ethernet interfaces, a 5A power supply for multiple hi-power card support, USB 2.0, and enhanced temperature operating performance and ethernet ESD protection for carrier applications.

MSRP $59

RouterStation Pro

In response to the outstanding demand for our initial RouterStation OEM platform, Ubiquiti Networks announces the RouterStation Pro. Breakthrough Price/Performance with a $79 USD MSRP.

Pro Version Enhancements:

- 48V 802.3af Power Over Ethernet
- 4-Port Gigabit Ethernet Switch
- 256MB RAM
- On Board SDIO Support
- On Board, USB 2.0, RS232/dB9, and DC power jacks

MSRP $79

www.ubnt.com

9 MILLION CUSTOMERS HAVE VOTED.

THANKS

FOR MAKING US THE WORLD'S #1 WEB HOST!

Your loyalty has helped make us the leading web hosting provider worldwide. 1&1 was built on a foundation of innovative products and outstanding reliability, and we continue to strive to bring you high-quality products at affordable prices. To show our appreciation, we're offering discounts on our most popular products.

DOMAINS

	1&1	Yahoo!	Go Daddy
.com	$8.99	$9.95	$10.69
Private Domain Registraion	FREE	$9.00	$8.99
ICANN Fee	Included	Included	$0.18
E-mail Account	FREE 2 GB Mailbox	NO Mailbox Included	FREE 1 GB Mailbox
Total Annual Cost	~~$8.99~~ $6.99 first year*	$18.95	$19.86

BUSINESS WEBSITES

Powerful website solutions for your small business.

1&1® Business Package

- **3 FREE Domain Names**
- **250 GB Web Space**
- **UNLIMITED Traffic**

~~$9.99 per month~~

3 months FREE!*

More special offers are available online. For details, visit www.1and1.com

THIS COMPANY USES CERTIFIED CLEAN ENERGY bef

*Offers begin November 1, 2009. "3 Months Free" offer valid with a 12 month minimum contract term only. Setup fee and other terms and conditions may 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. © 2009 1&1 Internet, Inc. All rights reserved.

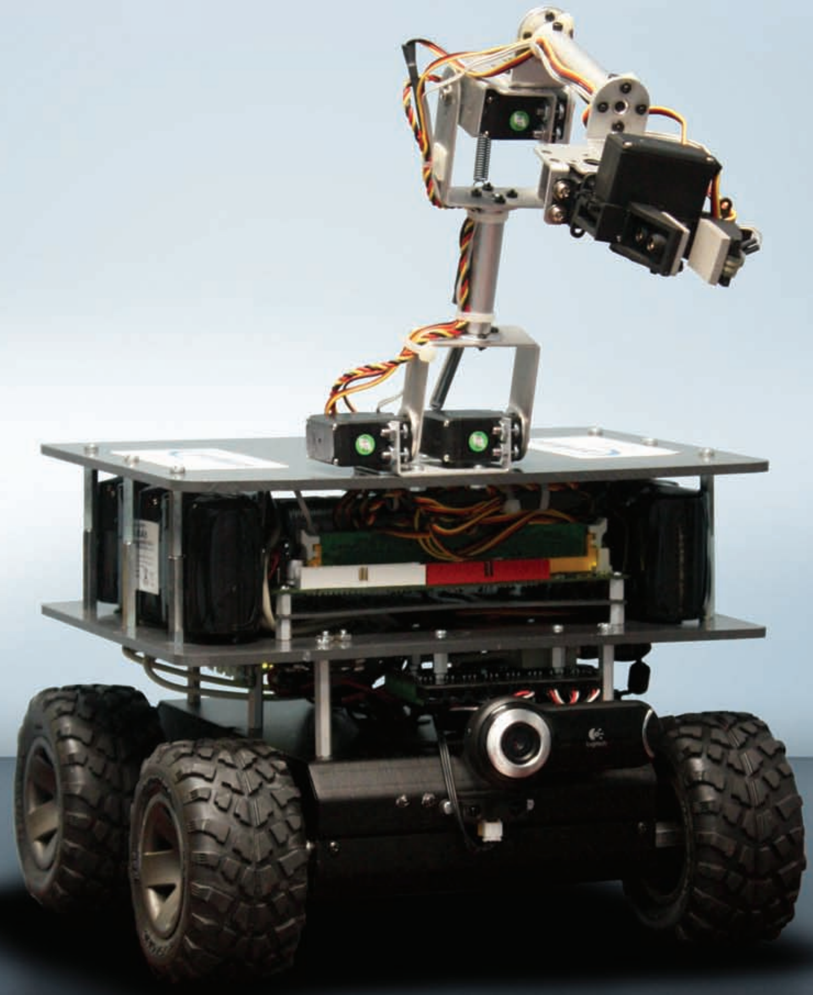
MEMBER OF united internet

Call **1-877-GO-1AND1**

Visit us now **www.1and1.com**

1&1®

CONTENTS DECEMBER 2009 Issue 188



FEATURES

EMBEDDED

44
PLAYING WITH THE PLAYER PROJECT
Programming mobile robots to interface with sensors, actuators and robots.
Kevin Sikorski

50
INTRODUCTION: A TYPICAL EMBEDDED SYSTEM
The common parts that go into an embedded Linux system.
Johan Thelin

54
CONTROLLING THE HUMIDITY WITH AN EMBEDDED LINUX SYSTEM
It's not the heat; it's the humidity.
Jeffrey Ramsey

62
REDUCING BOOT TIME IN EMBEDDED LINUX SYSTEMS
The years fly, but the seconds can drag on forever.
Christopher Hallinan

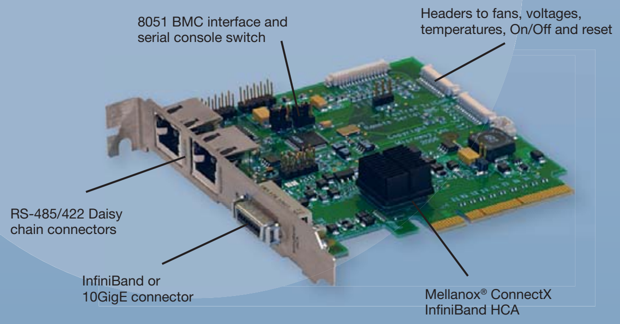
ON THE COVER
- **Intro to Embedded Linux, p. 50**
- **Multiple OS Desktops, p. 73**
- **Mech Potato—Custom Wi-Fi Hardware, p. 68**
- **Playing with the CoroWare CoroBot, p. 44**
- **Build a Humidity Controller, p. 54**
- **Reduce Boot Time on Embedded Systems, p. 62**
- **New Column: Economy Size Geek, p. 30**
- **Reviewed: Vuzix VR920 Video Goggles, p. 40**

Cover image courtesy of CoroWare.

Your Applications Will Run Faster With Next Generation Microway Solutions!

TriCom™X

- QDR/DDR InfiniBand HCA
- ConnectX™ Technology
- 1μsec Latency
- Switchless Serial Console
- NodeWatch™ Remote Management


8051 BMC interface and serial console switch
Headers to fans, voltages, temperatures, On/Off and reset
RS-485/422 Daisy chain connectors
InfiniBand or 10GigE connector
Mellanox® ConnectX InfiniBand HCA

Teraflop GPU Computing

For Workstations and HPC Clusters

- NVIDIA® Tesla™ GPU with 240 Cores on One Chip
  - CUDA™ SDK
- NVIDIA® Quadro® Professional Graphics
- AMD® FireStream™ GPU
  - Stream SDK with Brook+

NumberSmasher®

Large Memory Scalable SMP Server

- Scales to 1 TB of Virtual Shared Memory
- Up to 128 CPU Cores
- 8U System Includes 32 Quad Core CPUs
- QDR 1 μsec Backplane

FasTree™X

- Mellanox® InfiniScale™ IV Technology
- QDR/DDR InfiniBand Switches
- Modular Design
- 4 GB/sec Bandwidth per Port
- QSFP Interconnects
- InfiniScope™ Real Time Diagnostics

Call the HPC Experts at Microway to Design Your Next High-Reliability Linux Cluster or InfiniBand Fabric.

508-746-7341

Sign up for Microway's Newsletter at www.microway.com

Microway®
Technology you can count on℠

CONTENTS DECEMBER 2009 Issue 188

COLUMNS

18 REUVEN M. LERNER'S AT THE FORGE
2009 Book Roundup

22 DAVE TAYLOR'S WORK THE SHELL
Calculating the Distance between Two Latitude/Longitude Points

26 KYLE RANKIN'S HACK AND /
Message for You Sir

30 DIRK ELMENDORF'S ECONOMY SIZE GEEK
A Pico-Sized Platform with Potential

80 DOC SEARLS' EOF
Is "Open Phone" an Oxymoron?

REVIEW

40 THE GOGGLES, THEY DO SOMETHING
Kyle Rankin

INDEPTH

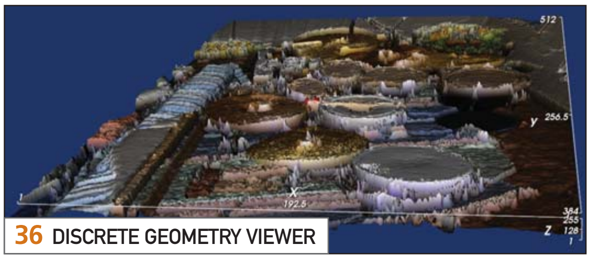
68 THE MESH POTATO
Everybody loves a good Spud!
David Rowe

73 ISOLATED MULTISESSION WORKSTATIONS
Access Ubuntu, Windows, Mac OS and Citrix from the same workstation.
Jorge Salgado

IN EVERY ISSUE

8 CURRENT_ISSUE.TAR.GZ
10 LETTERS
14 UPFRONT
34 NEW PRODUCTS
36 NEW PROJECTS
65 ADVERTISERS INDEX
78 MARKETPLACE

40 VUZIX VR920



36 DISCRETE GEOMETRY VIEWER

68 MESH POTATO

Next Month: AMATEUR RADIO

What do New Orleans, Galveston, Grenada and southern Sichuan have in common? Each was the site of a devastating natural disaster that knocked out communications into and out of the area except for Amateur (Ham) Radio.

What do Guglielmo Marconi and Linus Torvalds have in common? Each had an itch that needed scratching. Marconi's ushered in the era of radio communications and Linus' launched the Open Source revolution.

What happens when Amateur Radio and open source combine? Magical things. From software modems to knowing exactly where you are, and telling the world about it, next month, we are going to look at some of the ways these two areas of innovation have intersected and the results of these interactions.

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

Polywell Linux Solutions

More Choices, Excellent Service, Great Prices!

Serving the Industry for More Than 20 Years

Quiet Storage NAS/SAN/iSCSI

8TB $1,999

12TB $2,599

30TB $6,599

- Dual Gigabit LAN
- RAID-5, 6, 0, 1, 10
- Hot Swap, Hot Spare
- Linux, Windows, Mac
- E-mail Notification
- Tower or Rackmount

The Best Terminal PC

LD-001

Silent Eco Green PC

NVIDIA® GeForce Graphics
Energy efficient, Quiet and Low
Voltage Platform. starts at $199

Mini Fanless PC

ITX-10A Mini-PC $199

with Solid State Drive, DC Power

ITX-20A with SlimDVD

ITX with VESA / Wallmount

ITX-30A with PCI Riser

ITX-1000C with 4LAN
and WiFi Option

Nvidia® ION™ PC
ITX-40A with Blu-Ray or DVD

5U-48Bay 96TB Storage Server

5048A

4U-24Bay 48TB Storage

Hardware RAID-6, NAS/iSCSI/SAN Storage
Mix SAS and SATA, 4 x GigaLAN or 10Gbit LAN

4024SS

Mini-1U Internet Server $499

Intel Dual-Core or Quad-Core Processor,
Dual GigaLAN, 4GB to 8GB RAM,
2 x 500GB RAID HD

1002S-1U945GCL2

We Custom Design Chassis and Motherboard for OEM Customers

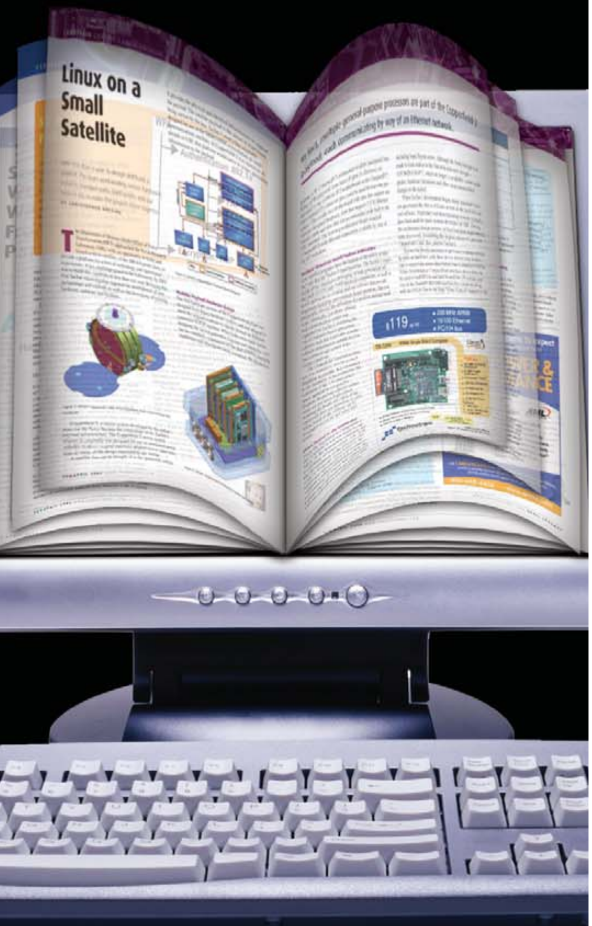
Polywell OEM Services, Your Virtual Manufacturer
Prototype Development with Linux/FreeBSD Support
Small Scale to Mass Production Manufacturing
Fulfillment, Shipping and RMA Repairs

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

888.765.9686
linuxsales@polywell.com
www.polywell.com/us/Lx

POLY
POLYWELL

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



LINUX JOURNAL

Executive Editor	Jill Franklin jill@linuxjournal.com
Senior Editor	Doc Searls doc@linuxjournal.com
Associate Editor	Shawn Powers shawn@linuxjournal.com
Associate Editor	Mitch Frazier mitch@linuxjournal.com
Art Director	Garrick Antikajian garrick@linuxjournal.com
Products Editor	James Gray newproducts@linuxjournal.com
News Editor	Justin Ryan news@linuxjournal.com
Editor Emeritus	Don Marti dmarti@linuxjournal.com
Technical Editor	Michael Baxter mab@cruzio.com
Senior Columnist	Reuven Lerner reuven@lerner.co.il
Security Editor	Mick Bauer mick@visi.com
Hack Editor	Kyle Rankin lj@greenfly.net
Virtual Editor	Bill Childers bill.childers@linuxjournal.com

Contributing Editors
David A. Bandel • Ibrahim Haddad • Robert Love • Zack Brown • Dave Phillips • Marco Fioretti
Ludovic Marcotte • Paul Barry • Paul McKenney • Dave Taylor • Dirk Elmendorf

Proofreader	Geri Gale

Publisher	Carlie Fairchild publisher@linuxjournal.com
General Manager	Rebecca Cassity rebecca@linuxjournal.com
Sales Manager	Joseph Krack joseph@linuxjournal.com
Associate Publisher	Mark Irgang mark@linuxjournal.com
Webmistress	Katherine Druckman webmistress@linuxjournal.com
Accountant	Candy Beauchamp acct@linuxjournal.com

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

Reader Advisory Panel
Brad Abram Baillio • Nick Baronian • Hari Boukis • Caleb S. Cullen • Steve Case
Kalyana Krishna Chadalavada • Keir Davis • Adam M. Dutko • Michael Eager • Nick Faltys • Ken Firestone
Dennis Franklin Frey • Victor Gregorio • Kristian Erik • Hermansen • Philip Jacob • Jay Kruizenga
David A. Lane • Steve Marquez • Dave McAllister • Craig Oda • Rob Orsini • Jeffrey D. Parent
Wayne D. Powel • Shawn Powers • Mike Roberts • Draciron Smith • Chris D. Stark • Patrick Swartz

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

Advertising
E-MAIL: ads@linuxjournal.com
URL: www.linuxjournal.com/advertising
PHONE: +1 713-344-1956 ext. 2

Subscriptions
E-MAIL: subs@linuxjournal.com
URL: www.linuxjournal.com/subscribe
PHONE: +1 818-487-2089
FAX: +1 818-487-4550
TOLL-FREE: 1-888-66-LINUX
MAIL: PO Box 16476, North Hollywood, CA 91615-9911 USA
Please allow 4–6 weeks for processing address changes and orders
PRINTED IN USA

LINUX is a registered trademark of Linus Torvalds.

PRINTED WITH SOY INK®

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® 5500 series processors, up to 48GB 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 Quad-Core or Dual-Core Intel® Xeon® Processor 5500 Series
- 3 x 3.5" SAS/SATA Hot-swappable Drive Bays
- Intel® 5520 Chipset with QuickPath Interconnect (QPI)
- Up to 48GB 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)

iX systems™

800-820-BSDI
http://www.iXsystems.com
Enterprise Servers for Open Source

intel® Xeon® inside™
Powerful. Intelligent.

SHAWN POWERS

Snug as a Bug in a Beagle Board

Here at *Linux Journal*, we take Linux very seriously. In fact, even my kids are embedded. No really, I tucked them in about 20 minutes ago. They're almost asleep. (See, every month you think Shawn can't possibly have another cheesy issue-focus-related joke.) Seriously though, when it comes to Linux, the biggest area of growth is in the tiniest of spaces—the embedded market. It sort of reminds me of a B spy movie:

> *Linux quietly takes over more and more devices; then one day Bill Gates wakes up in a hospital room.*
>
> Mr Gates: What happened, did the surgery go wrong? Did my pacemaker quit working?
>
> Nurse: No Mr Gates, it's working just fine. Your pacemaker is running Linux...
>
> Mr Gates: Noooooooooo!!!!!!

Okay, perhaps I should stick to editorials and not try my hand at writing movie scripts. With or without my movie, however, Linux really does dominate the embedded market. This month, we've got a wide variety of articles to help you learn more about developing for such markets or to introduce you to embedded Linux for the first time.

What better way to attract everyone's interest than with a remote-controlled rover? Kevin Sikorski tells us all about the Player Project. It's a software framework for interfacing with PC-based robots. For most of us, sending up missions like the Mars Rovers might be a bit much. However, after reading Kevin's article, a "Backyard Rover" might be possible—or at the very least, a robot that could find and fetch missing left socks from the laundry room. If you're not quite up to such a daunting task and would like to learn a little more about how embedded Linux systems actually work, you might want to read Johan Thelin's article first. It introduces the concepts and functions of embedded systems—possibly a primer course for your robot-building adventure.

If robots were the only type of embedded projects that ran Linux, it would be rough to devote an entire issue to it. The beauty of embedded Linux is that it can be so diverse. For instance, David Rowe shows us the Mesh Potato. It may seem like a simple wireless access point, but thanks to the operating system underneath, it's a lot more. This month, even Kyle Rankin is trying to embed Linux—onto his face, in fact. Check out his review of the Vuzix video goggles and see if they make him more awesome or more likely to be kidnapped by Romulans.

Don't worry if embedded Linux doesn't really tickle your fancy. Dirk Elmendorf shows us the Beagle Board this month. It's a full-blown computer system that you probably could fit in your wallet. Granted, it would make interacting with it difficult, and it likely wouldn't survive too many sit-downs, but because the Beagle Board is so tiny, hiding it in places a little more useful should be simple.

We have lots of non-embedded material for you this month too. Whether you want to run multiple operating systems from the convenience of a thin client (Jorge Salgado shows you how) or check your e-mail every minute of every day (Kyle Rankin explains a method you've probably never seen before), this month should satisfy. Plus, we've got book reviews from Reuven M. Lerner, shell scripting longitude and latitude from Dave Taylor and lots of new products from James Gray.

So make some chamomile tea, turn down the thermostat (possibly with the help of Jeffery Ramsey's article on controlling room humidity with an embedded system), and grab this month's issue of *Linux Journal*. Carefully crawl into bed and embed yourself under the covers. Enjoy an exciting month of projects and read well into the morning. If you have a hard time booting up in the morning, Christopher Hallinan's article on reducing boot time in embedded systems might help. Or, call in sick with your cell phone. It might just be running embedded Linux too.■

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.

MORE PRODUCTS, BETTER SERVICE, GUARANTEED.

GO STRAIGHT TO THE SOURCE!

1.877.727.7887 | www.ServersDirect.com

YOUR HIGH PERFORMANCE COMPUTING HAS ARRIVED.

The ServersDirect® Systems with the Intel® Xeon® Processor helps you simplify computing operations, accelerate performance and accomplish more in less time

STARTING AT **$899**

ENTRY LEVEL INTELLIGENT SERVER

SDR-S1341-T00 is among our most cost-effective 1U Xeon Servers, and it is ideal for large high-performance computing deployments

STARTING AT **$959**

APPLICATION SERVER

Refresh your servers with new **SDR-S1337-T02** powered by Intel® Xeon® processor 5500 series, based on intelligent performance, automated energy efficiency and flexible virtualization.

SDR-S1343-T04

STARTING AT **$1,099**

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

- Supermicro 1U Rackmount Server with 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

SDR-S2311-T08

STARTING AT **$1,159**

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

- Supermicro 2U Rackmount Server with 560W Power Supply
- Supermicro Server Board w/Intel® 5500 Chipset
- Support up to Dual Intel® 5500 series Xeon® Quad/Dual-Core, with QPI up to 6.4 GT/s
- Support up to 24GB DDR3 1333/ 1066/ 800MHz ECC Reg.DIMM
- 8x 3.5" Hot-swap SATA Drive Bays
- Dual Intel® 82574L Gigabit Ethernet Controller

SDP-IP308-T10

STARTING AT **$1,599**

PEDESTAL INTEL® XEON® PROCESSORS 5500 SERIES SERVER W/ 10X HOT-SWAP (OPT.) SATA BAYS

- Intel Pedestal Chassis w/ 750W (1+1) 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 96GB DDR3 1333/ 1066/ 800MHz ECC Reg./unbuffered DIMM
- Option 10x 3.5" Hot-swap SATA Bays
- Intel® 8257EB Dual-port Gigabit Ethernet Controller

SDR-S4313-T24

STARTING AT **$1,899**

4U INTEL® XEON® PROCESSORS 5500 SERIES SERVER W/ 24X 3.5" HOT-SWAP SAS/SATA BAYS

- Supermicro 4U Rackmount 900W (1+1) Red. 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 144GB DDR3 1333/ 1066/ 800MHz ECC Reg. DIMM
- 24x 3.5" Hot-swap SATA Drive Bay
- Intel® 82576 Dual-port Gigabit Ethernet Controller

SDR-S3305-T16

STARTING AT **$1,979**

3U INTEL® XEON® PROCESSORS 5500 SERIES SERVER W/ 16X 3.5" HOT-SWAP SAS/SATA BAYS

- 3U Rackmount Server with 1+1 900W Red. 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 96GB DDR3 1333/ 1066/ 800MHz ECC Reg.DIMM
- 16x Hot-swap SAS/SATA Drive Bays
- Intel® Dual 82576 Dual-Port Gigabit Ethernet (4 ports)

SDR-C9303-T50

STARTING AT **$4,339**

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

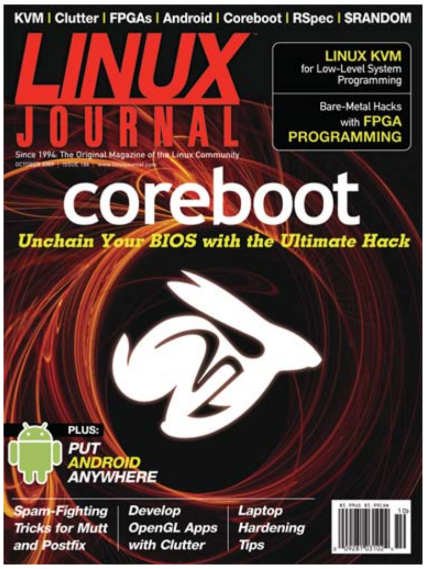
- 9U Chassis with 1620W Redundant Power Supply
- Supermicro Server Board w/ Dual Intel® 5520 Chipsets
- Support up to Dual Intel® 5500 series Xeon® Quad/Dual-Core, with QPI up to 6.4 GT/s
- Support up to 144GB DDR3 1333/ 1066/ 800MHz ECC Reg. DIMM
- 50 x 3.5"Internal SATA Drives Trays
- Intel® 82576 Dual-port Gigabit Ethernet Controller

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

Powerful. Intelligent.



Simple iptables Rules for Potentially Hostile Networks

Regarding the October 2009 issue's article on hostile network protection [see Mick Bauer's "Brutally Practical Linux Desktop Security"], I've also found the following iptables rules render my laptop effectively invisible without adversely affecting Web browsing, e-mail, SSH and nearly everything else I do from hotel rooms or while drinking my morning coffee:

```
iptables -A INPUT -i lo -j ACCEPT
iptables -A INPUT -m state --state
 ➥RELATED,ESTABLISHED -j ACCEPT
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT ACCEPT
```

While most canned kernels will come with everything needed, those who prefer to roll their own will need to ensure that Netfilter and its connection state matching component (NETFILTER_XT_MATCH_STATE) are required. Many of the other Netfilter modules can make life easier as well, so they're worth looking through if you'll need to recompile anyway.

Some of the messier protocols may not work through these rules, so it's best to stay within your distro's init.d firewall script to make them easy to turn on and off. Thanks for the great work guys!

--
E. Stuart Hicks

Normally I just don my invisibility cloak when I'm using public Wi-Fi, but your method is certainly more reproducible. And, it works outside of fantasy novels. All joking aside, I usually set up a VPN the moment I connect to public access points. Unfortunately, that can adversely affect bandwidth. Thanks for the tip on how to be a bit more stealthy.—Ed.

Linux Already Wins the Desktop

I am finishing a a Master's degree in Education and have generated many pages of various types of documentation in completing my work. I cannot imagine doing this with anything but Linux. The multiple desktops available in Linux made it easy to generate a research report while having multiple on-line journals open, simultaneously generating and inserting graphics into the report, without the multiple layers of overlapping windows the popular operating systems force you to use. It seems to me that the true utility of Linux gets lost when we compare ourselves to Windows and Mac, rather than setting the unique and useful aspects of Linux as metrics for Windows and Mac to meet. For me, the multiple desktop function is one of the single-most useful utilities of Linux in getting real work done. I cannot operate with a single window open. I wonder how many other Linux users recognize this as one of many important strengths of our favorite operating system?

--
Orlando Ide

Whenever I'm speaking about Linux to a group of enthusiasts, I stress that Linux is awesome enough to stand on its own. When it comes to the desktop experience, you're absolutely right, Linux has nothing to prove. If we can eliminate the dependence on proprietary software, I think Linux will be the obvious choice for most people on the desktop.—Ed.

Using RCS for Configure Files

David Penman, in the September 2009 Letters section, talked about something very important in system administration. I don't see it enough. When you modify system configuration files as root, always make a backup. RCS is a fantastic tool to see how a file changed—it takes discipline. Just make an RCS/ directory; you don't even see the backup files. But often changing configuration files has negative effects weeks after changing them. Rolling back to the original package is a last-ditch effort.

--
Marty Leisner

Ubuntu 9.04 and Modems

I've been reading your magazine and learning/using Linux on two desktops and purchased a Dell notebook with Ubuntu. Then, when my old desktop PC died, I bought a Dell 530/Vista because it was on sale (Dell's computers with Ubuntu don't seem to go on sale). I installed a second hard drive and proceeded to install Ubuntu 9.04. Imagine my surprise to find no modem support! I downloaded, but could not get gnome-network-admin to work. I wasted hours of time downloading/installing GNOME ppp and dependencies and configuring the modem. I had to download files with Vista and xfer to Ubuntu with a thumb-drive. Vista worked out of the box with its included Dell modem. Vista also worked out of the box with my US Robotics PCI modem (I didn't need to install any software). Ubuntu's decision to break or not offer the modem software seems to be a foolish thing to do—especially if they intend to reach out to the nongeek PC users. And we wonder why we can't get more people to use Linux on the desktop. I know I'm just one small voice in the Linux community. Thanks for reading!

--
Duane G.

I must admit it's been quite a few years since I've used dial-up networking, but it is sad you had such a hard time setting up your modem! I know in years past "winmodems" were very difficult to configure due to Windows-only drivers.

Now it seems the frustration is with Windows-only Wi-Fi drivers. It seems like a conspiracy to keep Linux users from communicating! It sounds like you did get things going, but hopefully the Ubuntu team won't forget about the many folks still using dial-up.—Ed.

Linux on the Desktop, Continued

I read with interest the continued discussion regarding Linux on the desktop [see the September 2009 Letters]. I am old enough to remember the OS/2 vs. Windows war. In those days, lots and lots of Microsofties were unleashed onto an unsuspecting Usenet; their job was to portray ordinary users trashing OS/2 and defending Windows. The two letters you published look like MS is doing again what it is known to have done before, only this time it is trashing Linux instead of OS/2. The incredulity of the original assertion (Linux lacks stability) is what makes me strongly suspect MS operatives are at work here. Back then, IBM didn't know what hit them. This old adage rings true: "Fool me once shame on you, fool me twice shame on me."

--
Robert Solomon

Desktop Hardening

Re: Mick Bauer's "Brutally Practical Linux Desktop Security" [October 2009 issue]: why not make the target for an aggressor as small as possible—a kernel with only the drivers and modules your laptop needs? A filesystem like debootstrap or your distro's base system? It's much less exposure, as you have installed only what you use from the hardware up.

Thanks for all the fine Paranoid Penguin articles Mick. Editor, I would like to see more meat in the diet.

--
Charles Hewson

Mick Bauer replies: *One cool thing about loadable kernel modules is that when you don't have a given piece of hardware attached, the corresponding modules generally won't load. But I get your broader point that just as unnecessary userspace software should be uninstalled or disabled, so should unnecessary kernel code—you're quite correct that hardening is about minimizing your attack surface.*

I've long advocated running custom-compiled kernels on bastion servers for that very reason. But in my article's specific scenario of preparing a laptop for a trip, that might be more trouble than it's worth (especially given my earlier point). It's the difference between spending 45 minutes or less hardening your system and spending hours. For most users (certainly for nonexperts), compiling kernels remains one of the uglier and more time-consuming parts of the Linux experience.

Thanks so much for your kind words! We're all doing what we can to maintain and even improve LJ*'s protein-to-carb ratio.*

OtherInbox.com

Having used a similar method to what Kyle Rankin describes in "Spam: the Ham Hack" [October 2009], I'm happy to have found OtherInbox.com, which automates most of the process. You can use it with your own domain or with their own and a personalized subdomain. You can create an e-mail address on the fly, and it automatically will create a corresponding mailbox. I encourage people who are having trouble managing their e-mail to check it out.

--
Josh Bernstein

Make My Headphones Work

I have used Linux since Slackware 0.91, but I still have trouble getting headphones to work. I have the latest Ubuntu and just expected that when I plugged in my new Logitech headphones, they would work automatically and all sound would go to/through them. How do I make that happen?

--
Eric

At LinuxCon in September 2009, I heard the kernel developers speak of this very issue. Apparently, audio hardware is one of those things that is so inconsistently built, getting all the different revisions to work proves to be very difficult. With Windows, you can download a specific driver from the vendor, but as Linux users, we must depend on drivers based on "standards" that should be built in to hardware. Sadly, those standards rarely are in place. Sometimes it's possible to Google for a specific hardware configuration and find settings to tweak in order to make things like headphones work. Either way, it's frustrating as an end user to have something as simple as headphones not work.—Ed.

Dark Days?

I am not a computer specialist, nor do I have any interest in computer code. But, I use a computer most of the day, every day. Having been stuck with Windows (which I don't like because of the way everything I do is controlled by Microsoft), I recently bought a small laptop with Linux as the operating system. It is an absolute disaster area. To start, it is incompatible with 3 mobile broadband (I have read a number of blogs, and even the experts agree on that). I have had no success in loading Java, which is essential for the work I do. And, I can't even load a 56k modem for emergency use. In short, it is totally useless to me, and I am going to have to load up Windows XP instead, much against my wishes. I had hoped that Linux was a serious competitor to Microsoft, but in reality, it is light-years away, strictly for computer specialists. Of course, I could spend days and days reading about how to make it work, but why should I? I only want to use the computer, not re-invent it. Kernels, shells, command prompts—these things are of no interest to me whatsoever. It's back to the dark days of MS-DOS all over again.

--
Richard

I'm sorry to hear you're having such a bad Linux experience. You should be able to install Java on your laptop without a problem. The Sun distribution works fine on my Linux system. I also see indications on the Internet that people have been able to get 3 mobile broadband to work with Linux. Modems shouldn't be a problem either. Without knowing more about what distribution you have and what hardware you have, it's hard to be much more specific.

Concerning your remarks about the command line and the dark days of MS-DOS, I always find these types of comments interesting, because in my opinion, Microsoft took a giant step backward when it decided to poo-poo the command line. A decent shell (which command.com and/or cmd.exe never were) and a good complement of shell commands, at least for certain types of work, give you power that doesn't exist anywhere in the GUI world.

Having said all that and implied much more, in no way should it be taken that I think Linux is perfect. It's not. But by the same token, Windows has its own set of problems. I often find it as frustrating to work with as you're finding Linux to be.

If you'd like to post some of the details of your Linux troubles on the LinuxJournal.com forums, we'll try our best to help you through them.—Ed.

"The Usual" sudo?

I was just reading John Knight's "Fresh from the Labs", specifically the article on htop, in the October 2009 issue. htop is great, and I have been using it for quite some time. To quote from the article: "...enter the usual:"

```
$ ./configure
$ make
$ sudo make install
```

"the usual"? I do not use sudo, and I do not use Ubuntu. A minor thing, I agree. Today it just annoyed me. Thanks for a great magazine.

PS. Yes, I work for Mandriva, but it's not the only distro I use. I also use Slackware, Fedora and Absolute Linux.

--
Stephen Germany

John Knight replies: *An angry letter, at last! This is my first one for LJ. I thought it'd come from a Debian developer though (I've been stirring them up for several years)....*

htop's brilliant, isn't it? Yes, I know what you mean about Ubuntu-isation of Linux, and it annoys me too, but isn't sudo on most modern distros, and its use encouraged? Note that sudo isn't a Ubuntu invention (quote from Wikipedia): "The program was originally written by Bob Coggeshall and Cliff Spencer around 1980 at the Department of Computer Science at SUNY/Buffalo. The current version is under active development and is maintained by OpenBSD developer Todd C. Miller and distributed under a BSD-style license."

I can't speak for Oklahoma, but here in Australia in the LUGs, the use of sudo is more or less assumed, and the use of root logins discouraged (and strangely enough, the local LUGgers seem to gravitate toward Debian). Nevertheless, I used to write "(as root or sudo)" before the make install command, but figured it was about time just to use sudo for cleanliness' sake. Do you think I should switch back?

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.

For the summertime, I went on holiday to the Dutch coast with my family. For a day on the beach, we took along typical beach stuff, like a windscreen, kites, food and sand toys and, of course, several *Linux Journal* issues to do some interesting Linux reading. Submitted by Geert Jan Klinkhamer.

LINUX JOURNAL

At Your Service

MAGAZINE

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

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

LETTERS TO THE EDITOR: We welcome your letters and encourage you to submit them at **www.linuxjournal.com/contact** or mail them to Linux Journal, 1752 NW Market Street, #200, Seattle, WA 98107 USA. Letters may be edited for space and clarity.

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

ADVERTISING: *Linux Journal* is a great resource for readers and advertisers alike. Request a media kit, view our current editorial calendar and advertising due dates, or learn more about other advertising and marketing opportunities by visiting us on-line, **www.linuxjournal.com/advertising**. Contact us directly for further information, ads@linuxjournal.com or +1 713-344-1956 ext. 2.

ON-LINE

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

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

ABERDEEN
SERVERS AND STORAGE

WHAT'S THE DEAL WITH THESE GUYS?

Sometimes you have to ask, "What are they thinking?"

Companies need to increase ROI without being taken to the cleaners by manufacturers selling servers featuring entry-level benefits with enterprise-level pricing.

Aberdeen gets it. Businesses are in desperate need of Network Attached Storage servers that simply deliver the best bang for the buck.

Look at these features and benefits:

	Dell PowerVault	HP StorageWorks	Aberdeen AberNAS
Hot-Swap Disk Drives	✓	✓	✓
Hardware RAID	✓	✓	✓
Dual Port Gigabit Ethernet	✓	✓	✓
Built-in Replication	✓	✓	✓
Microsoft® WSS 2008 Models	✓	✓	✓
iSCSI Target	✗	✓	✓
Linux Storage System Models	✗	✓	✓
System Recovery Disk	✗	✓	✓
DAS Storage Expansion	✗	✓	✓
VMware® Ready Certified	✗	✗	✓
Independent OS Drive	✗	✗	✓
Out of Band RAID Management	✗	✗	✓
Available w/ 2TB Drives	✗	✗	✓
Warranty	3 Years	3 Years	5 Years

Who gives you the best bang for the buck?

	Dell PowerVault NX300	HP StorageWorks X1400	Aberdeen AberNAS 163
Intel® Xeon® Processor	E5504 2GHz	E5504 2GHz	E5504 2GHz
Memory	3GB	2GB	3GB
Drive Interface	SATA	SATA	SATA
Installed Capacity	2TB	2TB	2TB
Rails	Included	Included	Included
Windows Storage Server 2008	$3,419	$4,635	$2,995
Linux Storage System	Not Available	Not Available	$2,995

intel® Xeon® inside™

Powerful. Intelligent.

Prices for the above specific configurations obtained from the respective websites on Oct. 12, 2009. Intel, Intel Logo, Intel Inside, Intel Inside Logo, Pentium, Xeon, and Xeon Inside are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries. VMware is a registered trademark or trademark of VMware, Inc. in the United States and/or other jurisdictions. For terms and conditions, please see www.aberdeeninc.com/abpoly/abterms.htm. lj032

888-297-7409
www.aberdeeninc.com/lj032

diff -u
WHAT'S NEW IN KERNEL DEVELOPMENT

The **big kernel lock** (BKL) is being removed from **ReiserFS**. In fact, so much work is going into that removal, the Reiser people had to create a new git tree just to handle it. **Frederic Weisbecker** has been submitting patches from that tree to the main kernel repository. As it turns out, it's not necessary to do overly much testing of patches going into the kernel. It's expected that on their way through the linux-next tree and into the distributions, they'll receive a far wider range of testing than people could do on their own.

John Hawley has revised the code that produces all the content on the **kernel.org** Web site. He'd been hearing a lot of complaints that the -mm tree and the linux-next tree were not linked from the home page, while old 2.2 kernels still were being shown. John's changes have eliminated the links to 2.2 kernels and added a link to the linux-next tree. At **Andrew Morton**'s urging, the -mm tree was left out, because his ultimate goal with that tree is to move all of its patches into linux-next and discontinue maintenance of -mm entirely. This change actually was more difficult than you might expect, because the code had all previously been written with a certain development cycle in mind, and that development cycle no longer existed. To fix the page, John essentially had to toss all the old code and rewrite it from scratch.

The **ARM mailing lists** no longer will be maintained by **Russell King**. He got fed up with people complaining about the requirement that no one could post who was not a member of a given list. One day, he just turned off all the lists, prompting harsh criticism from folks like **David S. Miller**, **Alan Cox** and **Theodore Y. Tso**. **Pavel Machek** created equivalent mailing lists on the **vger server**, and **David Woodhouse** set up his own set of mailing lists on **infradead.org**. The result was a lot of confusion, as users didn't know to which mailing lists to post. Ultimately, it'll all get sorted out, but Russell probably should have announced a search for a new mailing-list maintainer and had an orderly transfer of power. He possibly was just too stressed out by the conflict to do that.

Zachary Amsden made an attempt to write code supporting a userspace block device. He called it **aBUSE**, intending it to take its place next to **FUSE** (Filesystem in USErspace) and **CUSE** (Character device in USErspace). But, as Alan Cox pointed out, Zachary's code was a little too similar to **NBD** (Network Block Device). So even though aBUSE was a bit simpler than NBD, Zachary ultimately decided that the benefit wasn't worth creating something that was just so similar. But, he left the aBUSE patches in the e-mail archive for people to pick up if they are interested in it in the future.

Jonathan Cameron is pushing to get **IIO** (industrial input/output) included in the official sources. IIO is a subsystem for controlling hardware sensors like gyroscopes and light sensors. It looks like **Greg Kroah-Hartman** is letting the code go into -staging in the main tree, but he wants to see a detailed to-do list, explaining all the steps to get from -staging to the main kernel.

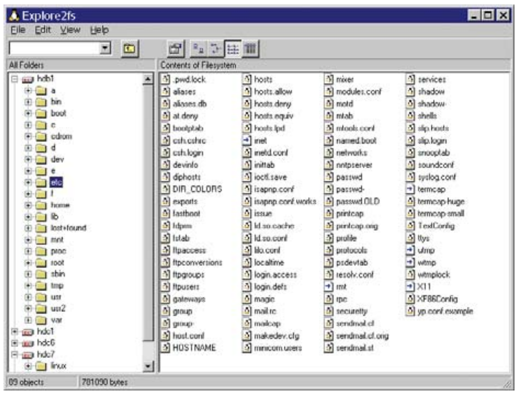
—ZACK BROWN

LJ Index December 2009

1. Percent of users using Firefox (2.0, 3.0 and 3.5): **32**
2. Percent of users using Internet Explorer (6.0, 7.0 and 8.0): **52**
3. Percent of users using Chrome 2.0: **3**
4. Percent of users using Windows: **86**
5. Percent of users using Mac OS X: **7**
6. Percent of users using Linux: **2**
7. Percent of users using 1024x768 screen resolution: **31**
8. Percent of users using 1280x800 screen resolution: **20**
9. Percent of users using 1280x1024 screen resolution: **12**
10. Percent of users using 1440x900 screen resolution: **9**
11. Percent of users using 1680x1050 screen resolution: **6**
12. Percent of users using 800x600 screen resolution: **5**
13. Percent of users in the United States: **29**
14. Percent of users in the United Kingdom: **5**
15. Percent of users in Brazil: **4**
16. Percent of users in India: **2**
17. 1999 projection of 2001 Itanium sales ($billions/year): **35**
18. 2009 Itanium sales ($billions/year): **2**
19. US National Debt as of 09/08/09, 08:12:03am MST: **$11,805,700,226,557.40**
20. Change in the debt since last month's column: **$209,747,044,879.10**

Sources: *1–16: W3Counter Global Web Stats / 17, 18: Sun, IDC / 19: www.brillig.com/debt_clock / 20: Math*

NON-LINUX FOSS



```
hdb1           .pwd.lock      hosts              mixer           services
  a            aliases        hosts.allow        modules.conf    shadow
  bin          aliases.db     hosts.deny         motd            shadow-
  boot         at.deny        hosts.equiv        mtab            shells
  c            bootptab       hosts.lpd          mtools.conf     slip.hosts
  cdrom        csh.cshrc      inet               named.boot      slip.login
  d            csh.login      inetd.conf         networks        snooptab
  dev          devinfo        inittab            nntpserver      soundconf
  e            diphosts       ioctl.save         passwd          syslog.conf
  etc          DIR_COLORS     isapnp.conf        passwd-         termcap
  f            exports        isapnp.conf.works  passwd.OLD      termcap-huge
  home         fastboot       issue              printcap        termcap-small
  lib          fdprm          ld.so.cache        printcap.orig   TextConfig
  lost+found   fstab          ld.so.conf         profile         ttys
  mnt          ftpaccess      lilo.conf          protocols       utmp
  proc         ftpconversions localtime          psdevtab        wtmp
  root         ftpgroups      login.access       resolv.conf     wtmplock
  sbin         ftpusers       login.defs         rmt             X11
  tmp          gateways       magic              rpc             XF86Config
  usr          group          mail.rc            securetty       yp.conf.example
  usr2         group-         mailcap            sendmail.cf
  var          host.conf      makedev.cfg        sendmail.cf.orig
hdc1           HOSTNAME       minicom.users      sendmail.st
hdc6
hdc7
  linux

89 objects   781090 bytes
```

Explore2fs Interface (from www.chrysocome.net/explore2fs)

Explore2fs is a filesystem tool for Windows systems. It allows you to read and explore ext2/ext3 filesystems on your Windows computers. Explore2fs can come in quite handy when you need a file off an external USB drive that was created on a Linux system or if your Linux system has died and you need to move the drive to a Windows system to access it.

Explore2fs is written in Delphi Pascal, and it works on all modern versions of Windows and a few not-so-modern ones like Windows 95 and Windows 98. Explore2fs is a read-only tool. It does not allow you to modify your ext2/ext3 partitions; it allows only viewing and extracting data from them. For write capabilities, you probably should look at one of the available Installable File System (IFS) drivers that has ext2/ext3 support or possibly the "new" version of Explore2fs.

The new version of Explore2fs is still in beta, and it has a new name, Virtual Volumes. It really is only loosely a new version of Explore2fs and more of a rewrite and a rethink of Explore2fs. As mentioned, it's still in beta, so be careful using it.

Virtual Volumes aims to support read and write access to ext2/ext3 volumes as well as some LVM2 volumes. It also provides reading and writing of RAID volumes and VMware disks. In addition, it supports read and write access via SFTP and read access to ReiserFS partitions.

The Explore2fs .exe and source can be downloaded from **www.chrysocome.net/explore2fs**. The project, like many open-source projects, can be helped along by donations and by providing feedback to the author.

—MITCH FRAZIER

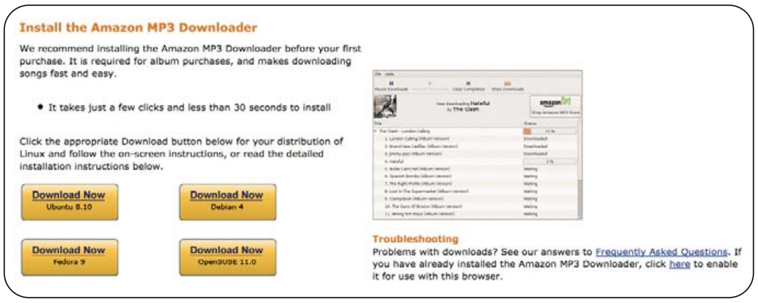
DRM-Free MP3s on Linux, Sweet!

iTunes is touting its DRM-free music as revolutionary and bold. Although I applaud Apple for offering music that isn't crippled, there still is the problem of actually purchasing the songs. Sure Windows and Macintosh users are able to enjoy the "luxury" of buying music from the iTunes Store, but those of us running Linux can only watch from the sidelines.

Thankfully, Amazon has been selling DRM-free MP3 files for quite some time in its digital store. The songs are available for download using a standard Web browser, and because they're MP3 files, they're playable anywhere. Fortunately, we're even able to buy full albums, thanks to Amazon offering its download software for Windows, Macintosh *and* Linux! Whether you prefer to buy one song at a time or collect the whole set and buy full albums, Amazon's native Linux downloader can help you spend money. Now, if only Amazon would remove the DRM from Kindle books.

—SHAWN POWERS

Way to go Amazon! A native Linux client for downloading albums.



Install the Amazon MP3 Downloader

We recommend installing the Amazon MP3 Downloader before your first purchase. It is required for album purchases, and makes downloading songs fast and easy.

- It takes just a few clicks and less than 30 seconds to install

Click the appropriate Download button below for your distribution of Linux and follow the on-screen instructions, or read the detailed installation instructions below.

Troubleshooting

Problems with downloads? See our answers to Frequently Asked Questions. If you have already installed the Amazon MP3 Downloader, click here to enable it for use with this browser.

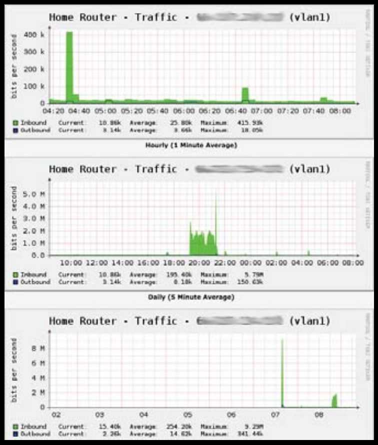
When Cacti Are Too Prickly

I'm a clock watcher. I'm a tachometer hound. I'm a speedometer freak. And, as cool as that might seem, more than anything, I love graphs. One of the problems with setting up good SNMP graphs is that not only do they require a server on which to run, but also those servers often are difficult to configure. Programs like Cacti offer incredible features, but for home use, they're usually not worth the effort to configure.

That's where odmon.com comes in. **Odmon** offers a free service that queries your SNMP devices and hosts Cacti graphs for your local devices. Setup couldn't be easier, and odmon even offers "wizards" to help configure your home router. Part of the process is opening your firewall to its servers, but because you need to provide only read access, and you can limit access to its specific IP address, the security is tolerable—at least for me on my home system.

To set up your free odmon account, go to **www.odmon.com**. You can be a bandwidth watcher like me in no time!

—SHAWN POWERS



Odmon offers a wide variety of graphs and the ability to publish them publicly or privately.

Real Work on Virtual Terminals

Screen is very good at multiplexing a terminal and running several login sessions over a single terminal connection. If you are SSHing into a server, screen is your best choice, allowing you to run multiple applications easily over that one connection. But, what if you are sitting down at your desktop or laptop? By using the tools mentioned here, you don't need to limit yourself to using a single virtual terminal anymore.

By default, most distributions are set up to activate six virtual terminals. This is done through the /etc/default/console-setup file. The number of virtual terminals initially activated can be set by editing the following line:

```
ACTIVE_CONSOLES="/dev/tty[1-6]"
```

Change the value to the number of virtual terminals you'd like to have on bootup. Your system may have the /etc/inittab file instead, which contains a series of commands to initialize those six terminals.

You can switch between the different virtual terminals with the Alt-F1 to Alt-F6 keys. Or, you simply can type `chvt N` on the command line to change to the virtual console number N.

You aren't limited to using only the virtual terminals created on bootup. You can use commands to create, destroy and interact with virtual terminals on the fly. First, you may need to know which terminal you are on. The `fgconsole` command gives you that information by printing out the number of the current console, and typing `fgconsole --next-available` will tell you the next available console number.

To open a new console, simply type the command `open bash`.

You can replace bash with any other command you want to execute on the newly opened virtual terminal. If you don't give a command to open, a bash shell is started. By default, your program will run, and the output will be written on the newly created terminal without actually switching there. If you want to switch to the new terminal automatically, simply add the command-line option -s to the open command.

You also may experience the error "Unable to open /dev/ttyN: Permission denied", depending on your machine's level of security. If you get that error, you simply need to run the command with sudo appended to the front, so you have root privileges. But, then the newly created bash shell will be run as root. If you want to run the given command as yourself, use the command-line option -u. This tells open to figure out who the current user is and to run the command given to open as that user.

You can find out which users are logged in to various virtual terminals by using the command `who`. This tells the user names, their virtual terminal and when they logged in.

You can send a message to other users with the command `write`. If they are logged in to more than one virtual terminal, write defaults to sending the message to the virtual terminal with the lowest idle time. Or, you can send the message to a particular virtual terminal by executing `write username ttyN`.

Everything you write will be mirrored on the remote virtual terminal. When you are done, press Ctrl-d. If you don't want to receive messages from other users, you can turn off reception with `mesg n`. Doing so blocks all users (except the superuser) from being able to send you messages. When you are ready to receive messages again, simply type `mesg y`.

If you want to send a message to everyone at once, use the command `wall`.

Now, how do you work with your consoles? You can clear output two ways. The first, with the command `clear`, simply clears away the currently displayed output. But, you still can scroll backward and see the previously displayed output. If you also want to clear that history, use the command `clear_console`.

If you want to record what you are doing at the terminal, use the command `script`. All of the output printed on your terminal is copied into a file. The default filename is typescript, but you can change it by adding a filename to the end of the command. Once you execute the script command, everything else that is displayed on your terminal also will be copied out to this file. When you're done, simply press Ctrl-d. You now have a full transcript of what just appeared on your terminal.

When you are done with your virtual terminals, shut them down by using the command `deallocvt`. If you simply execute deallocvt, it will deallocate all of the unused virtual terminals. If you want to deallocate a particular virtual terminal, execute `deallocate N`, where N is the virtual terminal to deallocate. Now you are ready to play with the virtual terminals on your desktop.

—JOEY BERNARD

They Said It

People don't change their minds. They die, and are replaced by people with different opinions.
—Arturo Albergati

No man who ever held the office of President would congratulate a friend on obtaining it.
—John Adams

It is as ridiculous for a nation to say to its citizens, "You must consume less because we are short of money", as it would be for an airline to say, "Our planes are flying, but we cannot take you because we are short of tickets."
—Sheldon Emry

Success is getting what you want, happiness is wanting what you get.
—Charles Kettering

Community Events at LinuxJournal.com

The *Linux Journal* staff had a pretty great summer this year. Shawn Powers and I enjoyed both OSCON and the San Jose weather tremendously. Shawn and Jill Franklin had a great time at the inaugural LinuxCon in Portland, and you may have been lucky enough to catch Shawn's keynote at Ohio LinuxFest. Meanwhile, I cavorted around Paris with the coolest Web people in the world at DrupalCon Paris. We love going to events and meeting people, learning something new and maybe having a beer or five. Although we don't make it to all of them, we have a list of the best conferences and events in the open-source world listed at **www.linuxjournal.com/events**. If you want the heads-up on what's going on and where you should be, make sure you check that page! We hope to see you there, so be sure to say hi. If you'd like to know where we'll be in the future, follow us on Twitter or Identica at **twitter.com/linuxjournal** or **identi.ca/linuxjournal**, and get the inside scoop.

—KATHERINE DRUCKMAN

Awesome Product, Painful Name

The eee-Book concept photo looks a bit more like a sideways laptop than a book, but we love the dual touchscreens in full color.

ASUS recently announced what looks to be the nicest, most affordable eBook reader on the market today. With dual color screens, wireless network access and even a Webcam, the ASUS eBook reader looks particularly sexy with its sub-$200 price tag. I just hope it's not another product in a line of technology with a few too many Es in the name. The name eee-Book makes me throw up a little. If the price remains that low and the features that high, however, I might be willing to forgive a name that has become cliché.

—SHAWN POWERS

Gord spent too many years learning clusters

Learn what Gord knows

Free Clusters for Dummies® book and HPC white paper

www.platform.com/gord

Platform Computing

2009 Book Roundup

A look at the publications Reuven recommends on Ruby, Rails, JavaScript, Git, Web development and more.

REUVEN M. LERNER

As I write these words, the global economy has been in a recession for more than a year, bringing with it untold financial ruin for a large number of businesses, organizations and individuals. Book and magazine publishers have not emerged unscathed, with many downsizing or otherwise trying to figure out how to profit (or survive) in the Internet era.

But despite the current situation, publishers continue to produce a large number of books, many of which have to do with Web- and Internet-related technologies. If you are a Web developer, you are fortunate to live in an era when high-quality, open-source software is available, Web development frameworks have become popular and easy to use, and there are dozens of blogs on any given open-source technology. Quaint as it might seem, printed books are tremendously useful resources that you can and should try to use to your advantage. Blogs can be excellent, but I still enjoy reading a well-written book that walks me through numerous examples of a new technology or concept.

But as Rails has grown in popularity, the needs of the sites that use it also have grown, either in functionality or scalability.

This month, I'm taking a break from my normal coding examples in order to share some of the books I have looked through and enjoyed during the past year. Most of these books are actually new, but some of them might be just new to me or of new importance to me. There is definitely some bias in favor of the technologies I typically use—Linux, Ruby, PostgreSQL and Git—but I try to remain up to date on a variety of technologies and subjects, and the list of books reflects those interests as well.

Ruby and Rails

Anyone who reads this column knows I am one of the many Web developers who loves to work with Ruby on Rails. Rails has been my preferred development framework for several years, and I continue to be impressed by the number of conveniences that it includes. Rails took off because it was easy to get started with it, as David Heinemeier Hansson demonstrated in his initial "blog" screencasts several years ago. But as Rails has grown in popularity, the needs of the sites that use it also have grown, either in functionality or scalability. Dealing with those issues—preferably before they cause trouble for your site—has become an important topic for Rails developers.

The best book I've seen on the subject is *Enterprise Rails* by Dan Chak (O'Reilly, ISBN 978-0-596-51520-1). One of the reasons I like this book so much is that it focuses on aspects of Web development that most Rails books either ignore or relegate to the sidelines. For example, the first chapter walks you through the creation of a Rails plugin, which Chak argues is a good way to organize your code for easier maintenance. Whether this actually is a good idea can be the subject of discussion and debate, but it is a rare Rails book that discusses the creation of plugins at all, to say nothing of addressing them as organizational tools. Chapter after chapter in this book is similarly interesting and includes informative discussions of database normal forms, SOA, caching, inheritance and the use of constraints and triggers within the database to enforce data integrity.

A book that covers more conventional ground, but one that is certainly quite useful, is *Advanced Rails* by Brad Ediger (O'Reilly, ISBN 978-0-596-51032-9). *Advanced Rails* covers many of the topics a developer needs to consider when deploying an application and when considering security and scalability issues. The book covers a great many topics, and my only complaint is that it tries to cover so much, it loses some of the depth I might have wanted. At the same time, the book is full of references to gems, plugins and Web sites that cover the information in greater depth (and with more working code) than any book could reasonably be expected to include.

If you're looking for information about the Ruby language, rather than the Rails framework, two new books have come out in the past year, both of which address not only the current 1.8.x series of Ruby, but also the 1.9.x series. These books have different purposes and styles, and they complement each other in many ways. *The Ruby Programming Language* co-authored by Java/JavaScript book-veteran David Flanagan and Ruby creator Yukihiro "Matz" Matsumoto (O'Reilly, ISBN 978-0-596-

51617-8) is an attempt to document, specifically and carefully, the language's behavior. If you are an experienced Ruby programmer, you probably will want this book on hand in order to explain how the language works.

The Well-Grounded Rubyist by David Black (Manning, ISBN 978-1-933988-65-8) is a much friendlier book, and it is a tutorial of sorts—not just on the Ruby language, but also in the Ruby way of thinking. A number of Ruby's constructs can be confusing for many programmers, and Black's book steps through them with numerous, well-documented examples. Black also provides a number of tips and explanations about things that aren't always obvious, such as the difference between singleton method definitions styles, built-in callbacks and the various forms of eval.

I also thoroughly enjoyed *Metaprogramming Ruby* by Paolo Perrotta (Pragmatic Programmers, ISBN 978-1-934356-47-0). It is easy to get started programming with Ruby, but the real power (as with Lisp) is not just with existing Ruby constructs, but the fact that you can modify the language to suit your needs. Metaprogramming, as this technique is called, lets you modify objects and classes in a variety of ways to turn your application into a language for solving your specific problems. Metaprogramming is a bit hard to grasp by its very nature, and it isn't necessarily obvious how to go about using it, or why it might be necessary. Perrotta's book offers a great deal of well-written detail on both fronts, showing you how to use metaprogramming techniques and suggesting when they might be appropriate or useful.

Finally, I should mention Mike Gunderloy's self-published on-line PDF book *Rails Rescue Handbook*, which you can get from **www.railsrescuebook.com**. Gunderloy is an active Rails developer, author and community member, and he wrote a book that describes what you should do when you are asked to work on a Rails project that is not working. This book is full of practical advice on how to attack a problematic Web site—from examining the existing codes, to looking for database indexing issues, to the use of metric_fu, to external monitoring with tools from New Relic or FiveRuns. A list of what functionality was deprecated in each version of Rails (going back to 1.0) is handy for those of us who often work on multiple projects simultaneously and might not remember what changed between Rails 2.1 and 2.2, for example. I didn't find any hidden tricks or clever hacks in this book, but that's just fine. The back-to-basics approach is thorough, well written and describes how every Rails project can and should look over time, even if it didn't start off following best practices.

Git Books

Software developers have been using version-control systems for some time. But Git, a distributed version-control system developed by Linus Torvalds, has taken much of the open-source world by storm. For me, the killer feature in Git is its ridiculously simple (and fast) branching and merging. I fall into the category of CVS and Subversion users who have worked with those tools for years, dreading any branching or merging operation that I would have to perform, because it was so painful and time consuming. Git has totally changed that for me, altering the way I develop software.

Numerous Web sites exist for Git users, such as "Git ready" (**www.gitready.com**) and the Git community book (**book.git-scm.com**), which offer useful information. But for a complete introduction to Git, you might want to consider one of three books on the subject. The first book that came out, *Pragmatic Version Control Using Git* by Travis Swicegood (Pragmatic Programmers, ISBN 978-1-93435-615-9), is a good introduction to Git and covers the basics nicely.

However, I felt that this book was lacking some depth and was happy to read *Version Control with Git* by Jon Loeliger (O'Reilly, ISBN 978-0-596-52012-0) and *Pro Git* by Scott Chacon (Apress, ISBN 978-1-4302-1833-3). Chacon is well known in the Git community as one of the founders of GitHub and as a screencaster, author and speaker about Git. Chacon convinced Apress to put the book on-line for free (**progit.org/book**), so you can take a look for yourself. I have found that the Loeliger and Chacon books complement each other, and I've been reading them in parallel, learning from both. You can't go wrong with either one of them.

Leading NTFS and exFAT drivers

Every device with mass storage, at home and at work, needs an interoperable file system. Your set-top-box runs Linux but needs to write the media in a Windows and Mac compatible format.

Tuxera NTFS for Embedded Systems
Performance increase 10-100x compared to our open source NTFS-3G. Proven reliability and data integrity. Low CPU usage, small memory footprint. Available for any system.

Tuxera exFAT for Embedded Systems
exFAT is part of SDXC and Memory Stick standards. New product available now for Linux.

+358 50 5980498
sales@tuxera.com
www.tuxera.com

TUXERA

Web Development and Administration

No matter what technologies your Web site uses, certain issues will crop up. For example, you will have to map out a database and server architecture, ensure that your server's performance is being monitored, and set your URLs and content to reflect best SEO (search engine optimization) practices. It is unlikely that one person on a Web site will need to tackle such a wide variety of problems alone, but if you are a freelance developer, knowing about different tools can be quite helpful and makes you even more valuable to your clients.

Website Optimization by Andrew B. King (O'Reilly, ISBN 978-0-596-51508-9) is a good introduction to the subject of optimizing your site in a number of ways, both for SEO and for speed. Generally, I've been quite skeptical of SEO in the past, thinking (somewhat naïvely) that with well-written content, Google and other search engines will find you and mark you as relevant. It turns out that well-written content is necessary but insufficient. This book shows what you can do to improve in that arena. To be honest, I have read the SEO portions of this book much more thoroughly than the performance-related optimization portions, some of which I have seen elsewhere.

JavaScript continues to occupy center stage in the Web development world. If you aren't yet using a JavaScript framework, by all means, you should start!

A well-known business mantra says, "If you can't measure it, you can't improve it." For this reason, getting a handle on how your Web site performs is a crucial task if you are to understand where and how you can improve it. The book *Complete Web Monitoring* by Alistair Croll and Sean Power (O'Reilly, ISBN 978-0-596-15513-1) is the most comprehensive list of Web monitoring tools and techniques I have seen to date, looking at every type of monitoring I can think of and then some. It describes how to monitor your site's network connectivity, performance, conversion rates and usability, recommending a mixture of built-in, open-source and commercial tools. It even describes ways in which you can use on-line communities and social networks to monitor reactions to your site, so that the analysis you get is not just a bunch of statistics.

JavaScript continues to occupy center stage in the Web development world. If you aren't yet using a JavaScript framework, by all means, you should start! Frameworks (such as Prototype/Scriptaculous, jQuery, YUI and Dojo) allow you to ignore most of the differences between browsers and provide a huge amount of support for the type of development that you likely want to do. I typically prefer to use jQuery and enjoyed the introductory *Learning jQuery* by Jonathan Chaffer and Karl Swedberg (Packt Publishing, ISBN 978-1-847192-50-9) as a good introduction to that framework.

Many serious developers now are working on JavaScript in various ways, and *JSMag* is a new for-pay, PDF publication that aims to give such developers serious articles (**www.jsmag.com**). *JSMag* reminds me of other language-specific magazines I have seen over the years, with very high-quality content that addresses topics developers need to deal with. It tries to be framework-neutral, meaning you are as likely to read an article about YUI or jQuery as Ext or Dojo, so if you are interested only in jQuery, you might be slightly disappointed. That said, a majority of the articles are about using the JavaScript language, rather than any one particular framework, which means that no matter what you're using, you will probably get something out of *JSMag*.

Finally, two well-known figures in the Ruby community, Amy Hoy and Thomas Fuchs (the latter of whom is the author of the Scriptaculous visualization framework that works with Prototype) have published an on-line PDF book called *JavaScript Rocks!*, available for purchase and download from **jsrocks.com**. If you are familiar with Hoy's writing, you won't be surprised that this book is an easy read, with a terrific sense of humor and high-quality technical content, addressing everything from reducing JavaScript code size to improving the perceived (not actual!) performance of your application's user interface. The book comes with a copy of the terrific "DOM Monster" tool for optimizing the number and type of DOM elements on a page and for an instant analysis of your page.

Conclusion

This article presents just a small sample of the books I've enjoyed this past year. Authors and publishers continue to release new, good books that are not only useful, but also interesting to read. It's rare for me to read a technical book and not be able to use at least something from it within a few days of reading it. The books I've mentioned here stand out from the crowd because they have provided me with a particularly large helping of food for thought. I look forward to this-coming year's helping of books and to the interesting things that I will learn from them as well.■

Reuven M. Lerner, a longtime Web/database developer and consultant, is a PhD candidate in learning sciences at Northwestern University, studying on-line learning communities. He recently returned (with his wife and three children) to their home in Modi'in, Israel, after four years in the Chicago area.

Lane Becker of Get Satisfaction speaks at Do It With Drupal 2008.

DO IT WITH DRUPAL

A 3 DAY SEMINAR
NEW ORLEANS, LA
DECEMBER 9 - 11, 2009

STILL GETTING YOUR HEAD AROUND DRUPAL?

Learn to harness the power of this flexible open-source social content management system and web application framework.

Do It With Drupal is in its second year and the 2009 event promises to be even better than last year.

- Learn from the world's top Drupal experts
- Examine and dissect successful Drupal sites
- Discover new site-building strategies
- Hear from community-building experts
- Connect with other Drupal professionals

"Your favourite community personalities, the developers of the most power-packed modules and experienced Drupal solution providers join together to teach you the tools to build amazing Drupal powered websites. I loved it! I can't wait to put what I learned at DIWD into action."
Matthew Nuzum
Ubuntu.com Webmaster

"Broad enough for beginners, focused enough for experts. A required event for the Drupal community."
Daniel Chvatik
CTO, Adulmec

REGISTER NOW
DoItWithDrupal.com

Lullabot
powered

Registration price includes:
- Admission to all sessions and keynote presentations
- Social and networking events with Drupal's leaders and top talent
- Breakfast and lunch each day of the event
- Access to entire 2009 Do It With Drupal video archive

Early registration and hotel room discounts available.

DAVE TAYLOR

Calculating the Distance between Two Latitude/Longitude Points

Finding your way at the command line.

Last month, I closed this column with a script that can return latitude/longitude values for two addresses, with the intent ultimately being to have the script calculate the distance between those two points. As an example:

```
$ farapart.sh "union station, denver co" \
      "union station, chicago il"
Calculating lat/long for union station, denver co
= 39.75288, -105.000473
And calculating lat/long for union station, chicago il
= 41.878658, -87.640404
```

The formula to calculate distance actually is pretty complicated. Here's a JavaScript implementation of the math I showed last month:

```
var R    = 6371;       // kilometers
var dLat = (lat2-lat1);
var dLon = (lon2-lon1);
var a    = Math.sin(dLat/2) * Math.sin(dLat/2) +
           Math.cos(lat1.toRad()) * Math.cos(lat2.toRad()) *
           Math.sin(dLon/2) * Math.sin(dLon/2);
var c    = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));
var d    = R * c;
```

This is going to be a wee bit tricky to convert into a shell script, needless to say, but because we have to use a more sophisticated math tool than the built-in capabilities of Bash anyway, this also means we have a number of options to work with, including Perl, awk and bc. For that matter, we also can just write a quick C program that solves this equation given four variables, but really, why make it easy when I can make it complex? If I wanted easy, I would whip out some Perl, right? Last month, I promised some bc, so let's see if we can make that rusty old app do the heavy lifting.

Degrees to Radians

The first step mathematically is to convert the lat/lon values we get from the mapping system from degrees to radians. This turns out to be straightforward:

Radians = degrees * (pi / 180)

Pi, of course, is 3.14159265358979323 84.

Given values like:

```
41.878658, -87.640404
```

The radians equivalent of those is then:

```
0.7309204767, -1.529613605
```

To warm up with bc, here's a simple command-line way to calculate one of these values:

```
echo "scale=8; -87.640404 * ( 3.14159265 / 180)" | bc
```

That's all well and good, but it turns out that the different equations I explored for calculating the distance between two points requires the atan2() function, which isn't part of bc.

Rather than beat my head against the old-school wall until the bits are bloodied, I'm going to throw in the towel and admit that this might just be a bit too complex a mathematical problem for a shell script and bc.

Dave Cries Uncle!

Having spent way more hours than I want to admit trying to get this to work properly in bc, I'm going to "cry uncle" and switch temporarily into a different programming language. I'm going to jump into C for a few lines and whip out a simple program that, given two lat/lon pairs in degrees, calculates the distance between them in miles (Listing 1).

Does it work? Let's find out:

```
$ distance 39.75288 -105.000473 41.878658 -87.640404
917.984
```

That seems reasonable. The great circle distance between those two points is 917 miles. Google Maps, of course, shows about 10% greater distance, but perhaps that's because there is no direct-as-the-crow-flies route via roads?

Of course, there also are errors with this formula too, because Earth isn't a perfect sphere but rather an oblate spheroid that has a different diameter depending on where you're measuring. But for our purposes, let's just gloss over that problem. You can Google it to

Lullabot-Powered
The most super powered sites in the world are created in Drupal, by you and Lullabot.
Suzi Arnold
Director of New Media
Sony Music
Lullabot™
New Lullabot Learning Series training DVDs at Lullabot.com

Listing 1. C Distance Program

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

#define EARTH_RADIUS        (6371.0072 * 0.6214)
#define TORADS(degrees)     (degrees * (M_PI / 180))

main(int argc, char **argv)
{
   double lat1, long1, lat2, long2;
   double dLat, dLong, a, c, d;

   lat1  = TORADS(atof(argv[1]));
   long1 = TORADS(atof(argv[2]));
   lat2  = TORADS(atof(argv[3]));
   long2 = TORADS(atof(argv[4]));

   dLat  = lat2 - lat1;
   dLong = long2 - long1;

   a = sin(dLat/2) * sin(dLat/2) +
       cos(lat1) * cos(lat2) * sin(dLong/2) * sin(dLong/2);
   c = 2 * atan2(sqrt(a), sqrt(1-a));

   printf("%g\n", EARTH_RADIUS * c);
}
```

learn about things like the Vincenty formula, but that's beyond the scope of this ridiculous sidetrack.

Now we have all the pieces we need: location to lat/lon and distance between two lat/lon points. Let's put it all together.

Grafting It All Together

To get everything to work well, I actually hacked and slashed at the original script to make it a bit more succinct and, of course, invoke the C "distance" program as shown in Listing 1. [Listing 1 also is available on our FTP site at **ftp.linuxjournal.com/pub/lj/listings/issue188/10606.tgz**.] Ready? It's surprisingly short:

```
#!/bin/sh
converter="http://api.maps.yahoo.com/ajax/
➥geocode?appid=onestep&qt=1&id=m&qs="

tmpfile="/tmp/bc.script.$$"

# Get lat/long for point 1
addr="$(echo $1 | sed 's/ /+/g')"
values="$(curl -s $converter$addr | \
  cut -d\" -f13,15 | \
  sed 's/[^0-9\.\,\-]//g;s/,$//')"

lat1=$(echo $values | cut -d, -f1)
long1=$(echo $values | cut -d, -f2)

# Get lat/long for point 2
addr="$(echo $2 | sed 's/ /+/g')"
values="$(curl -s $converter$addr | \
  cut -d\" -f13,15 | \
  sed 's/[^0-9\.\,\-]//g;s/,$//')"

lat2=$(echo $values | cut -d, -f1)
long2=$(echo $values | cut -d, -f2)

# Now we have the lat/long for both points, let's
# figure out the distance between them...
dist=$(./distance $lat1 $long1 $lat2 $long2)
echo "$1 to $2 is $dist miles"
exit 0
```

The script would be even shorter if we tweaked the C program to accept x,y location pairs, but I'll leave that one to you. Instead, let's do a few tests:

```
$ farapart.sh \
      "union station, denver, co" \
      "union station, chicago, il"
union station, denver, co to
    union station, chicago, il is 917.984 miles
```

Now, how about something a bit more ambiguous:

```
$ farapart.sh "long beach, ca" "boston, ma"
long beach, ca to boston, ma is 2597.53 miles
```

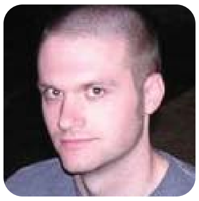
Well, darn it, that seems way too short. Let's see what Yahoo Maps reports as the distance between those two cities. Sure enough, it reports that the trip should be 3,015 miles, not 2,597 miles.

Debugging the Math Formula

Somewhere there's an error that's giving us poor results. My guess is there's some sort of significant rounding error going on in the C program (because we can verify experimentally that the lat/lon information we're getting is valid, simply by plugging it in to a mapping app and seeing where it places us).

I'm all tapped out on this example, however. It turned out to be far more tricky than I anticipated, and I leave it as an exercise to you, dear reader, to see if you can figure out what's broken in the C program and report your fix to us. We'll publish the best of them next month! Meanwhile, next column, I'll get back to something that's more about the shell and less about mathematics. I mean, heck, I didn't like math when I was working on my computer science degree, so why am I playing with it now?■

Dave Taylor has been involved with UNIX since he first logged in to the on-line network in 1980. That means that, yes, he's coming up to the 30-year mark now. You can find him just about everywhere on-line, but start here: www.DaveTaylorOnline.com. In addition to all his other projects, Dave is now a film critic. You can read his reviews at www.DaveOnFilm.com.



Message for You Sir

Why check for new e-mail every minute when a script can do it for you? Learn how to trigger desktop notifications in your own scripts.

KYLE RANKIN

It's easy to forget dæmons are there unless they demand your attention. A few years ago, I was walking through the expo floor at OSCON, when I noticed someone in a full BSD dæmon costume getting his picture taken with a few fans. When I saw them trying to figure out how to arrange everyone for the picture I couldn't help but yell, "No! The dæmon is always in the background!"

In case you didn't get the joke, *dæmon* is a name UNIX people give to processes that run behind the scenes (in the background). Dæmons perform all sorts of useful functions from executing scripts at a certain time (atd and crond) to listening for network connections and spawning the appropriate process to serve the request (inetd). In fact, the d at the end of those scripts stands for dæmon, and you might notice that a number of processes on your system right now end in d.

The whole point of a dæmon is to perform tasks without your intervention or knowledge, but sometimes, it's handy for a dæmon to alert you when a certain condition occurs. On a server, this usually means the dæmon will send an e-mail alert to the administrator, but what about on a desktop? What if you want a dæmon to alert you when you have new e-mail? In that case, it makes more sense for some sort of notice to pop up on your desktop. In this column, I discuss three different methods I use so that dæmons can get my attention on my desktop.

A Case of OSD

I think the first time I noticed OSD (On-Screen Display) notifications on Linux was with a volume control program. I increased the volume on my computer, and right in the middle of the screen was a volume meter floating above all my other windows, just like on a TV. I instantly was intrigued and had to figure out how they did it. These days, there are a number of different OSD libraries and programs, but my favorite is still osd_cat.

The osd_cat program is a command-line program that displays text sent to it in a pipe. The fact that it accepts piped input makes it ideal for dæmon notification, because it's easy to add to any shell script. This command is part of the xosd-bin package on Debian-based systems, or xosd on Red Hat, and has been around for a number of years.

The simplest way to test osd_cat is to pipe some text to it:

```
$ echo "Hello World" | osd_cat
```

If you look at the top left-hand side of your screen, you should see your message appear in a small red font for a few seconds and then disappear. Of course, if you didn't know to look there, you might assume the program is broken because the message is so small. Plus, everyone knows green is the ideal foreground color, so let's spruce up that notification a bit and put it front and center:

```
$ echo "Hello World" | osd_cat --align=center --pos=bottom
➥--color=green --font=lucidasanstypewriter-bold-24
```

Ahh, that's more like it, a notification right in the middle of the screen. As you can see, osd_cat accepts a number of options that can control how and where it displays your message. The man page covers all the options in detail, but I highlight the options I used here. The --align argument controls the text alignment much like a word processor and can be set to left (default), center or right. The --pos option controls the Y orientation on the screen and can be set to top (default), middle or bottom. The --color option is self-explanatory, as is --font. If you do want a different font but aren't sure what value to use, just run `xlsfonts` to see a full list.

In addition to the options I listed, osd_cat has many other options, such as --indent and --offset, that allow you to fine-tune where it displays your text, so you can position it virtually anywhere on the screen. You also can tweak the time the message appears on the screen with the --delay option, and if you plan to pipe multiple lines of text to the screen, be sure to look into the --lines, --age and --wait options, so you can control how osd_cat will scroll multiple lines. There are even --barmode and --percentage options that let you draw a slider bar, much like the OSD volume control I saw.

All of those options are nice, but honestly, I find myself sticking to basic text notifications with osd_cat, and although I have migrated many of my scripts to libnotify, I still like to use osd_cat for audio/video notification, such as when I use a script I wrote to turn on the VGA output on my laptop for presentations. I mentioned this script in my original Lightning Hacks column, but in case you didn't see it, here it is again:

```
#!/bin/sh

if xrandr | grep -q 'VGA connected'; then
    echo "LVDS + VGA" | osd_cat --shadow=2 --align=center
 ➥--pos=bottom --color=green --delay=2
 ➥--font=lucidasanstypewriter-bold-24 --offset 40 &
# choose my laptop screen's resolution by default;
# if that fails, try the auto-detected mode
    xrandr --output VGA --mode 1280x768@60 || xrandr
 ➥--output VGA --auto
else
    echo "LVDS only" | osd_cat --shadow=2 --align=center
 ➥--pos=bottom --color=green --delay=2
 ➥--font=lucidasanstypewriter-bold-24 --offset 40 &
    xrandr --output VGA --off &
fi
```

I find it's nice to add visual notifications like this whenever I trigger a script in the background with a keybinding and it's not immediately obvious the script ran, such as in this case, when I'd like to know whether I'm in presentation or regular mode, and it might take the projector a few seconds to respond.

Consider Yourself Notified

I used to use OSD notifications for all sorts of scripts on my system, including one that notified me when I got new e-mail. It worked fine, but sometimes I'd rather get a notification that's a little easier to ignore. Although I suppose I could move my OSD alert to a corner of the screen, then it would be almost too easy to miss. I wanted something that would get my attention but also would not get in the way. Currently, I use GNOME as my desktop environment, and I realized that its desktop notifications were ideal. They caught the corner of my eye, but they didn't jump in front of everything else I'm doing.

The library GNOME uses for notifications is the appropriately named libnotify, and it turns out, it was trivial to migrate my notifications from osd_cat to libnotify using the notify-send command. Because I already was using GNOME, the program already was installed. If it's not in your case, look for a package named libnotify-bin or search your package manager for notify-send.

The syntax for notify-send is substantially simpler than osd_cat, as the message's location and font already are handled for you. Here's a simple example:

Expert included.

As a Sales Engineer for Silicon Mechanics, Scott is here to consult with you, understand your processing and storage needs, and then configure the systems that will work for you. Energy efficiency, compute density, reliability, and serviceability are just some of the things he makes a priority. That's why Scott is excited about the Bladeform 8840 Blade for the Bladeform 8100 Series Blade Server Platform. Each Bladeform 8840 Blade supports four of the latest Six-Core AMD Opteron™ 8000 Series processors. With 24 cores per blade and 10 blades per enclosure, you can pack the power of 240 cores in just 7U of space.

At the same time, Scott is impressed that the Bladeform 8100 provides 90%+ high-efficiency redundant power supplies for operating cost reduction and earth-friendly computing. The Bladeform 8100 Platform with the Bladeform 8840 Blade is a perfect choice for mission-critical enterprise applications as well as scale-out and high performance computing environments.

When you partner with Silicon Mechanics, you get more than a high-efficiency AMD solution — you get an expert like Scott.

SILICON MECHANICS

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

See the Silicon Mechanics Bladeform 8100 and the 8840 Blade at www.siliconmechanics.com/8840

Silicon Mechanics and the Silicon Mechanics logo are registered trademarks of Silicon Mechanics, Inc. AMD, the AMD Arrow logo, AMD Opteron, and combinations thereof, are trademarks of Advanced Micro Devices, Inc.

```
$ notify-send "Message for you Sir" "Hello World"
```

This will pop up a basic message in a notification window on my desktop. The first set of quotes specifies the title of the message, and the next set of quotes defines the body of the message.

Of course, when I use notify-send to alert me of new e-mail, I use something a bit fancier. If you'd like to set up e-mail notification for yourself, here's a more simplified version of my personal script to get you started. First, set up fetchmail so that it can connect to your IMAP server. Just as a warning, don't ever run fetchmail without the -c option, unless you do want to download all your mail to your local machine. Once fetchmail is configured, you can test that it works with `fetchmail -c`:

```
$ fetchmail -c
991 messages (990 seen) for kyle at mail.example.net
 (folder INBOX).
530 messages (530 seen) for kyle at mail.example.net
 (folder INBOX.nblug).
284 messages (284 seen) for kyle at mail.example.net
 (folder INBOX.linuxjournal).
```

As you can see, there's a new message that I haven't seen in my INBOX folder. All you need to do now is write a script that will execute `fetchmail -c`, parse the output, and tally the total and seen messages. If the totals differ, you have new mail and can execute notify-send with the appropriate message. Here's a quick Perl script that goes a step further and keeps track of each folder with new messages as well, so it can list them and their tally:

```
#!/usr/bin/perl

open FETCHMAIL, "/usr/bin/fetchmail -t 10 -c 2>/dev/null
 ➥|" or die "Can't run fetchmail: $!\n";

while(<FETCHMAIL>){
   if(/^(\d+) messages \((\d+) seen.*?folder (.*?)\)/){
   # keep a running total of all messages and seen messages
      $messages+=$1; $seen+=$2;
      $folder=$3;
      $folder =~ s/INBOX\.//; # strip the INBOX.
                              # from the folder names
   }
# If there are more messages than seen messages,
# store the difference
   if($1 > $2){
      $folders{$folder} = $1 - $2;
   }
}
close FETCHMAIL;

$total = $messages - $seen;

if($total > 0){
   foreach $folder (sort { $folders{$a}<=>$folders{$b}
   ➥} keys %folders){
      push @list, "$folder:$folders{$folder}";
   }
   $output = join " ", @list;
   system ("notify-send -u low -i /usr/share/pixmaps/mutt.xpm
   ➥-t 5000 'New Mail' '$output'");
}
```

Notice I added a few extra options to my notify-send in this example. First, I used the -u option so I could set the urgency of the message to either low, normal or critical. The -i option lets me specify an icon to add to the image, so I picked my system's mutt icon, because that's the program I'll use to read the mail. Next, I used the -t option so I could set the timeout for the message in milliseconds. Finally, I added the title and body of my message.

If you set this up yourself, all you have to do now is save the script and add it to your user's crontab so it will run however often you want to check for new mail. I also recommend adding some sort of throttling to the script, so it will notify you of a current batch of new mail only a few times. That way if you can't get to your e-mail immediately, the notifications won't become annoying.

Don't Forget to Blink

Desktop notifications are great, but what happens if I'm not looking at the screen when a notification appears? Sure, if the script runs every minute, I eventually will see it, but I set up throttling for those sorts of notifications. I came up with a notification that's even less intrusive than libnotify, yet it will alert me of new mail even if my screen idles out and goes blank: my keyboard LEDs.

Now there certainly is nothing new about using keyboard LEDs for notifications—after all, their intended purpose is to notify you about the state of your caps lock, num lock and scroll lock keys. But, how often do you use your scroll lock or even your caps lock these days? I mean, a number of my friends even went full-UNIX nerd on me and remapped caps lock back to the Ctrl key. Your keyboard LEDs are three notification areas begging to be used, and Linux has plenty of utilities that can use them.

I've tried a few different tools that let me control the keyboard LEDs from a script, but I've settled on blinkd as my favorite. This program runs as a dæmon (see the d at the end?) when the system starts up, and what I like about it is that it not only lets you control all three keyboard LEDs, but you also can set a number of times for it to blink the LED—perfect for keeping track of new e-mail messages.

To install blinkd, look for the package of the same name with your package manager. Once it's installed, if your package manager didn't automatically start it, run `/etc/init.d/blinkd start`. After the dæmon is running, you can control the LEDs via the blink command. The syntax is pretty simple. For instance, if I want to blink the scroll lock key a single time, I would type:

```
$ blink -s -r 1
```

The -s argument tells it to activate the scroll lock LED, and -r tells it how many times to blink before it pauses. I also could have used -c or -n instead of -s to blink the caps lock or num lock LEDs, respectively. You also can set the number of blinks to 0 to turn off blinking for a specific LED, or type `blink -r 0` to turn off blinking for all of the LEDs.

Because the script I wrote above already has the total number of new messages, it's trivial to have my scroll lock key blink that number of times. Here's the modified section of my script:

```
if($total > 0){
   foreach $folder (sort { $folders{$a}<=>$folders{$b}
   ➥} keys %folders){
      push @list, "$folder:$folders{$folder}";
   }
   $output = join " ", @list;
   system ("notify-send -u low -i /usr/share/pixmaps/mutt.xpm
   ➥-t 5000 'New Mail' '$output'");
   system ("blink -s -r $total");
}
else {
   system ("blink -r 0");
}
```

The great thing about using a keyboard LED for notifications is that you end up noticing it out of the corner of your eye, especially if you left your computer for a minute, yet it won't steal your focus completely if you are working. I also like that I can tell how many new messages I have at a glance. If you want to extend the script further, I'd recommend separating your folders into regular and high priorities, and make the script blink different LEDs depending on which folders have new mail. If you have multiple e-mail accounts, you might even want multiple versions of the script with an LED assigned to each account. I'd say the possibilities are endless, but they aren't. You have only three LEDs to play with.

In this column, I've mostly mentioned checking for e-mail as a candidate for desktop notifications, but there are any number of other things for which you might want notifications, such as system temperature, LEDs triggered by the local mail or printer queues, notices triggered by RSS feed (or dare I say it, Twitter) updates, or even desktop notifications tripped by your server monitoring system. Why keep all those dæmons running silently in the background when you can make them speak up when there's something to report?■

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.

GENSTOR

Linux - FreeBSD - x86 Solaris - MS etc.

Proven technology. Proven reliability.

When you can't afford to take chances with your business data or productivity, rely on a GS-1245 Server powered by the Intel® Xeon® Processors.

Quad Core Woodcrest

2 Nodes & Up to 16 Cores - in 1U

Ideal for high density clustering in standard 1U form factor. Upto 16 Cores for high CPU needs. Easy to configure failover nodes.

Features:
- 1U rack-optimized chassis (1.75in.)
- Up to 2 Quad Core Intel® Xeon® Woodcrest per Node with 1600 MHz system bus
- Up to 16 Woodcrest Cores Per 1U rackspace
- Up to 64GB DDR2.667 & 533 SDRAM Fully Buffered DIMM (FB-DIMM) Per Node
- Dual-port Gigabit Ethernet Per Node
- 2 SATA Removable HDD Per Node
- 1 (x8) PCI_Express Per Node

Servers :: Storage :: Appliances

Genstor Systems, Inc.

780 Montague Express. # 604
San Jose, CA 95131
Www.genstor.com
Email: sales@genstor.com
Phone: 1-877-25 SERVER or 1-408-383-0120

intel Xeon inside

Intel®, Intel® Xeon®, Intel® Inside® are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

A Pico-Sized Platform with Potential

DIRK ELMENDORF

Desktop computing in the palm of your hand.

I saw a short video on YouTube about something called a Beagle Board, and it looked really interesting. It was this incredibly tiny board with plenty of ports. It even had an HDMI, which meant it would be easy to hook up to the LCD television in my living room. I put in my order to Digi-Key, and a few weeks later, it showed up. (I ordered it while Digi-Key was working on a new version, so your order should arrive faster.)

When the package arrived, and I pulled out the box, my first reaction was *wow*! This thing is tiny! When I opened the box, I realized that Digi-Key wasn't kidding by calling it the Beagle Board. The only thing in the box was a shiny red PCB (Printed Circuit Board). I had seen the board in the video, but until I got it, I didn't realize it came as a bare board. This led to my second order from Digi-Key. I didn't own the proper 5V power supply to make it work. I also added on a Beagle Board serial cable, which was lucky because it is an essential part of the setup process. I already had a serial-USB dongle, so if your computer does not have a serial port, you will need that as well.

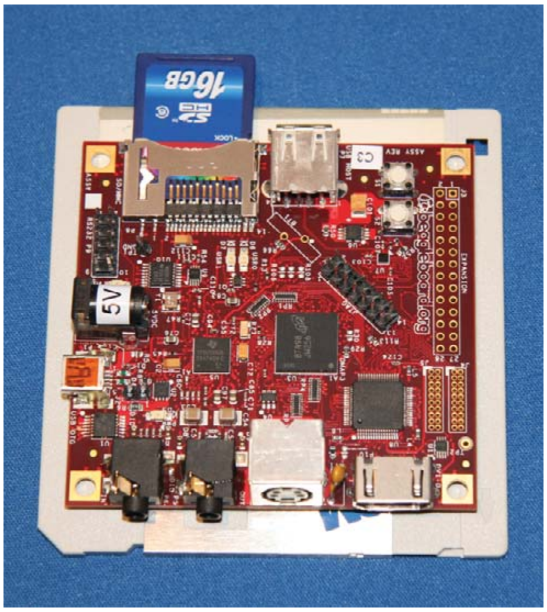


Figure 1. Up close with the Beagle Board—the gray thing underneath it is a floppy disk for scale.

Features

The Beagle Board has a ton of features built in, with options to add on more. The main feature of the Beagle Board, besides its 3"x3" form factor, is the OMAP 3530 processor. This tiny ARM processor handles all the heavy lifting on the board. The rev C. board that I have even has an extra USB port that is not shown in the picture, making it even easier to hook up USB peripherals to the board. In my case, that would be a USB hub, keyboard, mouse, Wi-Fi dongle and a thumbdrive.

History

The Beagle Board was developed by a very small team of engineers inside Texas Instruments. TI has a "small dreams" program where employees are given some funding to pursue a good idea. The first prototypes were released in June 2008. The engineers' goal was to provide a platform that gave users access to desktop-like performance without the bulk or expense. The team also was trying to find a way to make it easy for a hobbyist to pick up the board and experiment without requiring a lot of embedded experience.

I can say that in my time working with the Beagle Board, they were able to achieve both goals. It isn't the fastest computer I have ever used, but it is very usable. It is so tiny, I occasionally would misplace it on my work bench. On the embedded side, I had some bumps choosing the right components, but once I got a working system, it was very straightforward.

Tools

A great place to get started is at **elinux.org/BeagleBoardBeginners**. The two big things to make sure of when you get into playing with the Beagle Board are compatibility and connectivity.

I used the Beagle Board serial cable to monitor what was going on. In several cases, I had incompatibilities (it took some tweaking to get the Beagle to show up properly on my living-room TV). I also grabbed a Wi-Fi dongle from Linksys, figuring that it would just work. In this case, it was too new. I went back and got the Belkin G F5D7050 that everyone else seems to be using, and I was able to get on the wireless network.

The Beagle Board beginners page actually links to a shopping list, showing you what you need and which versions to buy. I did not find that information until much later, so I am trying to help you avoid one of my mistakes.

Common Install

The Beagle Board bootloader needs to be configured in order to run Linux. The good news is that once you do this setup, you don't have to do it again. You actually can do this step before you do anything else, and it's a good way to make sure you have everything ready for communication with your Beagle Board.

First, install minicom on your Linux host. You will need to change some settings:

```
sudo minicom -s
```

Inside minicom, you need to enter the "Serial port setup". Make sure you set the correct device to which to talk. In my case, I have the serial cable connected to a serial-to-USB dongle. That means my serial device is /dev/ttyUSB0. Make sure the Bps/Par/Bits is set to 115200 8N1. Turn off all hardware and software control. Once that is done, save it as the default.

Now, connect the serial cable to your Beagle Board, start minicom, and power on the Beagle Board. You should see a message about the OMAP3530 being ready. It also will show you a countdown until autoboot. You need to press a key to stop the boot process, so you can change

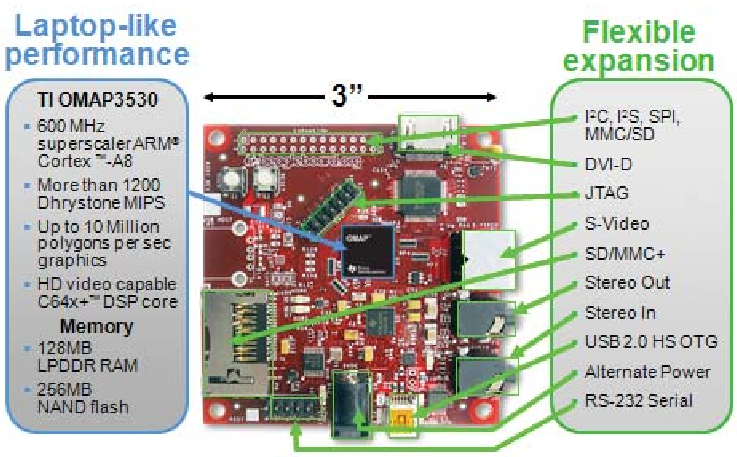


Figure 2. Detailed View of the Beagle Board

SILICON MECHANICS

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

intel Xeon inside™

Powerful. Intelligent.

Carlos knows a good thing when he sees it, but as one of the Silicon Mechanics Quality Assurance Experts, that's his job. When it comes to the new Storform iServ R515, Carlos sees more than just the care and attention to detail that go into the production of your custom-configured, versatile storage server.

The new R515 from Silicon Mechanics is a 4U server with 24 hot-swap SAS/SATA drives, optional integrated SAS controller, 6 PCI expansion slots, and redundant power. It features the new Intel® Xeon® processor 5500 series with Intel Turbo Boost Technology that delivers additional performance automatically when you need it. It also offers smart features like energy-saving DDR3 memory. With 2 internal SATA drive bays, you may want to think about incorporating Intel X25-E Extreme Solid State Drives for fast, reliable, energy-efficient OS drives. And Carlos will make sure that the servers we build and ship to you are exactly what you ordered.

When you partner with Silicon Mechanics, you get more than cost-effective, flexible, high-capacity storage—you get an expert like Carlos.

For more information about the Storform iServ R515 visit www.siliconmechanics.com/R515

Expert included.

some settings.

Once you are at the OMAP3 beagleboard.org prompt, type the following:

```
setenv bootcmd 'mmcinit; fatload mmc 0:1 0x80300000
➥uImage; bootm 0x80300000'
setenv bootargs 'console=ttyS2,115200n8 console=tty0
➥root=/dev/mmcblk0p2
rootwait rootfstype=ext3 ro omapfb.mode=dvi:1024x768MR-24@60'

saveenv
printenv
```

This configures your Beagle Board to look for the kernel image you are going to create. Now you need to get an SD card and prepare it to work for the Beagle Board.

SD Card Preparation

The important thing to understand about the SD card you're using is that it needs to be partitioned and formatted in a very specific way. Using fdisk or gparted, remove all existing partitions on the card. Then, create two partitions. The first one needs to be 50MB and formatted to fat32. The second partition can use the rest of the space on the card and should be formatted to ext3.

For the OS installs, you will be putting a uImage file on the fat32 partition. The rest of your files will go on the Linux ext3 partition. As you go through the process of building your install image, make sure you keep your uImage and Linux root filesystem in sync with the same kernel versions.

Ångström

The fine people at Ångström have worked hard on a distribution that is easy and stable to run on embedded devices. This is a good place to start if you just want to boot your Beagle Board into a Linux environment quickly.

I even found a handy tool to build your own custom Ångström image at **amethyst.openembedded.net/~koen/narcissus**. This is still experimental, but it was nice to be able to choose what I wanted in my image quickly.

If you want something even easier, Ångström also offers a tarball containing a working system that demonstrates the capabilities of the Beagle Board. Simply download the latest tarball from **www.angstrom-distribution.org/demo/beagleboard**, and get the uImage that has the same timestamp. Untar the tarball directly onto the ext3 partition of your SD card, and put the uImage (make sure you rename it to uImage) on the fat32 partition.

Ubuntu

The nice thing about Ångström is how easy it is to get up and running. The downside is I am not very familiar with that distribution as an environment. I ran into some problems configuring the Wi-Fi for the board, so I decided to switch to Ubuntu. Installing Ubuntu is more complicated upfront, but once installed, it was a lot easier for me to work with.

The other nice thing about Ubuntu is that Canonical announced late last year that it would officially support the ARM architecture (**www.ubuntu.com/news/arm-linux**). This means Canonical would be working on making sure that the packages I know and love on my x86 are all available on the ARM. Because the OMAP processor in the Beagle Board is an ARM chip, it will benefit from all this support.

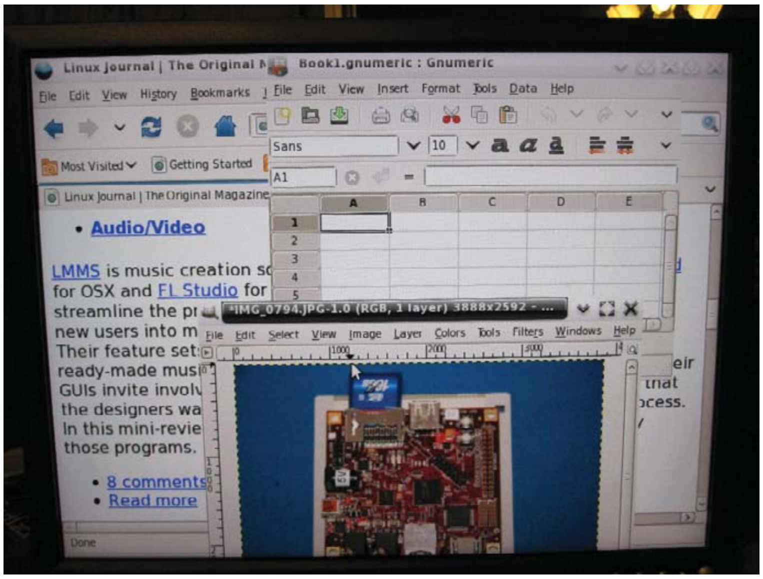
A step-by-step guide for installing Ubuntu can be found at **elinux.org/BeagleBoardUbuntu**. By the time this article is published, Karmic Koala should be officially released, so instead of giving you out-of-date instructions, I recommend you follow the link to get the latest information.

I actually installed Jaunty, as that was the stable version of Ubuntu at the time of this writing. Once it was up and running, any software I needed was simply an apt-get away (in most cases). In this case, that meant adding wicd (light network manager), Firefox and VLC.

Pitfalls

It would be easy enough to use the Beagle Board as a simple workstation. You can browse, look at documents, listen to music and even watch video. Those tasks demonstrate the board's flexibility combined with the wealth of open-source

Figure 3. Ångström Running GIMP, Gnumeric and Firefox


Linux Journal | The Original M
Book1.gnumeric : Gnumeric
*IMG_0794.JPG-1.0 (RGB, 1 layer) 3888x2592
Audio/Video
LMMS is music creation s
for OSX and FL Studio for
8 comments
Read more

software. On one hand, it is really amazing to have so much processing power packed into such a tiny form factor. On the other hand, turning the Beagle Board into a desktop seems like a waste of that form factor (unless you mount it inside a keyboard).

My initial goal was to use it as a set-top box. I was going to use it to stream Hulu to my living-room television. I did some research and found that you can build your own IR USB receiver (USB-IR-Boy). That would make it easy for me to interface a standard remote to the Beagle Board. Before I started ordering parts, I decided it would be better to see how good the streaming capabilities were. This walked me straight into a problem with no clear workaround. In order to stream Hulu, your browser must support Flash. At this time, the only Flash available for the ARM processor family is Flash Lite. Flash Lite seems to be licensed in a way that makes it impossible to distribute to a single end user. It is targeted more at handset vendors who will be shipping millions of units. I installed Gnash just to see if it was compatible, and it didn't work either. But, Adobe has announced that it is working on Flash 10 for ARM-powered devices (**www.adobe.com/aboutadobe/pressroom/pressreleases/200811/111708ARMAdobeFlash.html**).

So all is not lost. On the open standards front, I can hope that the progress being made on HTML 5 and video will remove the need for a proprietary solution to see video on the Web.

My first plan dashed, I realized there was another project that I've been meaning to play with—namely building a MAME cabinet. In honor of the Beagle Board's tiny size, I started looking at building a Mini-MAME cabinet. This basically is a smaller version of an arcade cabinet, and it has some advantages—namely that it will not take up too much space, and it will be usable by my one-year-old son sooner. Once again, before I ordered any parts, I installed all the MAME packages just to test things out. Bam! (That's the sound of me hitting another wall.) In this case, the MAME packages have been compiled and packaged, but they are not set correctly to run on an ARM processor. I made some attempts at hacking the Makefile to fix the problem, but I was unsuccessful. There are videos of the Beagle Board running MAME, but I was not able to duplicate their success.

The Future

Neither of these examples should dampen your enthusiasm for the Beagle Board. I included them to highlight the challenges involved in working in an embedded/non-x86 environment. The good news is that the software issues I ran into will likely be fixed as more energy is put into supporting the ARM platform.

All in all, it is a neat platform. It allows me to attack problems that are too small for a full-blown PC without requiring me to be an embedded expert (which I'm not).■

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

ASA COMPUTERS

Want your business to be more productive?
The ASA Servers powered by the Intel Xeon Processor provide the quality and dependability to keep up with your growing business.

Hardware Systems for the Open Source Community - Since 1989.
(Linux, FreeBSD, NetBSD, OpenBSD, Solaris, MS, etc.

1U Server - ASA1401i

- 1TB Storage Installed. Max – 3TB.
- Intel Dual core 5030 CPU (Qty-1), Max–2 CPUs
- 1GB 667MGZ FBDIMMs Installed.
- Supports 16GB FBDIMM.
- 4X250GB htswap SATA-II Drives Installed.
- 4 port SATA-II RAID controller.
- 2X10/100/1000 LAN onboard.

2U Server - ASA2121i

- 4TB Storage Installed. Max – 12TB.
- Intel Dual core 5050 CPU.
- 1GB 667MGZ FBDIMMs Installed.
- Supports 16GB FBDIMM.
- 16 port SATA-II RAID controller.
- 16X250GB htswap SATA-II Drives Installed.
- 2X10/100/1000 LAN onboard.
- 800w Red PS.

3U Server - ASA3161i

- 4TB Storage Installed. Max – 12TB.
- Intel Dual core 5050 CPU.
- 1GB 667MGZ FBDIMMs Installed.
- Supports 16GB FBDIMM.
- 16 port SATA-II RAID controller.
- 16X250GB htswap SATA-II Drives Installed.
- 2X10/100/1000 LAN onboard.
- 800w Red PS.

5U Server - ASA5241i

- 6TB Storage Installed. Max – 18TB.
- Intel Dual core 5050 CPU,
- 4GB 667MGZ FBDIMMs Installed.
- Supports 16GB FBDIMM.
- 24X250GB htswap SATA-II Drives Installed.
- 24 port SATA-II RAID. CARD/BBU.
- 2X10/100/1000 LAN onboard.
- 930w Red PS.

8U Server - ASA8421i

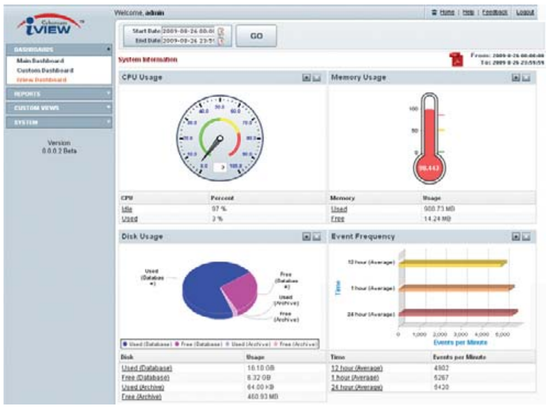
- 10TB Storage Installed. Max – 30TB.
- Intel Dual core 5050 CPU.
- Quantity 42 Installed.
- 1GB 667MGZ FBDIMMs.
- Supports 32GB FBDIMM.
- 40X250GB htswap SATA-II Drives Installed.
- 2X12 Port SATA-II Multilane RAID controller.
- 1X16 Port SATA-II Multilane RAID controller.
- 2X10/100/1000 LAN onboard.
- 1300 W Red Ps.

All systems installed and tested with user's choice of Linux distribution (free). ASA Collocation—$75 per month

2354 Calle Del Mundo,
Santa Clara, CA 95054
www.asacomputers.com
Email: sales@asacomputers.com
P: 1-800-REAL-PCS | FAX: 408-654-2910

intel Xeon inside™ Powerful. Efficient.

Cyberoam iView



In an effort to answer the question "Who is doing what on my network?", the company Cyberoam has released Cyberoam iView, a new, open-source logging and reporting solution. Cyberoam iView, says its creator, delivers centralized identity-based logging and reporting of multiple devices across geographical locations, thus enabling organizations to meet their security management and regulatory compliance requirements. The application further solves many drawbacks, such as the expense of existing logging-reporting solutions or the need to correlate individual logs from multiple devices like firewalls, antivirus and antispam solutions, intrusion-prevention solutions, proxy servers, routers, operating systems and more. One already can derive logs and reports via Linux iptables/Netfilter firewall, the popular open-source HTTP proxy Squid and other commercial UTM firewall solutions.

www.Cyberoam-iView.org

InMage Scout

The disaster recovery application InMage Scout—InMage's flagship product—recently was upgraded to version 5.1. InMage Scout is a single platform that supports transparent backup, quick disaster recovery from catastrophic failures and automated application failover/failback for improved restoration of application services. Enhancements in this new v5.1 include enhanced support for larger environments and multi-tenancy features for MSP customers, as well as improved scalability, platform coverage and ease of use. Support for Sun Solaris also has been added to the existing support for Linux and Windows.

www.inmage.com

AXIGEN Mail Server

If you're in the market for a mail server with a slick Web-based e-mail application, AXIGEN hopes you'll try its new Mail Server Version 7.2. AXIGEN says that service providers will appreciate the new AJAX-based Webmail, a "cool Web experience" for its users that "will help them create new services and generate new streams of income" due to its strong focus on monetization. The application provides multiple, customizable advertising capabilities and seamless integration with third-party applications, such as portals and community-related tools, thus helping SPs keep their customers on-line for a longer period. The application introduces features such as keyboard navigation and shortcuts, drag and drop, live e-mail list view, frequent folders and also allows users to employ shortcuts and time-saving tricks they already have been using with classic desktop e-mail clients, such as Outlook or Thunderbird. Supported platforms include Linux, various BSDs, Solaris and Windows.

www.axigen.com

WinSystems EBC-Z8510-G Single-Board Computer

The latest single-board computer to come from the house of WinSystems is the EBC-Z8510-G, this one powered by the Intel Atom processor (1.1GHz or 1.6GHz) and integrating the new COMIT (Computer On Module Interconnect Technology) and SUMIT-ISMT I/O expansion standards. The little guy measures in at 203mm x 147mm (8.5" x 5.75"). The I/O interface features two Gigabit Ethernet ports, CRT and LVDS flat-panel video, a MiniPCIe card interface for a wireless networking module, four USB 2.0 ports, four serial COM ports, HD audio, PATA controller for both a CompactFlash and hard disk, 48 lines of digital I/O, LPT and a PS/2 port for keyboard and mouse. The EBC-Z8510-G supports Linux and Windows OSes and development kits.

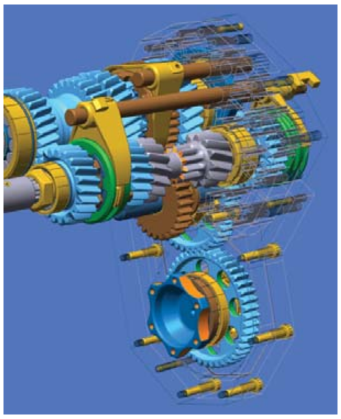
www.winsystems.com

Napatech's Time Synchronization Technology

Napatech has added extremely accurate packet timestamping—important for measuring quality-of-service factors in packet networks, such as latency—to its line of intelligent network adapters for real-time network analysis. The new time synchronization solution enables Napatech network adapters to be synchronized with a variety of time sources, such as GPS, IEEE 1588v2, CDMA and Pulse Per Second sources. This feature allows packets to be timestamped with an accuracy of 50 nanoseconds. It also lets Napatech adapters be daisy-chained, allowing a single time synchronization source for multiple adapters. Napatech calls its adapters "ideal for OEM network appliance vendors in the network performance monitoring, test and measurement, security and optimization markets." An extensive software suite is provided for integration supporting Linux, FreeBSD and Windows.

www.napatech.com

VariCAD



No stranger to CAD on the Linux platform, VariCAD, now in version 2009 2.0, is a 3-D/2-D CAD system intended for use in mechanical engineering design. Core features include tools for 3-D modeling and 2-D drafting and dimensioning, libraries of standard mechanical parts (ANSI, DIN), calculations of standard mechanical components, and tools for working with bill of materials (BOM) and blocks. This version adds new features like improvements in the geometric constraint module, parameters and geometric constraints within solid creation profiles, parameters for angles within a Boolean tree and improvements in areas such as solid insertion and transformation, selection of parts and printing capabilities. A free 30-day trial version is available for download.

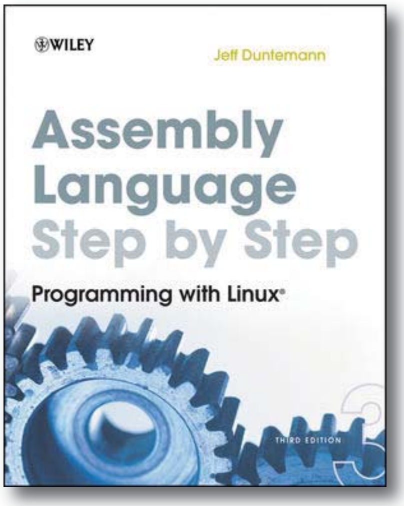
www.varicad.com

Arkeia Network Backup

The latest iteration of Arkeia Network Backup, version 8.1, which is designed for hosting providers that seek to generate revenue by offering backup services to their customers, offers a range of new features. These include Custom Restore Objects (CROs) that allow system administrators to assign restoration rights to end users and extensible reporting (for example, preconfigured reports, new tools for custom report generation and more ways to receive reports). In addition, support has been added for AIX 6, Fedora 11, NetBSD 5.0, OpenBSD 4.5, Mac OS X Snow Leopard and Microsoft Windows 7.

www.arkeia.com

Jeff Duntemann's *Assembly Language Step by Step: Programming with Linux* (Wiley)



New on bookstore shelves is the third edition of Jeff Duntemann's *Assembly Language Step by Step: Programming with Linux*, an introduction to the x86 assembly language. Although this new revision has been rewritten to focus on 32-bit protected-mode Linux and the free NASM assembler, the book retains Duntemann's distinctive lighthearted style as he presents a step-by-step approach to this difficult technical discipline. Duntemann starts by explaining the basic ideas of programmable computing, the binary and hexadecimal number systems, the Intel x86 computer architecture and the process of software development under Linux. From that foundation, he systematically treats the x86 instruction set, memory addressing, procedures, macros and interface to the C-language code libraries upon which Linux itself is built. The book assumes no prior experience in programming.

www.wiley.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

Discrete Geometry Viewer—Image Analysis and Manipulation

qcplusplus.sourceforge.net

Developed by Australian PhD student Shekhar Chandra at Monash University, Discrete Geometry Viewer (DGV) is an intriguing piece of software. As a general package, DGV, along with its various extensions, is a quantum mechanical toolkit and 3-D viewer for C++. It allows data visualisation via images, surface and volume plots using OpenGL, as well as rapidly developed Quantum Mechanical Simulations. It uses the Blitz++, VTK visualisation and open-source Qt libraries.

Breaking all that down, DGV allows you to do some really cool things with images, whether your interest is scientific or purely artistic. The program started out as a theoretical physics project under the moniker Quantum Mechanical Simulator, and Shekhar's main issue was in viewing the actual data, so he wrote DGV to fill the gap. As time goes on, Shekhar will be adding more of his PhD research work into DGV.

Quoting Shekhar, future advancements will include:

- Pixel values within viewer.
- Saving animations.
- Python shell rather than simple console output. See my project called QPythonShell, which allows one to embed a Python shell into Qt applications.
- More file formats.
- More transforms, like the Number Theoretic Transforms (via my new Number Theoretic Transform C library).

Installation Click on the Download link at the main Web site, and you'll be taken to a SourceForge page of hosted files. Under the Discrete Geometry Viewer heading, grab the latest package according to your system. Provided are x86 binary tarballs for Linux and Windows, as well as a source tarball. I went with the binary, which didn't have any dependency issues on my system and ran straight off the bat.

You can compile from source if you really want to—especially if you don't have an Intel-based machine—but the list of requirements is fairly stringent and may be a little obscure for many systems (see the project's Web site for more details).

Download and extract the tarball, and open a terminal in the new folder. To run the program, enter:

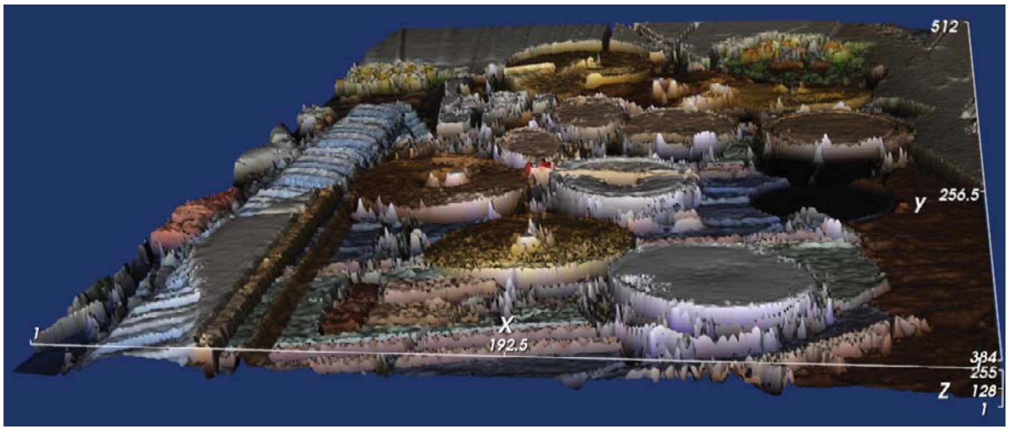
```
$ ./dgv
```

Usage First, you need to import a picture. Any picture will do, but in terms of number crunching, something with a smaller resolution (say 800x600) and common aspect ratio (such as 4x3 or 16x9) will make things easier, as both you and the computer will end up doing a fair bit of mathematical work. Open whichever image you like with File→Open, and the image will come up on-screen. The image that appears probably will be in grayscale depending on the release version, but don't fret, it isn't necessarily going to stay that way.

Now, let's go straight to the program's coolest ability. Right-click on the image and choose Surface Plot with Image from the drop-down box. Then, wait a while. There's a lot of math to be crunched, but it will be a quick process if you're lucky. A new 3-D landscape now will be on-screen (and back in color), which can be moved and rotated in real time and viewed at any angle.

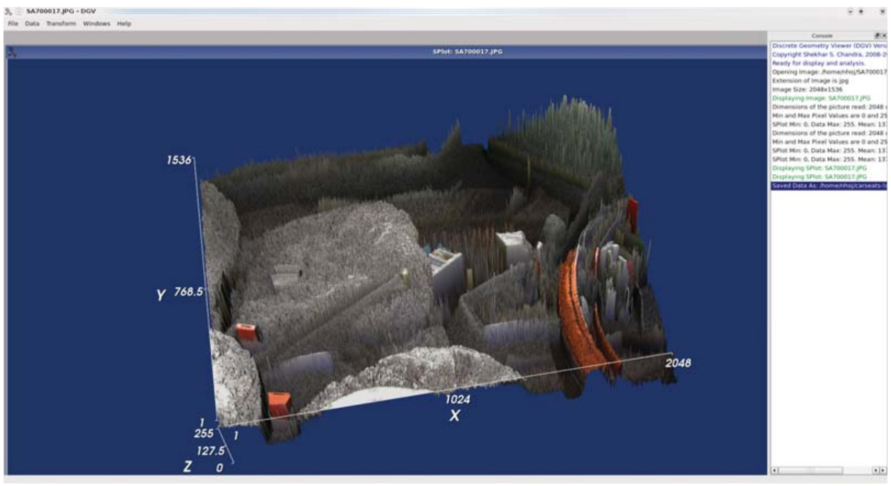
Left-click and move the mouse forward and backward, and the world tilts accordingly. Move the mouse right or left, and the world spins in that direction. Hold Ctrl while moving the mouse, and the image rotates in front of you as if it were on a 2-D plane, clockwise or anti-clockwise. Hold Shift or your middle mouse button, then move the mouse, and you can physically drag the object horizontally or vertically within your screen.

If you find the default values and the landscape they spawn a little crude (or even a little subtle), right-click on the image and choose Scale Factor. Decrease the given value, and the resulting terrain becomes smaller and closer to the original image. This can be



DGV Creating a 3-D Texture from a Photo of My Old Drumkit

My Old Drumkit



```
Discrete Geometry Viewer (DGV) Vers
Copyright Shekhar S. Chandra, 2008-2
Ready for display and analysis.
Opening Image: /home/nhoj/SA700017
Extension of image is jpg
Image Size: 2048x1536
Displaying Image: SA700017.JPG
Dimensions of the picture read: 2048
Min and Max Pixel Values are 0 and 25
SPlot Min: 0, Data Max: 255, Mean: 13
Dimensions of the picture read: 2048
Min and Max Pixel Values are 0 and 25
SPlot Min: 0, Data Max: 255, Mean: 13
SPlot Min: 0, Data Max: 255, Mean: 13
Displaying SPlot: SA700017.JPG
Displaying SPlot: SA700017.JPG
Saved Data As: /home/nhoj/carseats-
```

DGV made this amazing 3-D landscape from a picture inside my car!

used to apply some very subtle image enhancements to great effect. Increase the given value, and the bumpiness of the terrain becomes larger and more exaggerated.

This by itself, however, is more of a gimmick to show off to your mates. At the heart of this project is its mathematics and plotting abilities, combined with techniques to manipulate images, that can result in some stunning outcomes.

Let's see this in action with something a little more traditional. Close any working project windows and start again from scratch with a basic 2-D image. With the file open, right-click on the image itself and choose Data from the drop-down box. A table of data will be made, and it's this table that is especially important.

Each cell of numbers contains information that affects any geometry or effects you generate from this table. To put it in English, if you know what you're doing, you can control the way the final image ends up manually. Let's use Fast Fourier Transformation as a working example. (For the purposes of space, we'll run with my working filename, which is whole-kit.jpg. Substitute your own file in place of it.)

Right-click on the table and choose Transform→Fourier→FFT. After a moment, a dotted grayscale picture will come on-screen in a separate window above the original image. Now, combine these two into a final image. Click Data→Operate from the above menu. In the new dialog window, choose

Example FFT Image

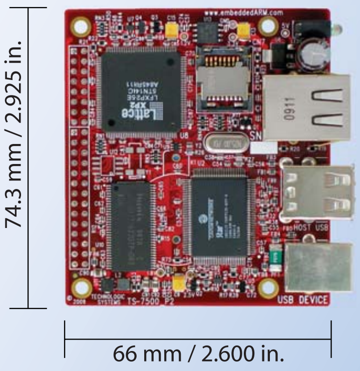
Multiply under Operation, whole-kit.jpg under Data Source 1 and Image: FFT-whole-kit.jpg under Data Source 2. And, there's a spiffing new image! The original image will be combined with the grainy FFT image to make a look that is unique to each picture.

We've barely scratched the surface here, so it's well worth checking out Shekhar's tutorial (**code.google.com/p/discrete-geometry-viewer/wiki/Home**) and blog (**l3mmings.blogspot.com**) to understand what this program really is capable of doing (and apologies to Shekhar for any inaccuracies there may be in this article). For anyone looking to explore this very different area of image processing, DGV definitely is worth a look.

TS-7500
Embedded Computer

Faster. Smaller. Cheaper.

Qu. 100 **$84**



Powered by a
250 MHz ARM9 CPU

- Low power, fanless, < 2 watts
- 64MB DDR-RAM
- 4MB NOR Flash
- Micro-SD Card slot - SDHC
- USB 2.0 480Mbit/s host (2) slave (1)
- 10/100 Ethernet
- Boots Linux in less than 3 seconds
- Customizable FPGA - 5K LUT
- Power-over-Ethernet ready
- Optional battery backed RTC
- Watchdog Timer
- 8 TTL UART
- 33 DIO, SPI, I^{2}C

Dev Kit provides out-of-box development + extra features

- Over 20 years in business
- Never discontinued a product
- Engineers on Tech Support
- Open Source Vision
- Custom configurations and designs w/ excellent pricing and turn-around time
- Most products ship next day

Technologic SYSTEMS

We use our stuff.

visit our TS-7800 powered website at

www.embeddedARM.com

(480) 837-5200

peekabot—3-D Robotic Visualization

www.peekabot.org

According to the project's Web site:

> peekabot is a distributed real-time 3-D visualization tool for robotics researchers and developers, written in C++. Its purpose is to simplify the visualization needs faced by a roboticist daily—using visualization as a debugging aid or making fancy slides for a presentation, for example.
>
> Our goal is to provide a flexible tool that will cater to the vast majority of a roboticist's visualization needs, yet is easy to use. Typical scenarios include visualization of simulations, data display from real robots and monitoring of remotely deployed robots.
>
> ...to enable remote data visualization, peekabot uses a distributed client-server architecture. All the gory details of networking is handled by the client library, used by your programs.

Installation Head to the Web site, and grab the latest tarball. In terms of library requirements, the documentation helpfully states the following (note, in the list below: *not required when building only the client API; **needed only if building the unit tests, which are disabled by default):

- A decently recent version of GCC.
- Boost 1.34.0+ (Boost.Thread, Boost.DateTime, Boost.Filesystem*, Boost.ProgramOptions* and Boost.Test**).
- Xerces-C++ 2.7.0+*.
- FLTK 1.1.6+*.
- OpenGL*.
- GLU*.
- libpng*.

I also had to grab these development files: libxerces-c2-dev and libfltk1.1-dev. Once you download the tarball, open a terminal wherever it's saved and enter the following:

```
$ tar xvzpf peekabot-x.y.z.tar.gz
$ cd peekabot-x.y.z
$ ./configure --prefix=/usr
$ make
$ sudo make install
```

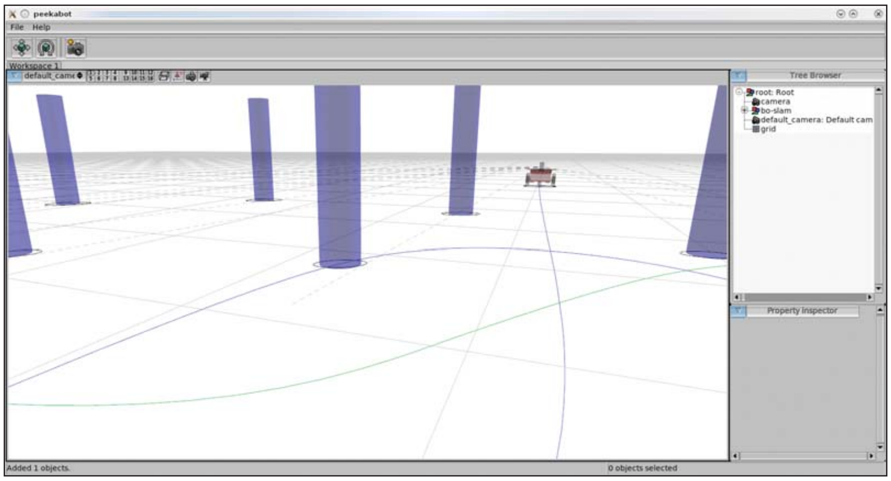
Assuming all went well, when the compilation has finished, you can run the program with the command:

```
$ peekabot
```

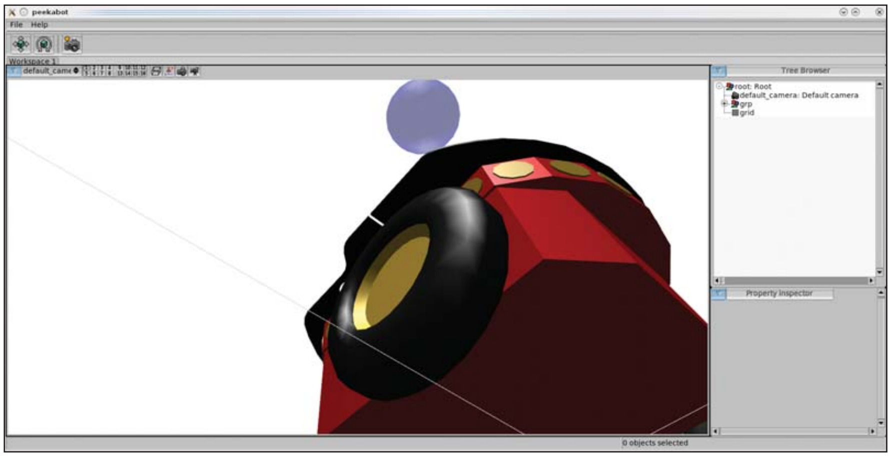
Before we jump in, I have to warn you that we've covered only half the equation. peekabot is made of two key parts: the server and its clients. After the initial building process, you will have the server by itself. The server is the main GUI screen where you'll be testing and interacting with your client programs. The clients generally will be standalone programs that communicate with the peekabot server while following their own coding structure. Although this may be daunting for new users (me included), it does make the system very open, powerful and flexible.

Okay, I'll assume you have the server window open and are ready to take peekabot for a spin. Let's take a look at a working example program and explore the GUI while it's running.

Open another terminal in the peekabot source directory, and look under the examples folder. Here you will see the folders bo-slam, results and skeleton. Enter any of these folders, and run the command `make`.



peekabot's low-level control of actions allows for some very advanced scripting, such as the object pathing shown here.



A number of Blender-created models allow for some snazzy active objects, available freely on the Web.

After make has compiled each example's code, a new program will be available in the same directory. To use bo-slam as an example, here's the terminal commands to enter (we'll assume you have a folder open in the peekabot main folder):

```
$ cd examples
$ cd bo-slam
$ make
$ ./bo-slam
```

Run by itself, nothing will happen, and you will get a bunch of error messages in the console. However, when started with the peekabot server running alongside, a robot and a bunch of pylons will appear, with the pylons moving themselves around the world into position and the robot making its way around each pylon on a preprogrammed path. Okay, now that we have something running, let's explore the GUI.

The main window contains the scene you'll be working with, along with a camera whose point of view is adjustable. The left mouse button will pan the camera, and the right button will rotate it. To zoom in and out, use either the middle mouse button or the mouse wheel. If you need the camera controls to be more or less sensitive, the Shift and Ctrl keys will modulate the sensitivity accordingly (useful in extreme close-ups or when looking from very far away, for example).

On the right is the Tree Browser, which contains all the active objects, scenery elements and so on. You can select objects in view by left-clicking, and multiple objects can be combined selectively like any file manager, using Ctrl to toggle select and Shift to add to the selection.

I've covered only the basic operations of this program here, because you'll need to do some actual coding to get into the nitty-gritty of this program. And quite frankly, I'm rubbish at coding! If you're interested in learning more, check out the basic documentation (manual: **www.peekabot.org/doc/latest/manual_the_basics.html**; models: **sourceforge.net/apps/mediawiki/peekabot/index.php?title=Model_repository**). There you will find the coding to get you started, along with its structure and so on.

Ultimately this looks to be a powerful project for robotic visualization, albeit a rather intimidating one. Despite the relatively difficult learning streak from its mostly coding-based interface, it's probably this same kind of interaction that will bring it longevity. Not having the restrictions of GUI design to hold back the mechanics surely will be a godsend to those who want to approach their mechanical design at a low level without the restrictions that accommodating to beginners so often imposes.■

John Knight is a 25-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.

Project at a Glance

Danger from the Deep

dangerdeep.sourceforge.net

Danger from the Deep is a free, open-source, WW2 German submarine simulator. The Web site says, "This game is planned as tactical simulation and will be as realistic as our time and knowledge of physics allows". I've had a brief chance to play it, and so far, I've seen a game full of polish and passion. I'm looking forward to covering this more next month.

Danger from the Deep

DEDICATED SERVERS WITH BLAZING FAST CONNECTIONS

Gigabit ports / MULTI-Gig options
High-capacity bandwidth plans, including:
* 3000 GB/month for $200
* 5000 GB/month for $375
* 10000 GB/month for $800
Custom clusters with private VLANs
Flexible storage and RAID options
Intel Premium Partner
Numerous OS choices (Linux or Windows)
FREE 24x7 "6-Star" support

www.CARI.NET/LJ
888.221.5902
carinet
Better Servers. Better Service

HARDWARE

The Goggles, They Do Something

How do video goggles designed for gaming on Windows stack up to a Linux geek's desktop? Find out below. KYLE RANKIN

I'm a sucker for cyberpunk. It probably has to do with all those *Shadowrun* sessions when I was a kid, but even the worst cyberpunk movies and books can grab my interest. Although my friends shake their heads at the cheesy acting and special effects in movies like *Johnny Mnemonic*, I still love the concept of total immersion into your computer and virtual reality that you see in classic cyberpunk. With all of this in mind, it shouldn't surprise you that I've been keeping close watch on the current state of the art with consumer video goggles. Even though I'm not quite ready to become a *Snow Crash*-esque gargoyle just yet, I still jumped at the chance to review the Vuzix VR920 video goggles (**vuzix.com**, $399.95).

I've been following the Vuzix company's product line for a number of years—before they even were called Vuzix—and it's been interesting to watch the product line progress. At the moment, Vuzix has a few different consumer-grade video goggles: lower-res glasses aimed at portable video devices like the iPod, and the VR920 that it aims at the computer gaming market. All of the goggles include built-in headphones, and each model has different audio and video inputs. The AV920 and VR920 feature the higher-res 640x480 screens, but the AV920 still is designed to connect to video devices, such as portable DVD players, and includes a rechargeable battery and standard DVD player video inputs. If you want to connect goggles to a computer, you'll definitely want to go with the VR920, as it not only comes with a VGA connector, but it also can be powered from USB.

VR920 specifications (from the product page):

- Twin high-resolution 640x480 (920,000 pixels) LCD displays.
- Equivalent to a 62" screen viewed at nine feet.
- 24-bit true color (16 million colors).
- Visor weighs 3.2 ounces.
- 60Hz progressive scan display update rates.
- Fully iWear 3-D-compliant and supports NVIDIA stereo drivers.
- Built-in noise-canceling microphone for Internet VoIP communications.
- Built-in three-degree-of-freedom head-tracker.
- USB connectivity for power, tracking and full duplex audio.
- Analog VGA monitor input.
- Support for up to 1024x768 VGA video formats.

The VR920 definitely is aimed at the gaming market and has some pretty interesting features, such as an accelerometer that under Windows can be used (along with compatible games) to track your head movement, so when you turn your head left, for instance, your character's head turns left. The VR920 is powered by your USB port, and the USB connection also is used, so sound can be sent to the included earbuds. You also can take advantage of the NVIDIA stereo drivers under Windows to display different images for each eye and get a 3-D-like experience. Unfortunately, even though you can find a few homegrown projects to get basic accelerometer support and stereo video working under Linux, as of yet, I wouldn't call it fully functional, so in this review, I focus on what it would be like for average users to use the VR920 with their Linux desktops.

Look and Feel

Before I go into how to set up the hardware in Linux, I first should get something out of the way. You see, the primary thing that has made the video goggle market grow so slowly in my mind isn't so much the lower resolutions on the screens or the price, as much as the fact that you still look a little bit dorky wearing any of the major vendors' goggles. I mean, we are all geeks here, so we are used to looking a little bit dorky anyway. Plus, many people would think it's a bonus to look like a character from *Star Trek: The Next Generation*, but still, the look is not exactly for everyone. The next revision of

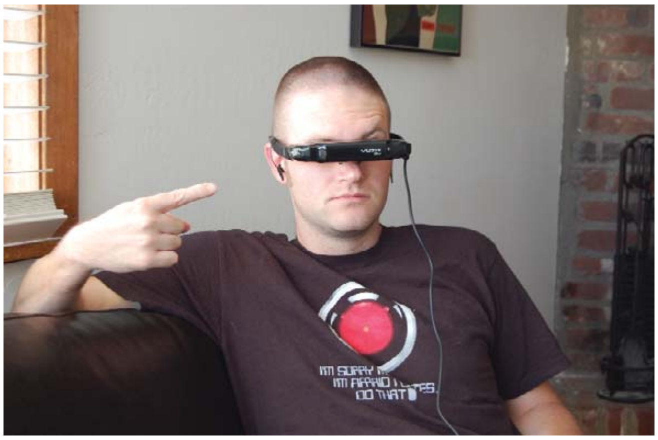


Figure 1. Space Odyssey Indeed

goggles apparently is going to address this issue somewhat, as they appear to look more like large sunglasses. Figure 1 shows a picture of me wearing the goggles, so you can see what they look like. It's definitely a fashion statement. I know some people will have no qualms walking around their daily lives with these on, but others will use them only in the privacy of their own homes.

Also, if you can't tell from the picture, these goggles don't completely obstruct your vision. You can arrange them so that they sit a bit higher on your nose, and if you lean back a bit, you can look up and see through the goggles and look down to see your computer screen. This means you still can use your regular display if you want and extend the desktop to the goggles.

So, I wasn't surprised that I looked a little dorky with the goggles on, and to be honest, I didn't care that much. What surprised me though was how uncomfortable the nose bridge was out of the box. I think one of the most important things you can do up front is adjust the nose of the goggles so it's comfortable. The goggles don't feel very heavy, but after a full movie, you will start to feel fatigue on your nose, especially if the bridge is pinching too much. Once I had everything adjusted, it was pretty comfortable, but I still wouldn't expect to wear them all day.

I also had a rather pleasant surprise with respect to eye fatigue, or the relative lack of it. After all, you have these screens very close to your eyes, so I figured my eyes would be focused very closely when I used them. It turns out that the way they have engineered the optics, they are telling the truth when they say it appears like a 62" screen viewed at nine feet. If you connected your computer output to a large LCD TV mounted on your wall and looked at it from your couch, it would look a lot like a desktop through the VR920. To be honest, the effect is so complete, I found myself squinting not because the image was too close, but because it appeared too far away! You see, I'm nearsighted, but it's mild enough that it doesn't affect my daily life in front of a computer. I wear my glasses only when I'm driving at night, at a presentation or when I'm watching a movie with subtitles. The downside for me is that it's a bit tricky to wear glasses and the goggles at the same time, but of course, if you use contact lenses, it wouldn't be a problem.

Hardware Setup

There is really very little hardware setup to do. Just plug in the VGA and USB plugs to your machine. As I mentioned before, the USB port is used to power the display as well as to send audio to the goggles (it's also how you would access the accelerometer). Here is the relevant syslog output I got when I connected the VR920 to my Ubuntu machine:

```
Sep 14 19:51:01 moses kernel: [13069.884651] usb 6-1:
 new full speed USB device using uhci_hcd and address 2
Sep 14 19:51:01 moses kernel: [13070.101323] usb 6-1:
 configuration #1 chosen from 1 choice
Sep 14 19:51:02 moses kernel: [13070.291377] usbcore:
 registered new interface driver hiddev
Sep 14 19:51:02 moses kernel: [13070.361931] usbcore:
 registered new interface driver snd-usb-audio
Sep 14 19:51:02 moses kernel: [13070.397112]
 generic-usb 0003:1BAE:0002.0001: hiddev96,hidraw0:
 USB HID v1.00 Device [Icuiti Corp. VR920 Video Eyewear]
 on usb-0000:00:1d.0-1/input3
Sep 14 19:51:02 moses kernel: [13070.397155] usbcore:
 registered new interface driver usbhid
Sep 14 19:51:02 moses kernel: [13070.397162] usbhid:
 v2.6:USB HID core driver
Sep 14 19:51:02 moses pulseaudio[11722]: alsa-util.c:
 Cannot find fallback mixer control "PCM" or mixer
 control is no combination of switch/volume.
Sep 14 19:51:03 moses pulseaudio[11722]: alsa-util.c:
 Device hw:1 doesn't support 2 channels, changed to 1.
Sep 14 19:51:03 moses pulseaudio[11722]: module-alsa-source.c:
 Your kernel driver is broken: it reports a volume range
 from 0 to 0 which makes no sense.
Sep 14 19:51:03 moses pulseaudio[11722]: module-alsa-source.c:
 Your kernel driver is broken: it reports a volume range
 from 0.00 dB to 0.00 dB which makes no sense.
```

It turns out that the way they have engineered the optics, they are telling the truth when they say it appears like a 62" screen viewed at nine feet.

So pulseaudio does appear to see the audio mixer, and even though it throws some strange errors in syslog output, the sound worked fine. From the output, I can tell that it sees it as the hw:1 ALSA device. I also saw a new /dev/dsp1 device, and the device even showed up in my GNOME sound properties window, so I could select the device from there.

The screen itself is detected without any extra effort on my part and shows up in xrandr:

```
greenfly@moses:~$ xrandr
Screen 0: minimum 320 x 200, current 1280 x 800, maximum 1280 x 1280
VGA connected (normal left inverted right x axis y axis)
   1024x768       60.0
   800x600        60.3
   640x480        59.9
   720x400        70.1
```

You certainly can use xrandr to toggle whether the display is on, but you also can configure it in the default GNOME Display Preferences window (Figure 2). Personally, I set up a quick xrandr script to toggle the goggles on when I wanted to use them:

```
#!/bin/sh

if [ -f /tmp/.goggles_on ] ; then
        xrandr --output VGA --off &
        echo "Goggles Off" | osd_cat --shadow=2 --align=center
        ➥--pos=bottom --color=green --delay=2
        ➥--font=lucidasanstypewriter-bold-24 --offset 40 &
        rm /tmp/.goggles_on
else
```

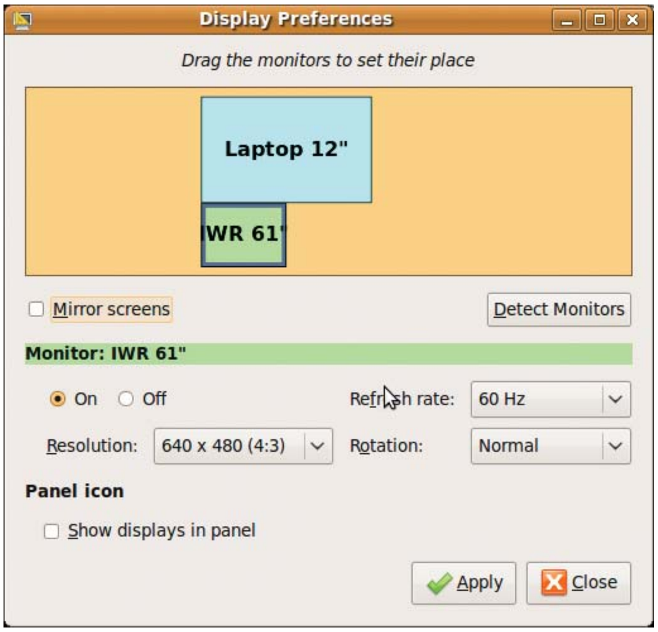


Figure 2. GNOME Display Preferences

```
        xrandr --output VGA --mode 640x480 --below LVDS
        echo "Goggles On" | osd_cat --shadow=2 --align=center
        ➥--pos=bottom --color=green --delay=2
        ➥--font=lucidasanstypewriter-bold-24 --offset 40 &
        touch /tmp/.goggles_on
fi
```

In this configuration, the goggles act as a 640x480 screen below my regular desktop, and I can drag windows there just like with any other monitor.

General Desktop Use

The main thing to realize when you use the VR920 like a regular monitor is that even though the display supports 1024x768 input, the actual screens go up to only 640x480, and anything higher resolution gets scaled to fit. For my uses, I stayed with 640x480. Honestly, at that resolution, the screen worked pretty well as an extra screen on my desktop, and I moved IRC windows and other small terminals over to it.

The main limiting factor for how useful the screen is on a regular desktop is the resolution, but nearsightedness aside, I thought it actually was a pretty slick way to have IM, IRC or a terminal window always in your field of view. As a sysadmin, I also can see how it might be useful to tail logfiles in that screen or possibly put all of your monitoring applets there. Just realize that if you arrange the goggles so you can look up to view them and look down to view your regular screen, you won't necessarily be able to use your peripheral vision to see changes in the goggles—you'll have to look up every now and then. Also, I don't think it would work quite as well for programs like The GIMP or for word processing or anything else that might need more screen real estate.

Video Games

Video games are the main function the VR920 is geared toward, and I wasn't surprised that it worked really well for that purpose. Again, the display is 640x480 at maximum, so don't expect complete realism. Now, I'm an old-school *Quake* gamer, so I just had to see what *Quake 3* was like through the goggles. Once the desktop was set up to span to that screen, I didn't have to do any extra tweaking. I launched *Quake 3*, and it output to both of my displays. Even though I didn't have the head-tracking feature enabled, I have to admit it was really cool to play games with the goggles on. You do feel even more immersed in the environment than normal, so any FPS (First-Person Shooter) games should work well here. Any other games that could benefit from more virtual-reality-like immersion, such as flight simulators, would get an extra dimension of fun here as well. I should let my fellow Point/Counterpoint columnist Bill try these out in *Second Life*. I bet it'll knock his socks off.

I think that apart from gaming, movie viewing is one of the best uses for the VR920 under Linux.

Movies

I think that apart from gaming, movie viewing is one of the best uses for the VR920 under Linux. The resolution isn't a real issue for anything up to DVD quality, and once I had adjusted the goggles so they were comfortable, they worked great for movies. Just be sure to tell your video player to use the correct audio device. For my machine, this just meant adding the `-ao alsa:device=hw=1,1` to MPlayer, but of course, it will vary from program to program. I have to say it was really nice to be able to watch an entire movie on my computer and still have my full desktop to use. In my mind, the killer app for this is airplane travel, as they are essentially headphones for your eyes—you can watch movies with complete privacy.

Conclusion

I had been anticipating all kinds of scenarios where I would use the VR920 before I had them. I could see myself at my desk at work displaying a terminal and maybe wearing them combined with some sort of video camera for augmented reality and become a true *Snow Crash* gargoyle. The reality of having the goggles around for daily use isn't quite as exciting though. To be perfectly honest, I didn't find myself using them nearly as often as I thought I would. Part of the reason is that I haven't been gaming much recently, but I think the main reason is due to the low resolution. There are only so many programs that work well in 640x480. Having said that, I'm definitely going to bring the goggles with me the next time I travel. If you do play a lot of FPSs or other games with a first-person perspective, or if you watch a lot of videos in public and are tired of people looking over your shoulder, I definitely think you should give the VR920 a try.■

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.

Celebrating 15 years of *Linux Journal*, we've brought together every article ever published in the world's #1 Linux magazine and packaged it in one convenient CD.

release date: february 2009
LINUX JOURNAL
ARCHIVE 1994-2008
Includes issues 1–176 of Linux Journal

With nearly 4,000 articles written by industry experts on everything from cool projects, desktop how-tos, security, embedded systems, networking, virtualization, multimedia, system administration and programming tricks and techniques—this unique collection is a must-have for every Linux enthusiast.

Get your *Linux Journal* Archive CD today featuring all issues from 1994 through 2008. Just $34.95.

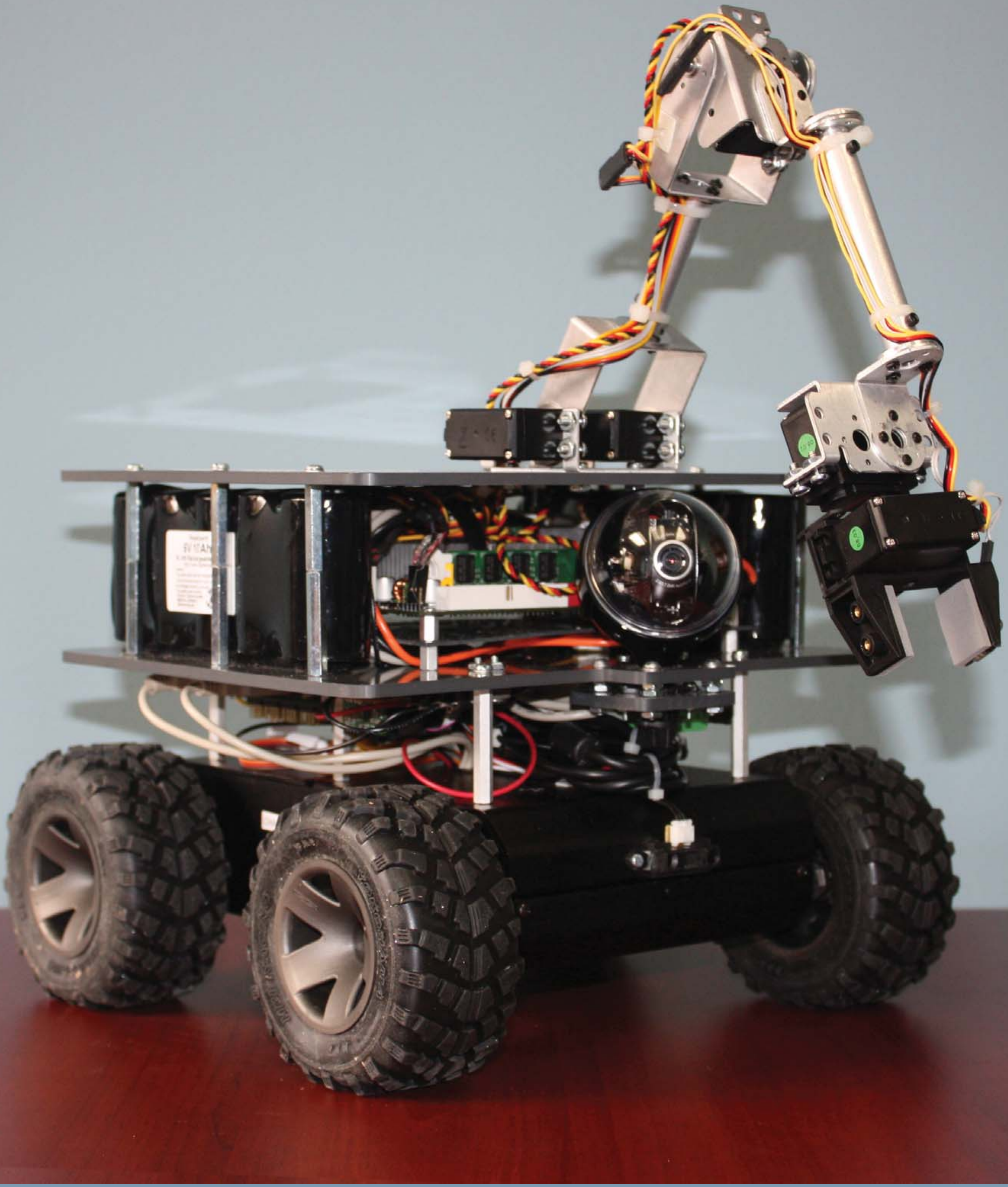
www.linuxjournal.com/archivecd

Playing with THE PLAYER PROJECT

THE PLAYER PROJECT IS A ROBOT-CONTROL SOFTWARE FRAMEWORK FOR INTERFACING WITH PC-BASED ROBOTS. LEARN HOW TO USE IT TO INTERFACE WITH SENSORS, ACTUATORS AND EVEN FULL RESEARCH ROBOTS. **KEVIN SIKORSKI**

We've all heard that PC-based computers have increased in power and decreased in size, power consumption and cost. These improvements mean more people have access to them, but also that PCs are becoming more suited to being the brains of a mobile robot. It brings to robots a number of advantages, such as USB connectivity, greater memory capacity, more powerful processors and even allows for plugging in a mouse, keyboard and monitor to debug your robot *on* your robot. The largest cost associated with the choice of a PC-based robot, besides the dollar cost, is power consumption.

Programming your PC-based robot can be a little different from programming a robot that uses a smaller, embedded processor. As you would expect, PC-based robots can take advantage of programming features, such as threads, using multiple languages and leveraging third-party libraries to perform complex tasks.



What Is Player?

One such third-party library is the Player Project. The Player Project is a framework for building robot-control software. It provides a wide-reaching infrastructure that gives you a network protocol, data serialization support, a message loop and supports a large number of common off-the-shelf components, such as Webcams, laser range finders, RFID, GPS devices and interface boards. It even supports a large number of commercially available robots, and some robot manufacturers, such as CoroWare, provide Player drivers for their robots. And, of course, it's open source.

Requirements to Run Player

The newest version of Player runs on the majority of Linux distributions with little effort. It is also cross-platform, with version 3.0 supporting Microsoft Windows and even Cygwin under Microsoft Windows. It has low memory requirements and is pretty easy on your CPU. Although it's not completely cross-language, it does offer native support for C and C++, and has nice Python and Ruby interfaces.

You don't really need a real robot to work with Player. If you have a few sensors and actuators that you can connect to your laptop or desktop computer, you still can run the Player server and control those disembodied devices—think of it as a way to accomplish home automation. Player also can be used with some of its close cousins: Stage (a 2-D simulation system that integrates with Player) and Gazebo (a 3-D simulation system that also integrates with Player). In this way, you can simulate a full robot, or even a fleet of robots, without any special hardware.

What Does Player Look Like?

Several layers of the infrastructure are diagrammed in Figure 1, which should be familiar if you have written code that interacts with hardware devices.

At the lowest level, we have the hardware layer. This corresponds to the hardware that embodies the sensor, actuator or any other physical component of the robot. The driver level sits on top of the hardware level. This is one layer where a programmer writes code. For example, a driver that interacts with a USB Webcam will provide whatever code is necessary to make a connection with the camera, read the sensor's output, package it up neatly and make it available to the rest of the system. A lot of drivers of this type are provided with Player as *static drivers*. This means you won't have to provide any special shared object libraries to use them, just the usual Player libraries. You also can write your own drivers, called plugins. The code for these drivers lives in a shared object library (.so file).

You can create multiple instantiations of a single driver. When you do this, you need a way to refer to a specific instance. For example, if your robot has two cameras, one facing forward and one facing backward, you will need to tell them apart to know in which direction you are driving. In order to do this, Player gives you the concept of a device. Each time you instantiate a driver, you assign it a device name and number. For example, in your robot, camera:0 might refer to the forward-facing camera, and camera:1 might refer to the backward-facing camera.

Finally, we arrive at the concept of an interface. The interface is just like the API for a software package; it defines the software interfaces for getting data out of and putting data into the device. In our example, the camera interface defines a set of messages to get images out of the camera, and a set of messages to set up the camera and capture images.

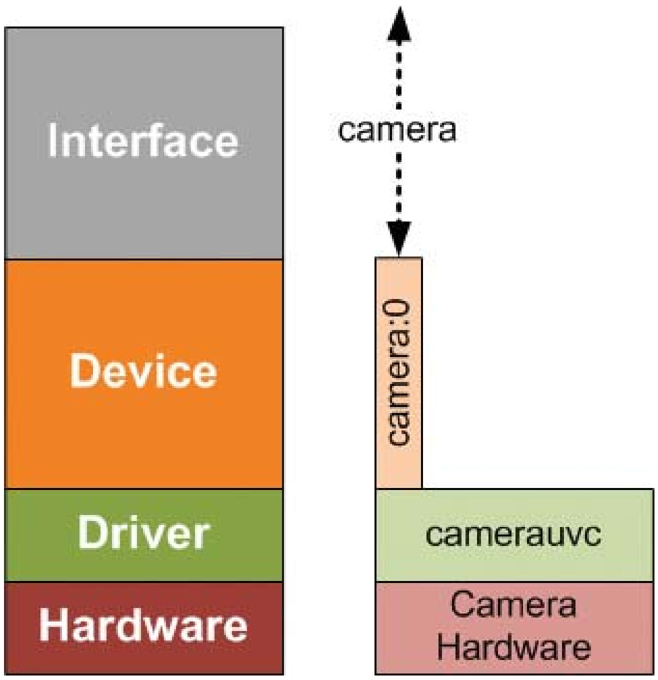


Figure 1. The relationship between Webcam hardware, the driver that interfaces with it, the device created by the driver and the interface that it exposes.

Drivers don't have to exist solely for communicating with a piece of hardware. Player supports a number of algorithmic drivers, such as blob-finders or wavefront planners, or even camera image compressors. These operate just like normal drivers, consuming and producing data and exposing interfaces.

Player Configuration

Now that we have dealt with the vocabulary, we can get down to business. Let's say you've just acquired a CoroWare CoroBot robot (Figure 2). You've installed Player, and you want to make the robot do something interesting. You can type `player corobot.cfg` to run the Player drivers on the robot. This loads a configuration file that describes all the drivers Player must load to make your robot work. Here's an excerpt of a configuration file for the CoroBot:

```
driver
(
  name        "corobot"
  plugin      "libcorobotdriver"
  provides    ["position2d:0" "power:0" "ir:0"
               "limb:0" "gripper:0" "ptz:0"]
  requires    ["aio:0"]
  ssc32port   "/dev/ttyS1"
  ptzport     "/dev/video0"
)
```

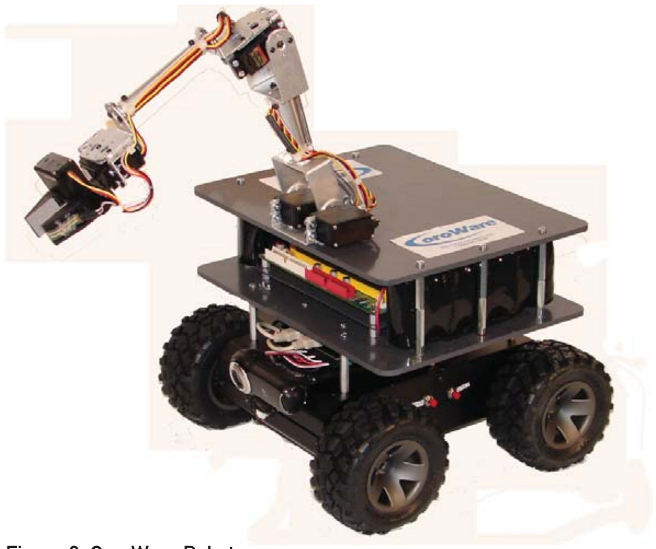


Figure 2. CoroWare Robot

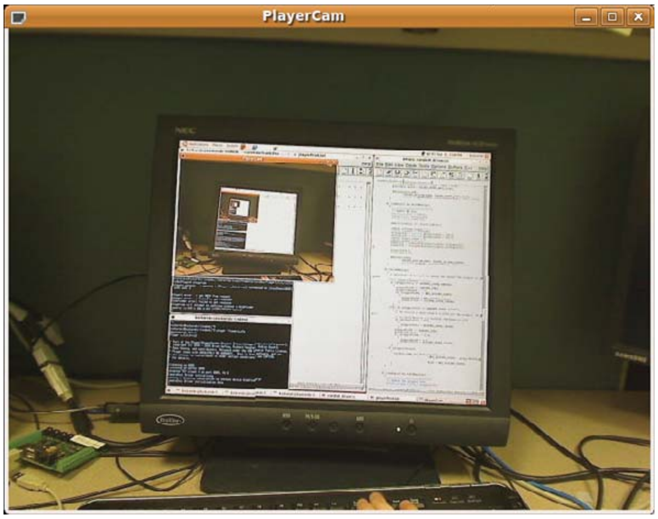


Figure 3. Player's playercam Tool, with the Webcam Pointed at the Computer Monitor

We start off by telling Player that we are constructing a driver. The first two lines in the driver description tell Player where it can find the relevant code: the `name` line indicates the name of the driver in the code (a single library can supply multiple drivers), and the `plugin` line indicates the name of the shared object library that contains the code for this driver. In this case, it is stored in libcorobotdriver.so, in the same directory as the configuration file. The `provides` line specifies the devices that this driver makes available—in this case, the CoroBot driver exposes six devices. The `requires` line specifies the devices that this driver will consume. If any devices here are not present on the system when Player tries to instantiate the driver, Player will abort.

The last two lines in the driver description are not standard. Any driver is free to parse its driver description and make use of special identifiers. The CoroBot driver uses two of these: ssc32port specifies the Linux device (a serial port) through which it will communicate with its servo controller and a video device that controls its pan-tilt camera.

Player Tools

Player provides a number of tools to make working with Player easier. For example, you can use the playercam program to view the image provided by a Webcam. Let's say you have Player installed on your computer and a simple configuration file that brings up a camera driver:

```
driver
(
    name     "camerauvc"
    provides ["camera:0"]
    port     "/dev/video0"
    size     [640 480]
)
```

You can run Player with this configuration file with `player camera.cfg`, and then run `playercam` to see the camera image in real time (Figure 3). If your Webcam is on another computer—for example, on a PC-based robot—and connected by a network, you can just as easily see the Webcam output by running:

```
playercam -h hostname -p port
```

playerprint is a tool that works much like playercam, displaying the output of a device to the user. But, playerprint does this textually and can support a large number of interfaces, while playercam can support only the camera and blob-finder interfaces. For example, if we have a CoroBot running its Player drivers, we can display its infrared sensor readings with:

```
playerprint ir -h hostname -p port
```

Listing 1. Player's playerprint Tool Inspecting a GPS Device

```
#GPS (13:0)
#lat|long|alt|utm_e|
utm_n|err_horz|err_vert|num_sats|quality
47.6470103 -122.1414822  112.3      564477
5277425.1      0       0    6    1

#GPS (13:0)
#lat|long|alt|utm_e|
utm_n|err_horz|err_vert|num_sats|quality
47.6470107 -122.1414812 112.28    564477.07
5277425.14      0       0    6    1

#GPS (13:0)
#lat|long|alt|utm_e|
utm_n|err_horz|err_vert|num_sats|quality
47.6470113 -122.1414807 112.28    564477.11
5277425.21      0       0    6    1
```

THE COROBOT ROBOT COMES WITH A NUMBER OF SENSORS AND ACTUATORS—PROBABLY THE EASIEST OF WHICH TO INTERFACE WITH ARE THE FRONT- AND REAR-FACING INFRARED RANGING SENSORS AND THE MOBILITY BASE'S DRIVE MOTORS.

Player also lets you control your robot, not just inspect it. playerv is a utility that also knows how to interact with many interfaces. Once you have started the Player server on your robot, you can run it with `playerv` (if you are on the same machine) or `playerv -h hostname -p port` (if you are on another computer). playerv will show a graphical display of the world around your robot, but it does not automatically connect with any devices. You will have to go to the Devices menu, and subscribe to the devices that you are interested in playerv plotting. In order to drive your robot around, you'll want to subscribe to the position2d device and select the option to "command" the interface. Then, you will be able to drag a small targeting reticle around the window to drive the robot (Figure 4).

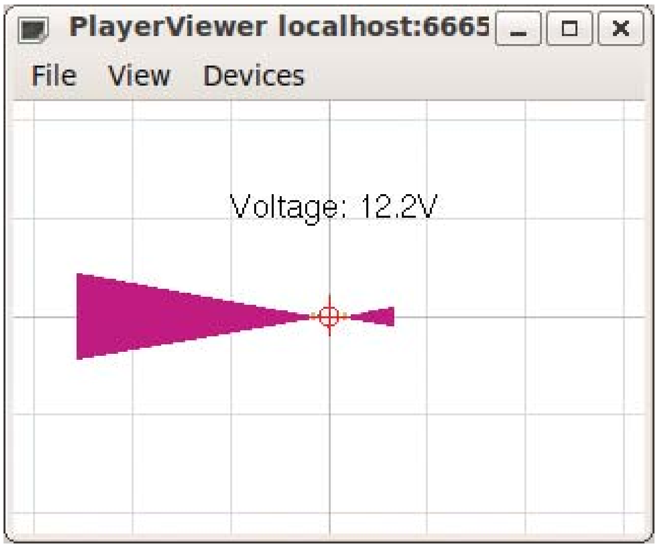


Figure 4. Player's playerv tool running on a CoroBot after subscribing to the IR and position2d devices. The large triangles are the cones shown to be obstacle-free by the infrared sensors.

Playing with Player

So far, our robot is awake, alert and ready to be told to do something interesting. Let's give it something to do. The CoroBot robot comes with a number of sensors and actuators—probably the easiest of which to interface with are the front- and rear-facing infrared ranging sensors and the mobility base's drive motors. Thus, we can write a small C program to talk to the Player server, read the IR sensors and drive the robot until it is 10cm away from an obstacle in front of the robot.

The first thing we have to do to interface with the Player server is open up a connection to it. For the sake of brevity, we will skip a lot of error checking, but you can download the full version of the code from the *LJ* FTP server (see Resources).

This code defines the variables we will use to talk to the Player server and the device interfaces in which we are interested:

```
#include "libplayerc/playerc.h"
static playerc_client_t*       clientHandle;
static playerc_position2d_t*   positionProxy;
static playerc_ir_t*           irProxy;
```

The clientHandle is used for talking to the Player server itself. The second position2d interface talks to the position2d interface, providing us with encoder information about how the wheels are moving and allowing us to send motor commands to the robot. We'll ignore the encoder information for this example. Lastly, the IR interface gives us information about the distances that the robot's IR sensors are reporting.

The next code snippet uses these proxies to interface with the server and these devices:

```
playerc_client_connect(clientHandle);

// convert our interface to a PULL interface,
// only updates when we read
playerc_client_datamode(clientHandle, PLAYER_DATAMODE_PULL);

// tell the robot to drop older messages
playerc_client_set_replace_rule(
          clientHandle, -1, -1, PLAYER_MSGTYPE_DATA, -1, 1);

// create the position proxy (controls the motors)
positionProxy = playerc_position2d_create(clientHandle, 0);
playerc_position2d_subscribe(positionProxy, PLAYER_OPEN_MODE);

// create the IR proxy (controls the IRs)
irProxy = playerc_ir_create(clientHandle, 0);
playerc_ir_subscribe(irProxy, PLAYER_OPEN_MODE);
```

We start off by connecting to the Player server and configuring our connection. We want to get new messages from the server only when we are ready for them, so we configure the connection for a pull-type arrangement. And, because we want only the most recent information (we don't care what the IR sensors were indicating a second ago, we care about what they are saying right now), we tell the server to report only the most recent data. If we really wanted, we could let Player ensure that every IR message was delivered, but that might result in getting less-than-fresh data and possibly driving into a wall.

After our connection is configured, we open up the position2d interface on the Player server and subscribe to it. Then, we do the same with the IR interface. So far, so good. Now we need to get the state of the IRs from the robot and tell it how to move the motors:

```
while (!timeToQuit) {
  // attempt to read from the client
  if (playerc_client_read(clientHandle) == 0)
    continue;      // nothing to read, try again.

  // read the IR distances and verify we have good data
  if (irProxy->data.ranges_count == 2) {
    frontIr = irProxy->data.ranges[0];
    rearIr  = irProxy->data.ranges[1];
  }

  // figure out how to drive
  runController(frontIr, rearIr,
                &desiredTranslation,&desiredRotation);

  playerc_position2d_set_cmd_vel(
      positionProxy, desiredTranslation,
      0, desiredRotation, 1);
}
```

Each time through the loop, we try to read the newest data from the robot. After a little sanity checking, we take the ranges reported by the IR sensors and feed them into a controller function. This controller does some magic processing (we'll talk about that later) and returns information on how we should drive the robot. Finally, we pass these driving commands back to the Player server and start it all over again.

All that's left now is to provide a runController function that maps from IR sensor readings to drive commands. The CoroBot driver accepts numbers in the range of –1.0 to +1.0 to tell how to drive the robot forward and backward: +1.0 means 100% power forward, –1.0 means 100% power in reverse, and 0.0 means stop. It accepts the same range for telling the robot how to turn: –1.0 means turn full power left, +1.0 means turn full power right, and 0.0 means drive straight ahead. Noting that the IR readings are provided in meters, we can use the following P-controller to drive our robot forward until we are 10cm away from a front obstacle. We even get a bonus for free—if we are closer than 10cm away, the robot will back up a bit until it is at the proper distance:

```
void runController(double frontIr, double rearIr,
                   double *translation,double *rotation)
{
  // convert our IR readings into drive commands
  *translation = (frontIr-0.1) * 3.0;
  *translation = *translation > 0.9? 0.9: *translation;
  *rotation    = 0.0;
}
```

And finally, good programmers always shut down their server connections when they are done:

```
void shutDownProxies()
{
  // close down proxies we have opened
  playerc_ir_unsubscribe(irProxy);
  playerc_ir_destroy(irProxy);
  playerc_position2d_unsubscribe(positionProxy);
  playerc_position2d_destroy(positionProxy);
```

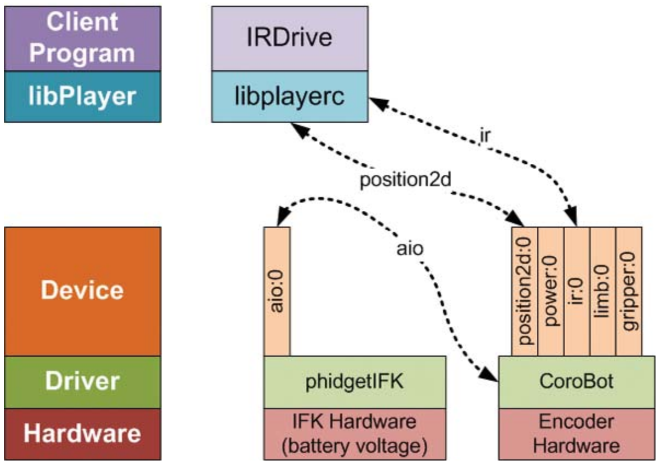


Figure 5. The Relationship between Several Devices and Interfaces When Using the Drive-by-IR Program

```
  playerc_client_disconnect(clientHandle);
  playerc_client_destroy(clientHandle);
}
```

Building on the design we showed earlier, we can see how our drive-by-IR program interacts with the Player infrastructure. The CoroBot configuration file loads the phidgetIFK driver, which exposes an aio:0 device. This device allows the CoroBot driver to read the robot's onboard infrared sensors. The CoroBot driver also exposes the position2d and IR interfaces, which the drive-by-IR program reads with the help of the libplayerc library (Figure 5).

The Player Project offers a lot of functionality that there just isn't room to get into in one article. This includes robot simulation, support for numerous commercial robots of many different prices and qualities, and support for a whole slew of readily available devices. Its plugin system even allows you to build your own drivers for new devices, either to support new hardware or to implement new experimental algorithms. Give it a try, and give your computer a chance to stretch its legs.■

Kevin Sikorski is a Robotics Architect at CoroWare Technologies where he designs, builds and programs mobile robots, and develops simulation software. In his spare time, he enjoys hiking in the Cascades and stargazing with his telescope.

Resources

The Player Project's Main Web Site: **playerstage.sourceforge.net**

Full Source Code for the Drive-by-IR Program: **ftp.linuxjournal.com/pub/lj/listings/issue188/10566.tgz**

CoroWare's CoroBot (a robot that provides drivers for working under the Player system): **www.corobot.net**

A List of Institutions That Use Player: **playerstage.sourceforge.net/wiki/PlayerUsers**

INTRODUCTION: A TYPICAL EMBEDDED SYSTEM

It's not always clear what separates ordinary Linux from embedded Linux. This article takes a look at the parts that make up a typical embedded system, starting with the bootloader and ending with end-user applications. **JOHAN THELIN**

The very first step in starting an embedded Linux system does not involve Linux at all. Instead, the processor is reset and starts executing code from a given location. This location contains a bootloader that initializes the device and sets up the basic necessities. When everything has been prepared, the Linux kernel is loaded and started. The kernel then initializes all the devices before mounting the filesystems and starting the userspace applications.

The Linux kernel and userspace are not merely a simple blob that is loaded and run. The kernel consists of a system-specific configuration and usually some tweaked initialization code. The userspace holds software libraries, data and several applications, all interacting to form a system. Each of these components is handpicked for the task and device in question in order to get a compact and well-performing system. Figure 1 shows the basic sequence of events.

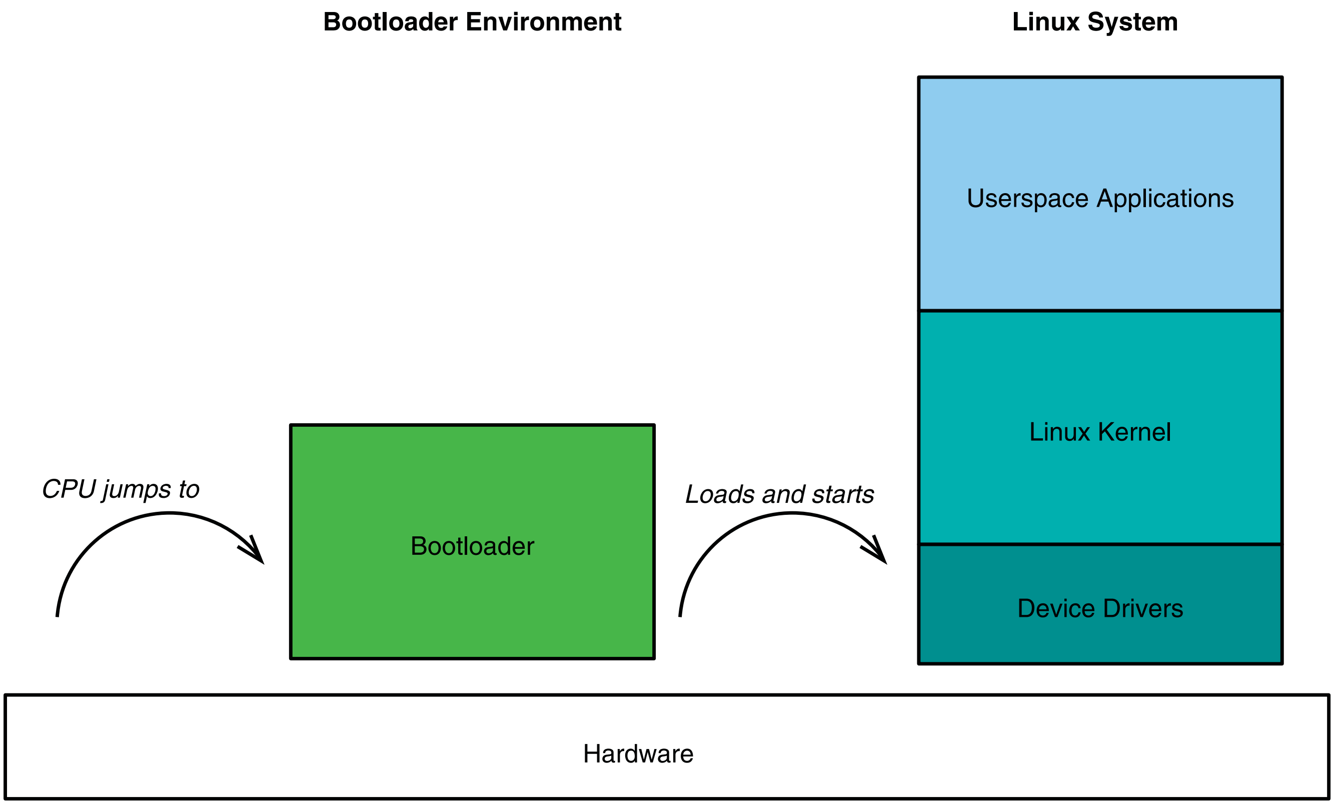


Figure 1. An Embedded Linux System Booting

THE BOOTLOADER

The bootloader is among the first pieces of software to run on the system. It basically has two tasks: initialize the system and load the kernel. The initialization can be to set up a UART to be used as a serial debug console and to configure the system's memory controller. For instance, if your system is using an SDRAM, you probably will have to set up the controller with regard to the memory's physical features. This includes page sizes, the number of columns, supported read and write widths, latencies and so on. In these days of portable devices, there is usually a plethora of settings for saving power when it comes to memory.

In addition to the basic tasks required by the bootloader, it is typical to provide some sort of command prompt where common low-level tasks can be carried out. These tasks usually include peeking and poking at random memory addresses, downloading and storing a Linux kernel image in Flash and setting bootargs for the kernel to interpret.

Examples of common bootloaders for embedded systems are Das U-Boot and RedBoot. Both support the basic tasks—meaning they can manage Flash, networking and serial communication. They also are available for several processor platforms, such as x86, ARM, PowerPC and more. You can add your own commands to both of them as well. This makes it possible to debug custom hardware without involving Linux, reducing the complexity of the system during the testing phase.

FILESYSTEMS

Choosing a filesystem for your embedded system depends on many factors. Do you need to be able to write to it? Do you value size or speed? Do you want to be able to replace the filesystem without replacing the kernel?

You also need to be aware of your storage medium's limitations. For instance, Flash has a limitation when it comes to how many times each cell can be written. To prolong the life of a Flash-based device, it's a good idea to use a filesystem that has been adapted for this purpose.

There are numerous filesystems from which to choose, but the following three are interesting as they show some important factors you should take into consideration:

- initramfs: a filesystem that is embedded into the kernel image. If your kernel is compressed, the initramfs filesystem is decompressed alongside the kernel. This gives the system a performance advantage. The filesystem is kept in RAM as the device operates and can be modified. However, all modifications are lost upon reboot.

- cramfs/squashfs: two compressed read-only filesystems. Both of these systems let you create a compressed image that you can mount at runtime. The filesystem can be replaced without touching the kernel.

- jffs2/ubifs: compressed filesystems tuned for Flash devices. These filesystems can be written to permanently, and they try to minimize the "wear and tear" of the Flash blocks by spreading write operations across the device.

Luckily, you do not have to pick one of these filesystems; instead, you can mix them—for instance, starting from an initramfs image with the most basic tools and then mounting a jffs2 Flash partition for storing user data. As Linux allows you to mount filesystems into any location in your directory tree, you can make this transparent to the applications using the filesystem.

THE LINUX KERNEL

The kernel itself is not very different from an ordinary desktop kernel. However, there are two major differences. First is the initialization, which often is system-specific. Second is that you probably know exactly what hardware will be used, so you can include all the drivers as part of the kernel and avoid the need for modules (unless you have proprietary drivers, of course).

When starting a desktop or a server system, the common scenario is that the kernel probes for hardware and loads the corresponding drivers as modules. This makes it possible to add hardware and still have a working system. You also can add drivers for new hardware without having to recompile the entire kernel. On an embedded system, you can optimize boot time by including all drivers in the kernel, but also by hard-coding parts of the available hardware, avoiding the need to probe for all devices and settings.

Returning to the standard PC, each machine starts and looks about the same during initialization. In the embedded case, each piece of hardware is unique, and you generally have to initialize the custom hardware. This means you actually will have to write code to set up your kernel for your board, which is usually easier than you think. For starters, lots of boards already are supported in the Linux kernel, and you usually can choose one of those as a starting point. Second, there are drivers for the most common peripherals, and again, you typically can find a good starting point, even when you have to create something of your own. So, the process is more or less to study the data sheets for your board and express what you learn to the kernel (something that can be both intimidating and daunting).

Embedded systems often are more limited than your average computer when it comes to system resources, so it is important to keep your kernel's footprint small. That, in turn, makes the kernel configuration stage important. By limiting configuration to a minimum, you can save those extra bytes needed to fit everything in.

THE C LIBRARY

The standard C library is one of the key components of any Linux system. It provides the userspace applications with a predefined interface, making them portable across different versions of the Linux kernel, as well as between different UNIX dialects. It basically acts as a bridge between the userspace applications and the kernel.

CROSS COMPILATION

One of the interesting aspects of embedded development is that you are likely to encounter new processor families. Most of you have x86 hardware at home; some might have a SPARC, 68k or a MIPS system lying around. With embedded systems, you are likely to run into ARM, SH, PowerPC or MIPS, among others.

The implication of this is that you must cross compile everything from your desktop build machine (your host machine) for your target device. The resulting binaries cannot be run directly on your desktop machine. You can do it using emulators such as QEMU that allow you to emulate common CPUs, but you will have to do some testing and probably some debugging on your target device.

Sometimes you can get a cross compiler from a vendor or distribution. You also can build your own. Building your own cross compiler used to be a real pain, but these days, you can use crosstool from Dan Kegel. Crosstool is a set of scripts and patches that allows you to build gcc and standard libraries for your platform of choice.

Crosstool's greatest feature is that you can (attempt to) build any combination of compiler and standard library. This makes it easy to try to build a toolchain for an existing device.

DISTRIBUTIONS and FRAMEWORKS

As fun as it is to roll your own, sometimes time does not permit it. A number of commercial players exist in the embedded Linux field, and many freely developed tools for building a complete embeddable environment also are available. The following list contains a few tools you might consider using:

- Buildroot: a set of Makefiles and patches for building a complete embeddable system. It generates everything from the cross compiler, the kernel and software libraries, and the userspace applications. The resulting system uses uClibc.

- Ångström distribution: another build framework for building embedded Linux systems. It also sports a package manager. This makes it possible to add and remove applications from the device directly, instead of having to build and download an entire system image or copy the application's files to the right locations manually.

- ScratchBox: a build framework for making embedded Linux application development easier. It has gained adoption through the Maemo development platform (the Nokia N7/8/9xx Internet tablets). It supports cross compiling entire distributions, can switch between glibc and uClibc and uses QEMU emulation of targets.

In all of those cases, it takes a bit of work to get the distributions running on a new system. As always with embedded systems, nothing is standardized, and size usually matters, so a bit of tweaking is more or less inevitable. However, having a framework for building a working system can be a real time-saver.

The version of the C library you usually find on your desktop machine is the GNU C library, glibc. It is a full-fledged C library, and, thus, a very large piece of software. For embedded systems, a few smaller alternatives are available: uClibc, newlib, dietlibc and others. These libraries try to implement the most commonly used interfaces in a minimalist way. This means they are mostly compatible with glibc, but not fully.

So, what does the C library contain that can be removed? uClibc, for example, skips the database library, limits the number of authentication methods that are supported, does not fully implement locale support, limits the math library mostly to doubles and leaves out some encryption functions. In addition, the kernel's structures are used directly whenever possible. Those and other things significantly reduce the size of the library.

What does this mean to you as an embedded developer? Most important, it means you can save quite a bit of memory, although you do so at the cost of compatibility. For instance, the decision to use the kernel's structures when applicable means the stat structure is different from the one used by glibc. You also have to limit yourself to flat password files and shared password files, unless you want to add a third-party library to handle authentication. More limitations exist, but generally speaking, most software compiles happily without patching.

BUSYBOX

When you have a bootloader, a kernel and a standard library, the next thing on the wish list usually is a command prompt. One of the big stars in the embedded Linux world is BusyBox. The idea behind the project is that most standard applications, such as ls, cd, mkdir, ping and so on, share a lot of code. Compiling each program separately means that code handling things such as command-line arguments is repeated in each application. BusyBox solves this problem by providing a single program, busybox, that can handle all the tasks provided by all the standard applications. By creating symbolic links for all the individual commands and pointing them to BusyBox, the user can still enter the expected commands and get the expected results.

As with everything else in the embedded world, tuning and tweaking is important. When it comes to BusyBox, you can handpick which commands to include, and for some commands, you even can handpick which command-line arguments are supported. If you don't need a particular command, simply don't include it in BusyBox. For instance, why keep ifconfig if you don't have a network?

When building a dynamically linked, default configured BusyBox on a desktop PC, it results in a binary that is just less than 700KB. This binary represents more than 200 commands and occupies more than 6MB of disk space on my Kubuntu-based system.

ADDING MORE

Once you have all the key components in place, you can start building and populating a root filesystem. This involves adding BusyBox, device files and expected directories. You also might want to add /etc/password and /etc/shadow, init scripts and so on. All this is necessary, but to get your device to do something, you need to add your own applications.

When developing for embedded devices, you might find yourself in a system completely without a graphical interface. This usually means implementing your functionality as some sort of server. As more and more devices are networked, a Web server often takes the place of a user interface. Because Apache is a large piece of software, a common solution is to use a lightweight server, such as Boa, for configuration and information.

If you happen to have a display, you likely will want to put graphics on it. An X sever might sound like a solution, but the two most common toolkits for building graphical interfaces, Qt and GTK+, also support using the framebuffer directly—again, saving both memory and computing resources.

And, that is what engineering embedded devices is all about: making the most with as little as possible. Being able to fit the coolest features into a small system means bringing an attractive device, at a good price, to consumers. Using embedded Linux to do that means you can get done more quickly, cheaply and be more hackable than with a closed-source system.■

Johan Thelin has worked with software development since 1995 and Qt since 2000. Having seen server-side enterprise software, desktop applications and Web solutions, he now works as a consultant focusing on embedded systems. He can be contacted at johan@thelins.se.

Resources

Crosstool: **www.kegel.com/crosstool**

Das U-Boot: **www.denx.de/wiki/U-Boot**

RedBoot: **www.sourceware.org/redboot**

uClibc: **www.uclibc.org**

newlib: **www.sourceware.org/newlib**

dietlibc: **www.fefe.de/dietlibc**

Buildroot: **buildroot.uclibc.org**

Ångström Distribution: **www.angstrom-distribution.org**

ScratchBox: **www.scratchbox.org**

BusyBox: **www.busybox.net**

Boa: **www.boa.org**

Qt: **qt.nokia.com**

GTK+: **www.gtk.org**

SMALL, EFFICIENT COMPUTERS WITH PRE-INSTALLED UBUNTU.

3677 Intel Core 2 Duo Mobile System
Range of Intel-Based Mainboards Available
Excellent for Mobile & Desktop Computing

GS-L08 Fanless Pico-ITX System
Ultra-Compact, Full-Featured Computer
Excellent for Industrial Applications

DISCOVER THE ADVANTAGE OF MINI-ITX.
Selecting a complete, dedicated platform from us is simple: Pre-configured systems perfect for both business & desktop use, Linux development services, and a wealth of online resources.

ubuntu solution provider

LOGIC SUPPLY
www.logicsupply.com

Controlling the Humidity with an Embedded Linux System

Using an inexpensive embedded Linux board and a few extra devices, you can control things like room humidity.

Jeffrey Ramsey

Charles Darwin, in his *Beagle Diary* that led to the book *Voyage of the Beagle*, wrote while in Peru, "On the hills near Lima, at a height but little greater, the ground is carpeted with moss, and beds of beautiful yellow lilies, called Amancaes. This indicates a very much greater degree of humidity, than at a corresponding height at Iquique." Like Darwin, I always have been conscious of humidity. For years, I've struggled with the humidity in my music room, as my Carlos Pina concert-grade classical guitar went out of tune frequently with wild swings in humidity. Pennsylvania winters are cold and dry, summers hot and humid, and this plays havoc on my classical guitar.

Commercially available humidifiers and dehumidifiers have humidity sensors that are far too coarse for certain applications. One such application is the humidity control for my music room. Being an embedded developer for my entire career, with a particular interest in embedded applications for Linux, I decided to build my own humidity controller for my music room. After a bit of research, I settled on a hardware architecture that includes a Cirrus EP9301 ARM9-based controller, several solid-state relays and a capacitive humidity/temperature sensor. Linux was my selection as the embedded OS, and with several Linux device drivers to control the relays and monitor the humidity and temperature, the basis of a humidity controller was born.

I decided to use the humidifying and dehumidifying capability of my retail humidifier and dehumidifier units. The humidity controller that I built switches power on and off to the humidifier and dehumidifier, essentially assuming the role of the humidity sensor. To finish off the humidity controller, I added a Web interface that allows me to monitor and control the system through any network-attached browser, such as Firefox.

Before I began developing the embedded humidity controller, I had to decide on the system-level requirements. Even though this was for personal use only, it's always good practice to do a bit of systems engineering on the front end of the design process. I decided on the following requirements:

- The humidity control system should control humidity with a minimum range of plus or minus 3.5% rH.
- Humidifier and dehumidifier control will be through switching of 120V AC and neutral power lines.
- Current humidity and temperature will be displayed through a browser interface.
- Configuration of the desired humidity setting will be done through a browser interface.
- All humidity and temperature settings will be stored persistently in an SNMP MIB.
- All software will operate in an embedded Linux environment.

Figure 1 shows the overall embedded hardware architecture of the humidity controller. The ARM9-based controller I selected is the TS-7200 from Technologic Systems. In addition to the controller board, I used a TS-RELAY8 peripheral board connected to the TS-7200's PC/104 bus. The daughter board contains eight SPDT relays. To house the system, I used a TS-ENC720 enclosure. Figure 2 shows the main board and peripheral board mounted on the back plate of the enclosure.

The capacitive humidity/temperature sensor is a Sensirion SHT11, which is controlled through a two-wire data/clock interface. The SHT11 control interface connects to two of the TS-7200's discrete I/O pins. Switching power on and off is accomplished with the single pole double throw (SPDT) relays on the peripheral board. I used a pair of relays for the humidifier and another pair for the dehumidifier. I used a pair as it

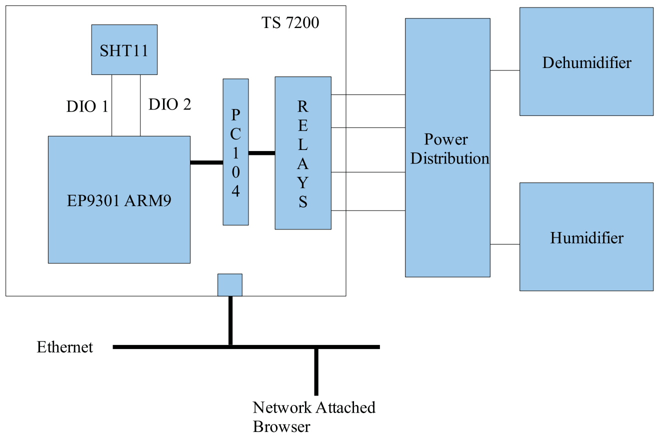


Figure 1. Hardware Architecture

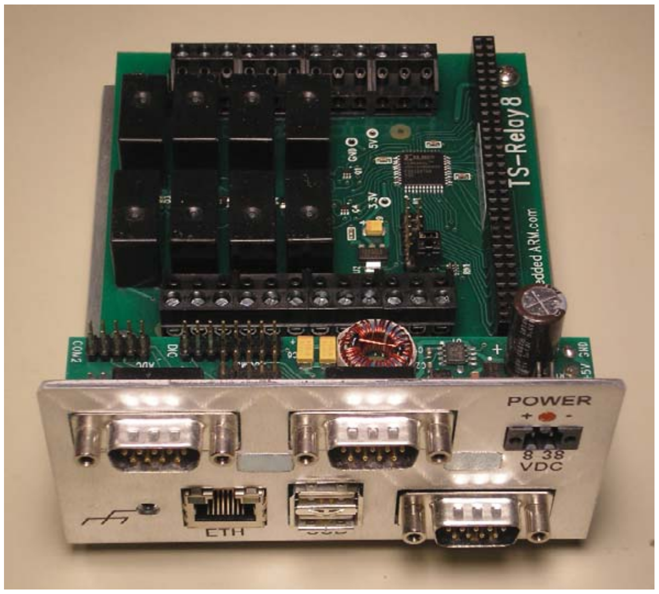


Figure 2. Hardware

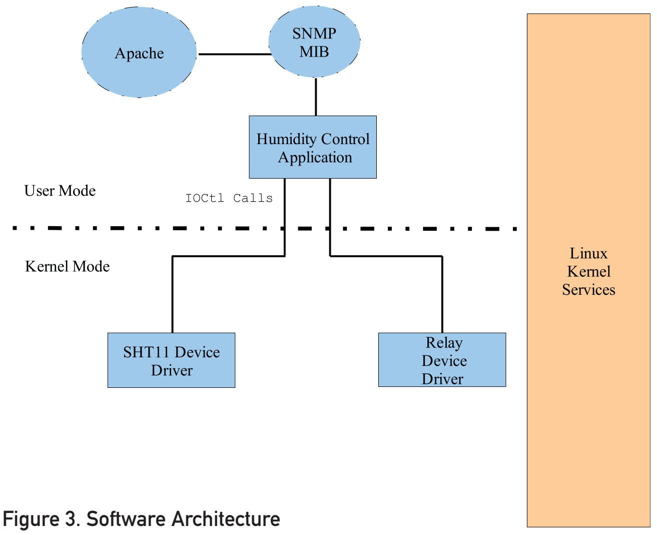


Figure 3. Software Architecture

Listing 1. Generate SHT11 Start Transmission Sequence

```
void writeSHT1xTransmissionStartSequence(void)
{
    writeSHT1xOne(DATA_SHT);
    writeSHT1xZero(SCK_SHT);
    udelay(2);
    writeSHT1xOne(SCK_SHT);
    udelay(2);
    writeSHT1xZero(DATA_SHT);
    udelay(2);
    writeSHT1xZero(SCK_SHT);
    udelay(2);
    udelay(2);
    udelay(2);
    writeSHT1xOne(SCK_SHT);
    udelay(2);
    writeSHT1xOne(DATA_SHT);
    udelay(2);
    writeSHT1xZero(SCK_SHT);
    udelay(2);
}
```

seemed much safer to switch both the 120V and neutral lines, rather than just the 120V.

The TS-7200 single-board computer (SBC) runs Linux on an ARM9-based processor. The system's software architecture is shown in Figure 3. Two Linux drivers are required: one to sense the humidity (and temperature, which came almost free) and the second to control the position of the relays. A user-mode application on top of the drivers periodically polls the humidity and temperature data, and controls the relay position depending on SNMP MIB configuration settings. The SNMP MIB is managed by the Linux snmpd dæmon. The SNMP MIB also serves as the basic bridge to an Apache custom module that exposes the MIB data to a Web browser for control and monitoring of the entire humidity control system. Each component of the humidity control system is described in more detail later in this article.

Linux Device Drivers

The two required Linux drivers, which I designed as loadable modules, are rather basic as far as Linux drivers go. They both are character devices with ioctl interfaces that provide access to the SHT11 sensor and control of the power relays. The SHT11 driver requires only two ioctl functions:

- SHT1X_IOC_READ_HUMIDITY: read the current SHT11 humidity.
- SHT1X_IOC_READ_TEMPERATURE: read the current SHT11 temperature.

With both the temperature and humidity, I have the option of calculating the dew point (even though the system is indoors, and the last thing I expect is dew to form on the components). The SHT11 driver reads humidity and temperature using a two-wire interface that is well defined in the Sensirion SHT11 data sheet. The clock has no real minimum frequency, but has a maximum frequency of 10MHz. I had no reason to run the clock at the maximum rate. In fact, the messages required to transfer the temperature and/or

Listing 2. Transmit Command Sequence

```
void writeSHT1xCommand(int iMode)
{
    unsigned char  ucBitToCheck;
    unsigned char  ucAckBit;

    driveDataLine(DATA_SHT);
    /* All 3 address bits always zero
     * so start with those */
    writeSHT1xZero(DATA_SHT);
    udelay(2);
    writeSHT1xOne(SCK_SHT);
    udelay(2);
    writeSHT1xZero(SCK_SHT);

    writeSHT1xZero(DATA_SHT);
    udelay(2);
    writeSHT1xOne(SCK_SHT);
    udelay(2);
    writeSHT1xZero(SCK_SHT);

    writeSHT1xZero(DATA_SHT);
    udelay(2);
    writeSHT1xOne(SCK_SHT);
    udelay(2);
    writeSHT1xZero(SCK_SHT);

    /* Now transmit the 5 command bits,
     * in the order of MSb to LSb */
    for (ucBitToCheck=0x10; ucBitToCheck != 0;)
    {
        if (iMode & ucBitToCheck)
            writeSHT1xOne(DATA_SHT);
        else
            writeSHT1xZero(DATA_SHT);
        udelay(2);
        writeSHT1xOne(SCK_SHT);
        udelay(2);
        writeSHT1xZero(SCK_SHT);
        ucBitToCheck >>= 1;
    }

    /* Now tri-state the data DIO so the
     * device can ACK the transfer */
    tristateDataLine(DATA_SHT);
    udelay(2);
    writeSHT1xOne(SCK_SHT);
    udelay(2);
    ucAckBit = readSHT1x(DATA_SHT);
    writeSHT1xZero(SCK_SHT);
    mdelay(250);
}
```

humidity data are so short, the clock rate could be anything within reason, so I decided to run the clock at 250KHz.

Accessing the SHT11 is relatively straightforward. A start and end sequence for each transfer is achieved using a prescribed combination of data and clock discrete I/O transitions. For example, in order to request the current humidity or temperature, a start of transmission sequence is issued that consists of the sequence of data and clock transitions as shown in Listing 1.

In Listing 1, note the use of udelay kernel calls. The timing requirements of the SHT11 two-wire access is satisfied using delays in the microseconds and, in some cases, milliseconds. This is most easily achieved using the kernel udelay call, and when millisecond delays are required, the mdelay call. I suppose there are some developers who shudder at the use of busy loops, but remember, this is a dedicated, embedded system. It does nothing but read humidity and check whether relays need to be switched on or off, and it repeats this forever.

After the start transmission sequence, the driver is free to write an 8-bit command sequence that identifies the operation to the humidity sensor, such as measure the humidity or temperature. A second procedure actually transmits the 8-bit command sequence and is shown in Listing 2.

Listing 2 not only demonstrates the bit-twiddling necessary to drive a two-wire interface solely with software, but it also reveals how the sensor acknowledges receipt of a valid command. The data DIO must be tri-stated (that is, not driven to either a 0 or a 1 by the ARM) in order for this two-wire interface to permit slave devices, such as the SHT11, to transmit back to the two-wire interface master—in this case, the SHT11 device driver in the ARM. In addition, note that the last line of code in the procedure will cause a 250-millisecond delay. This is because the SHT11 takes a good deal of time, relatively speaking, to measure either the temperature or humidity. The specification requires 210 milliseconds for the most accurate form of measurement, with a +–15% tolerance. This puts the worst-case delay at 241.5 milliseconds, which I increased to 250 milliseconds, just to be safe.

The third and final required piece of code necessary to read data from the SHT11 humidity sensor is shown in Listing 3. The Read Sensor Data procedure will read 16 bits of data from the sensor after it has measured either the humidity or the temperature. The SHT11 has the option of sending an 8-bit CRC at the end of the 16 bits of data, but I opted not to check the CRC, as it is unlikely the data ever will be corrupted due to environmental effects in my music room.

The procedures shown in Listings 1, 2 and 3 form the core of the SHT11 two-wire interface device driver code. When the driver receives an ioctl requesting the humidity, the three instructions shown in Listing 4 are all that is needed to read the current humidity from the sensor.

The second device driver controls the relays and switches the 120V AC and neutral lines to the humidifier and dehumidifier. The ioctl interface for the relay driver required

ServerBeach
by geeks, for geeks™

Linux servers from $69/mo

Order ANY dedicated hosting plan and get a new 16GB iPod Nano (video, FM, & more!)

Use the promo code: LJ2010 when ordering.

Call 1.800.741.9939 | www.serverbeach.com

Follow us on Twitter: @serverbeach

* Promotion awarded after 90 days of hosting with an account in current good standing. Offer expires March 31st, 2010.

Terms and conditions:
© 2009 ServerBeach, a PEER 1 Company. Not responsible for errors or omissions in typography or photography.
This is a limited time offer and is subject to change without notice. Call for details.

Listing 3. Read Sensor Data

```
unsigned int readSHT1xData(void)
{
    int           iLoop;
    unsigned int  uiBitRead;
    unsigned int  uiMSB=0;
    unsigned int  uiLSB=0;
    unsigned int  uiRetValue;

    /* Read MSB from SHT1x */
    for (iLoop = 0; iLoop < 8; iLoop++)
    {
        uiMSB <<= 1;
        writeSHT1xOne(SCK_SHT);
        uiBitRead = readSHT1x(DATA_SHT);
        udelay(2);
        writeSHT1xZero(SCK_SHT);
        udelay(2);
        if (uiBitRead)
            uiMSB |= 1;
    }
    /* Acknowledge sequence; must drive data
     * line as it is tri-stated at this point */
    driveDataLine(DATA_SHT);
    writeSHT1xZero(DATA_SHT);
    udelay(2);
    writeSHT1xOne(SCK_SHT);
    udelay(2);
    writeSHT1xZero(SCK_SHT);
    tristateDataLine(DATA_SHT);
    udelay(2);
    /* Read LSB from SHT1x */
    for (iLoop = 0; iLoop < 8; iLoop++)
    {
        uiLSB <<= 1;
        writeSHT1xOne(SCK_SHT);
        uiBitRead = readSHT1x(DATA_SHT);
        udelay(2);
        writeSHT1xZero(SCK_SHT);
        udelay(2);
        if (uiBitRead)
            uiLSB |= 1;
    }
    /* Don't acknowledge last byte so the device
     * doesn't transmit the 8-bit CRC as it isn't
     * really necessary for this application */
    uiRetValue = u8tou16(uiMSB, uiLSB);

    return(uiRetValue);
}
```

Listing 4. Read Humidity Sequence

```
writeSHT1xTransmissionStartSequence();
    writeSHT1xCommand(SHT1x_MEASURE_HUMIDITY);
    uiHumidity = readSHT1xData();
```

the following ioctl functions:

- RELAY8_IOC_READ_RELAYS: read the current relay settings.
- RELAY8_IOC_WRITE_RELAYS: set the relays to the supplied state.

Reading the relay settings is used to ensure that the relays are in the desired position. The relay hardware actually includes eight relays, and all eight relay values are written in one shot. The data register used to control and report the relay positions consists of one 8-bit register. This register either is read to report the current relay settings or written to change the relay settings. Unlike the SHT11 driver, the relay driver can affect a change in a relay state with one writeb Linux driver C instruction. Listing 5 shows the relay read and write procedures, along with an excerpt from the ioctl processing that differentiates between read and write. It doesn't get much simpler than this!

Listing 5. Read/Write Relays

```
unsigned char readRelay8(int iRelay8Address)
{
    /* Read Relay8 register and return the value */
    return(readb(iRelay8Address));
}

void writeRelay8(int iRelay8Address, unsigned char ucValues)
{
    /* Write Relay8 register with the values */
    writeb(ucValues, iRelay8Address);
}

// Excerpt from ioctl function:
    switch(cmd) {

    case RELAY8_IOC_READ_RELAYS:
        /* Read Relay8 relay values */
        ucRelayValues = readRelay8(relay8_base + RELAY8_CONTROL);
        if (copy_to_user((typeof(relay8_relayValues)) arg,
                        &ucRelayValues,
                        sizeof(relay8_relayValues))) {
            ret = -EFAULT;
        }
        break;

    case RELAY8_IOC_WRITE_RELAYS:
        /* Write Relay8 relay values */
        writeRelay8(relay8_base + RELAY8_CONTROL,
                *(typeof(relay8_relayValues)) arg);
        break;

    default:
        ret = -ENOTTY;
    }
```

User-Mode Application

I wrote a user-mode application that periodically polls the SHT11 device driver for the current humidity and temperature using the ioctl SHT1X_IOC_READ_HUMIDITY and SHT1X_IOC_READ_TEMPERATURE, respectively. Depending on the desired humidity setting, the application determines whether the current humidity is either too high or too low, taking into account the tolerance of plus or minus 3.5% rH. If an actionable event is determined, the specific relays are turned either on or off using the relay device driver RELAY8_IOC_WRITE_RELAYS ioctl function. For example, when the user-mode application reads the humidity and determines that the humidifier must be turned on, it issues an ioctl RELAY8_IOC_WRITE_RELAYS function to switch on both relays that are dedicated to the 120V A/C and neutral lines of the humidifier. At the same time, the application also ensures that the two relays associated with the 120V A/C and neutral lines of the dehumidifier are switched off. Relay control can be one of three options: 1) the humidifier is turned on, and the dehumidifier is turned off; 2) the dehumidifier is turned on, and the humidifier is turned off; or 3) both the humidifier and dehumidifier are turned off. The application never turns both the humidifier and dehumidifier on at the same time. The application is loaded at Linux boot time and, like most embedded applications, runs perpetually.

Along with controlling the humidifier and dehumidifier relays, the application accumulates and saves statistics. In this control system, the actual data that is acted upon is required to be persistent—that is, the humidity data must be saved somewhere for later use. The user-mode application is responsible for saving the data for later use by a browser, and it does so with the use of the SNMP (Simple Network Management Protocol) support provided by the net-snmp Linux package.

SNMP is a standard set of protocols and policies for managing networks and devices. The net-snmp implementation of SNMP consists of an agent, which runs as a Linux dæmon snmpd, and a database called a Management Information Base, or MIB. A MIB is structured as a tree, with branches grouping together similar items. I extended the standard Linux MIB that is shipped with the net-snmp package and added a new branch off of the “enterprises” node, which includes all the humidity controller items that I need (Figure 4). The snmpd agent acts on the MIB at the request of SNMP clients—that is, the agent reads/writes data from/to the MIB on behalf of client get and set requests. In this architecture, there are two clients: the user-mode application and the Web browser.

In order to adapt SNMP to any application, a MIB must be defined in a standard MIB ASN.1 format. I defined a MIB for my humidity controller and called it HUMIDITYCONTROLLER-MIB, which gets loaded when the snmpd dæmon runs during the Linux boot process. The MIB contains data items that are represented by object identifiers, or OIDs. An example of an OID definition from my MIB for the humidity controller targetHumidity variable is shown below:

```
targetHumidity OBJECT-TYPE
    SYNTAX      Integer32(0..2147483647)
    MAX-ACCESS  read-write
    STATUS           current
    DESCRIPTION
    "Target humidity."
    ::= { humidityEntry 3 }
```

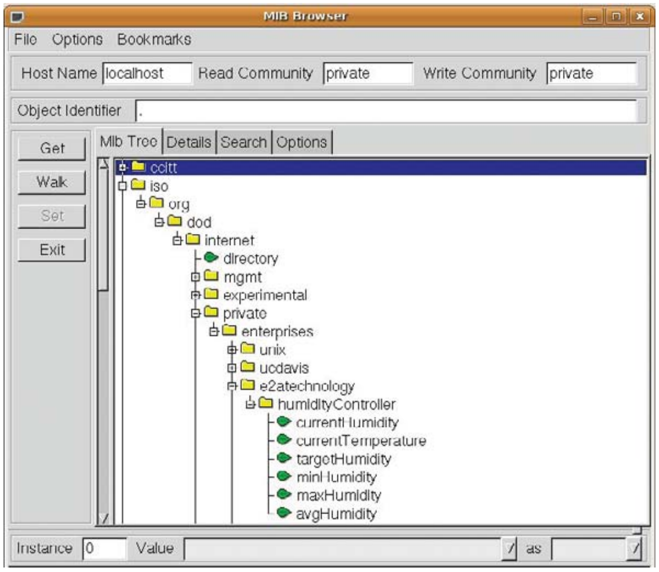
The previous ANS.1 MIB definition phrase generates an OID with the value .1.3.6.1.4.1.2200.2.3. This rather cryptic-looking sequence of numbers is a scheme used to identify a leaf in the MIB tree. The branch that I added to the enterprises node is identified by the integer 2200. Under the 2200 node is the node identified by a 2, which contains all of the overall humidity controller items that the system needs. The leaf node identified by a 3 is the targetHumidity.

The Linux SNMP package contains a very useful tool called mib2c. mib2c takes a MIB definition, such as HUMIDITYCONTROLLER-MIB, and generates C code that can be used to extend the standard Linux snmpd agent. Several options exist when generating code with mib2c. I used the more general option for generating C code from a MIB with the mib2c.scalar.conf configuration, which causes code to be generated for general-purpose scalar OIDs, as opposed to table-based OIDs. The generated C code is used by the snmpd dæmon. Listing 6 is a distilled example of the generated C code from mib2c for the targetHumidity OID that shows the code framework needed to support the SNMP get

Listing 6. mib2c-Generated C Code

```
netsnmp_register_scalar(
    netsnmp_create_handler_registration(
        "targetHumidity",
        handle_targetHumidity,
        targetHumidity_oid,
        OID_LENGTH(targetHumidity_oid),
        HANDLER_CAN_RWRITE));


int
handle_targetHumidity(netsnmp_mib_handler           *handler,
                      netsnmp_handler_registration  *reginfo,
                      netsnmp_agent_request_info    *reqinfo,
                      netsnmp_request_info          *requests)
{
    switch (reqinfo->mode) {

    case MODE_GET:
        break;

    case MODE_SET_RESERVE1:
        break;

    case MODE_SET_COMMIT:
        break;
    }

    return SNMP_ERR_NOERROR;
}
```

To finish off the humidity controller, I added a Web page interface that includes a recipe that uses a tad of HTML, a smattering of JavaScript and a pinch of AJAX with server-side scripting to create an end-user browser interface.



Host Name localhost Read Community private Write Community private

Object Identifier .

Instance 0 Value

Figure 4. mbrowse Screenshot

(MODE_GET) and SNMP put (MODE_SET_RESERVE1 and MODE_SET_COMMIT) operations.

The code shown in Listing 6 makes reference to the generated callback procedure, handle_targetHumidity, which is supplied in skeletal form only by mib2c. Not much code is needed in order to support scalar OIDs, which the humidity controller uses exclusively. Anytime a specific OID, in this case the targetHumidity OID .1.3.6.1.4.1.2200.2.3, has an operation performed, the snmpd dæmon will invoke this callback procedure with an indication of the requested operation being performed on the OID.

I rebuilt the snmpd dæmon so that the newly created humidity controller MIB structure and generated framework code could be supported. Before rebuilding the snmpd dæmon, the new MIB must be configured into the build environment. This is accomplished easily with the following command:

```
$ ./configure --with-mib-modules="humidityController"
```

Once configured, the entire net-snmp package was rebuilt with the make command. Once the snmpd dæmon was rebuilt, I tested the new MIB structure by using the net-snmp command-line interface utilities snmpset and snmpget. For example, in order to set the targetHumidity OID to 50% rH, the following command can be issued:

```
$ snmpset -Ovqe -v 1 -c private localhost targetHumidity.0 i 50
```

Note the use of relative, symbolic OIDs in the snmpset command. The actual OID .1.3.6.1.4.1.2200.2.3 could be used as well, because it's statically defined and should never change. But, I prefer symbolic references where possible, as it helps in readability. The -Ovqe switch controls the output format that results from the snmpset. Although I built the net-snmp package to support all three major versions of SNMP (1, 2 and 3), I really needed only basic version 1 support, which is why the -v 1 switch appears. The SNMP community string is indicated by the -c private switch and appears in set operations because only private communities are permitted to set OID values (this is a one-time option when the snmpd dæmon is configured).

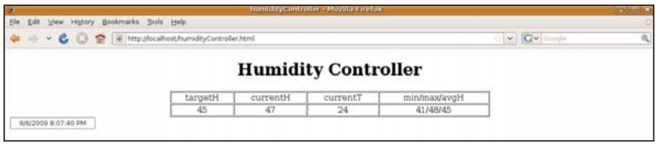
The humidity controller MIB can be viewed with a tool included in net-snmp called mbrowse. mbrowse is a GUI that bolts onto the system MIB structure and permits manipulation of specific OIDs. Figure 4 shows a screenshot of mbrowse and the humidity controller MIB tree branch.

Once the snmpd dæmon was complete with support for the newly added humidity controller OIDs, I was able to complete the user-mode application code. Listing 7 contains the complete user-mode application, and it is too long to print here, but it is available on the *LJ* FTP site (see Resources). It is very typical of an embedded application, as it perpetually reads data and then takes actions on the data. Note the use of snmpget and snmpset. The net-snmp package does include APIs for both C and Perl, but I decided it was simpler to leverage the existing snmpget and snmpset utilities.

To finish off the humidity controller, I added a Web page interface that includes a recipe that uses a tad of HTML, a smattering of JavaScript and a pinch of AJAX

Listing 8. Perl Script setTargetHumidity

```
use CGI;
$query = new CGI;
$targetH = $query->param('targetH');
$SNMP_SET_CMD = "snmpset -v 1 -c private";
$SNMP_TARGET = "localhost";
$SNMP_TARGETHUM_OID = "targetHumidity .0";
$SNMP_TYPE = "i";

chomp($retVal = `${SNMP_SET_CMD} ${SNMP_TARGET}
      ➥$SNMP_TARGETHUM_OID $SNMP_TYPE $targetH`);
```


humidityController - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost/humidityController.html
Google

Humidity Controller

targetH	currentH	currentT	min/max/avgH
45	47	24	41/48/45

9/6/2009 8:07:40 PM

Figure 5. Humidity Controller and Firefox

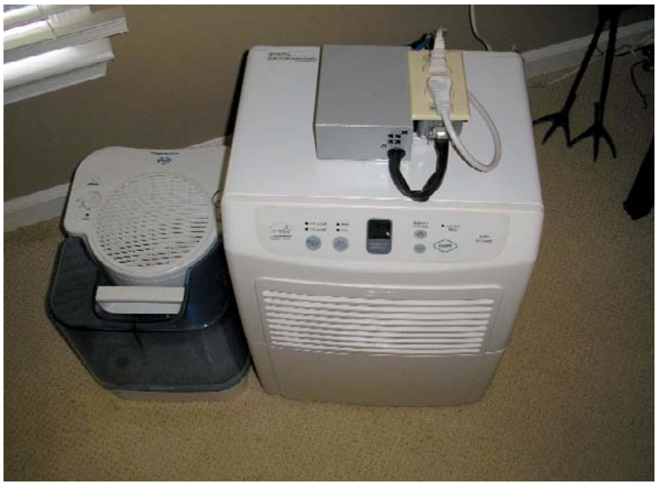


Figure 6. Completed Humidity Controller with Humidifier and Dehumidifier Connected

Figure 7. Carlos Pina Classical Guitar

with server-side scripting to create an end-user browser interface. The humidity controller in a Firefox browser looks like what is shown in Figure 5. The targetHumidity (targetH) cell in the table has a JavaScript function associated such that editing is possible, and when a new value is entered, it is POSTed to Apache. Apache will invoke a Perl script to set the target humidity in the SNMP MIB. Listing 8 is an excerpt from the Perl code that shows the SNMP actions. The other cells are read-only and are refreshed periodically with values from the SNMP MIB with the help of a second Perl script, humidityController.cgi. This second Perl script pushes out only the data necessary to generate the table of values shown in Figure 5.

The humidity controller (Figure 6) has been keeping my music room within a humidity range that makes my Carlos Pina classical guitar quite happy (Figure 7). The work involved to build the system was a real pleasure. But the best part is sitting down to play the opening arpeggios in Bach's *Prelude* and hearing the notes ring true without retuning my guitar. It not only makes me smile, but I think it would make Bach smile as well.■

Jeffrey Ramsey has been an embedded developer his entire career, and when not pouring through Linux kernel and driver source code, he can be found plucking a guitar. Jeffrey can be contacted at jeffreyramsey@e2atechnology.com.

Resources

Listing 7 (the Complete User-Mode Application):
ftp.linuxjournal.com/pub/lj/listings/issue188/10534.tgz

TS-7200 Main Board (from Technologic Systems):
www.embeddedarm.com/products/board-detail.php?product=TS-7200

TS-RELAY8 Daughter Card (from Technologic Systems):
www.embeddedarm.com/products/board-detail.php?product=TS-RELAY8

TS-ENC720 Enclosure (from Technologic Systems):
www.embeddedarm.com/products/board-detail.php?product=TS-ENC720

Reducing Boot Time in Embedded Linux Systems

Using some reasonably simple techniques, you may be able to reduce dramatically the boot time of your embedded Linux system.

Christopher Hallinan

It is no secret that Linux has won the race in the embedded device marketplace. Tremendous advantages in Linux have broken almost every barrier to entry for using Linux on embedded systems across a wide variety of processor architectures. Today's developers are not asking, "Should I use Linux for my embedded system?", but instead are asking questions like, "How can I get more performance out of my embedded Linux design?" Reducing boot time has become one of the more interesting discussions taking place in the embedded Linux community.

As it turns out, it is relatively easy to save substantial time on system boot. Without a significant expenditure of engineering resources, savings of more than 80% are possible with certain system configurations. Of course, there is a point of diminishing returns. The graph of engineering effort against boot time would rapidly approach infinite effort as time reduced into the milliseconds and lower.

Fast Boot Requires Definition

Before you can measure boot time, you must define what it means. (I introduce measurement techniques later in this article.) Most often, your customers or end users provide, or at least influence, the definition. The type of product you design certainly impacts your definition. Most systems that appear to boot very quickly actually are just providing early feedback to users in the form of graphical banners, audible feedback, animation or some combination thereof. You as the system designer must specify what it means for your embedded device to be booted and exactly what the user experience will be during power-on.

Do you define boot time as the time from power-on to playing your favorite music? Or, maybe you design big iron, and boot time eats into your annual "five-nines" reliability budget. A cellular radio node controller that takes two minutes to boot eats up almost half your annual downtime budget! Yet, many systems we perceive as fast boot systems are not actually booting from power-on. Consider a popular cell-phone design, such as the BlackBerry Curve. The only time these systems perform a full boot is when the battery is removed and replaced. Power "on" is actually a resume from a low-power system state that largely preserves its current operational status.

It Starts with the Hardware

Although it may seem trivial to mention, sound hardware design is a fundamental component of a fast boot system. Many aspects of hardware design can have a marked influence on 1) how quickly your first bits of code get to execute and 2) how quickly that code can be read out of a nonvolatile storage device during initial boot. Pay particular attention to power-on reset circuitry and initial hardware strapping, which provides default timings for external buses and chip selects on certain processors. It is not uncommon to find "conservative" values being employed here that often can be improved upon.

Your overall hardware architecture will set the stage for what performance you will be able to achieve. Choice of processor, clock speed, choice of nonvolatile storage used for boot images and many other factors will influence how fast your design can fetch and execute its startup image (usually a bootloader) and then go on to load and execute an operating system. Your hardware choices at design time must be carefully considered if single-digit boot times are part of your product requirements.

Typical Boot Sequence

To understand where time is being spent, it helps to visualize the boot sequence of a typical embedded Linux system. Figure 1 shows the basic sequence.

Upon power-on, the hardware needs time for voltages (and often clocks) to stabilize and for reset to be released. The first code executed upon release of reset depends on the hardware architecture and processor, but often it is your bootloader running from nonvolatile memory, such as NOR Flash. A small section of code performs some low-level initialization that includes the memory controller and typically copies itself into DRAM for further execution. This copy operation can consume a significant portion of boot time. It is easy to see that keeping the bootloader small and simple (the KISS principle) will help

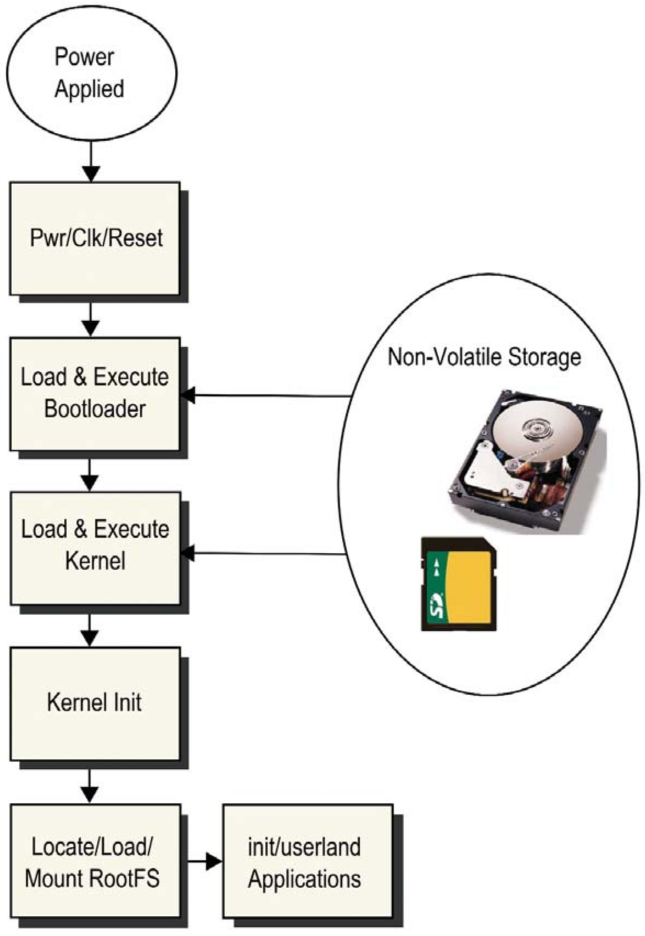


Figure 1. Typical Boot Sequence of Events

keep boot time to a minimum. The bootloader's primary responsibility after hardware initialization is to locate, load and pass control to your Linux kernel. Once the kernel has completed its own initialization, it must locate and mount a root filesystem. Your root filesystem will contain a set of initialization scripts as well as your own applications. There are numerous opportunities for optimization in all of these steps, as I explain below.

Bootloader Considerations

Virtually every embedded system has some type of bootloader, and there are many bootloaders from which to choose. Some of the more popular include U-Boot for PowerPC, MIPS and ARM processors, and RedBoot, which is frequently found on ARM processors. Most popular bootloaders today contain far more functionality than actually is required for the task of initializing a system. Indeed, they have become valuable tools used by developers during initial board bring-up and system development.

Modern bootloaders are packed with features, such as Flash erase and program utilities, memory management utilities, network capabilities for loading images and for self-configuring (DHCP and BOOTP for example), drivers for PCI, IDE, USB and support for various partition types and filesystems. Some even have scripting language support useful for development, manufacturing (production test, image load and so on) and system upgrade.

These features make bootloaders an indispensable tool during product development. However, the size of bootloader images has significant impact on boot time. Bootloaders are stored in nonvolatile storage media, most commonly NOR Flash. However, embedded systems rarely execute code directly from Flash, mostly because it is far too slow. Read times of DRAM are orders of magnitude faster than read times from Flash.

The first job of the bootloader is to load itself into DRAM and continue execution from there. Consider the operating environment of early boot code. There is no stack, no C context (meaning all this code is written in the processor's native assembly language), and quite often, the processor's instruction and data caches are not yet enabled. This means the size of the bootloader image, which needs to be copied into RAM, will have a major impact on startup time.

The quickest path to performance improvement in your bootloader is to keep it small. Remove features that are unnecessary in a production environment. Some bootloaders, such as U-Boot, make it easy to do this. Its features are driven by a board-specific configuration file that contains directives to enable or disable features. Your requirements will ultimately rule, but prudent trimming of all but the most essential features will yield significant savings in boot time.

Uncompressed Kernel

When you build a Linux kernel image, it is virtually always compressed as one of the final build steps. It is the responsibility of your bootloader (or a small bootstrap decompression loader that is appended to your kernel image) to decompress the kernel image and place it into system memory. One of the single largest easy gains you can achieve is to remove the decompression stage. Some architecture/bootloader combinations don't bother to enable caches, making decompression take much longer. It is not uncommon to find systems that take several seconds to perform the decompression and relocation stage. Using an uncompressed kernel can significantly reduce your system boot time.

Eliminate initrd/initramfs

Linux distributions use initrd/initramfs (hereinafter referred to as simply initramfs) primarily as a tool to enable a generic kernel to be used across a huge variety of system configurations. It is the job of the initramfs to provide the necessary device drivers to enable the devices that are required to complete system boot. Because embedded systems often are restricted to limited well-known configurations, they usually can eliminate initramfs with a corresponding reduction in boot time. Furthermore, removing support for initramfs in the kernel results in a smaller kernel (and, thus, faster boot.)

Smaller Kernels Boot Faster

If you compile a kernel with a "default" configuration, it often contains a vast number of features your system may not need. You may be surprised to discover how many features are enabled by default that your embedded system does not need. Spend some quality time with your favorite kernel configuration editor (menuconfig, gconfig and so on), and go through each and every kernel configuration parameter. Evaluate whether your system requirements can do without it. Yes, it may take you the better part of a day (or even more if you add in some research time), but your savings in boot time reduction can be substantial. Some examples of features found in many default kernel configs include IPv6, RAID, support for many filesystems you may not need, extended partition support and many more. There also may be numerous device drivers compiled into the kernel for devices that are not present in your system. They are harmless, but each driver runs initialization code, including registration functions, and some spend lengthy milliseconds (or more) in device probe routines for devices that are not present—precious boot time can be spent probing for non-existent devices.

Calibration Routine

You may have seen the interesting "BogoMIPS" message plastered on your screen or terminal during boot. Linux calibrates its internal software timing loops to your processor system clock on each boot, arriving at a constant value used by the loops_per_jiffy variable (lpj). Although the technique varies across different architectures, this can be a time-consuming routine. It is easy to bypass this dynamic calibration routine by "hard-coding" the time constant calculated by this routine. This is quite easily passed to the kernel through the kernel command line. Simply add `lpj=xxxxx` to your kernel command line, where xxxxx is the lpj value printed to your boot log during boot. This is what the boot message looks like on the Intel Atom-based Netbook on which this article is being drafted:

```
Calibrating delay using timer specific routine..
    3194.85 BogoMIPS (lpj=6389712)
```

From this information, simply add the string `lpj=6389712` to your kernel command line. This will bypass the often lengthy calibration routine and instead use the fixed value for loops_per_jiffy.

Filesystem Selection

One of the keys to achieving single-digit boot times is your choice of root filesystems. Some filesystems designed for Flash use, for example the ubiquitous JFFS2, can get into a state that requires a significant and noticeable time delay while the kernel reads the sequential journal entries and reconstructs the files and directories on the filesystem. Consider using a small, compact and fast root filesystem for your initial system boot, and then mount a more general-purpose filesystem later in the initialization sequence.

CRAMFS is a read-only, compressed filesystem that is perfectly suited for this purpose. Configure a preliminary root filesystem using CRAMFS, which contains all the executables and libraries you need to get your system into a preliminary operational state. Later, while other less critical tasks are being executed, you can mount a writable JFFS2 partition when time is not so critical. Also consider the liberal use of tmpfs for volatile data such as /tmp, /var and others. Tmpfs is fast and efficient, and dynamically resizes itself to meet storage requirements. Remember, the contents of all tmpfs filesystems are lost on power-down, so if there are any files (log files, configuration data and so on) that must be saved, it will be up to your application to save this data periodically to nonvolatile storage.

Udev Considerations

Udev has become an efficient and powerful system configuration tool. Its primary role is to create device nodes for devices that the kernel discovers. Virtually every modern Linux distribution uses udev coupled with a set of rules for device naming. Udev also has the capability to run external programs in response to device detection. The most common example of this is to run modprobe to install a device driver upon device detection. For example, if you plug an SD card into an appropriate socket, a properly configured udev-based Linux system will perform all the actions required to enable the device. This includes loading device drivers and creating the device nodes associated with the device and driver.

This powerful and flexible scheme has one drawback. Although udev itself is fast and efficient, some of the external programs it runs may require significant time to complete.

When a Linux system is booted and reaches userland, udev basically "plays back" all the device notification events generated by the kernel and performs the required actions (primarily device node creation and module loading). This can take a significant amount of time. One solution to this problem is to configure your Linux system with statically generated device nodes for critical system devices (those that you need to be operational immediately) and defer the running of udev until your fast-path boot chores are complete. For each device you need to have immediately available at boot time, create a static device node in /dev as part of your root filesystem. Later, when udev takes over, your udev startup script can merge these static devices with those devices that udev creates dynamically.

Measuring Boot Time

Several tools are available to help you identify the long paths in your system boot. They vary in complexity and ease of use, but most can be mastered quickly. The simplest tool (and perhaps a good starting point) is to configure your kernel to add timestamps to the kernel messages that are displayed on boot. Select CONFIG_PRINTK_TIME in the Kernel Hacking section of your kernel configuration to enable this feature. This lets you see at a glance where significant time is being spent during the actual kernel boot sequence.

Another easy tool to use is to enable printout of each kernel initcall. An initcall is a special type of kernel function call specifically related to subsystem initialization. This is accomplished by adding the single parameter initcall_debug to your kernel command line. When enabled, the kernel will display a line that lists the kernel virtual address of each initcall together with return data and call duration. While you must "decode" the kernel virtual address into its symbolic function name, this data is readily available in the System.map file in your Linux kernel source tree. If you have CONFIG_KALLSYMS enabled (found under General setup→Configure standard kernel features for small systems), the initcall line will be decoded for you. A sample of the output from the system boot log with initcall_debug enabled is displayed in Listing 1.

Using initcall_debug in this scenario reveals that almost nine seconds could be saved by eliminating or deferring ide

Advertiser Index

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

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

Advertiser	Page #	Advertiser	Page #
1&1 Internet, Inc. www.oneandone.com	1	Natuba Natuba.com	79
Aberdeen, LLC www.aberdeeninc.com	13	Platform Computing www.platform.com	17, 78
Archie McPhee www.mcphee.com	79	Polywell Computers, Inc. www.polywell.com	5, 79
ASA Computers, Inc. www.asacomputers.com	33	Rackspace Managed Hosting www.rackspace.com	C3
Cari.net www.cari.net	39	Saint Arnold Brewing Company www.saintarnold.com	78
Digi-Key Corporation www.digi-key.com	78	ServerBeach www.serverbeach.com	57
Emac, Inc. www.emacinc.com	61	Servers Direct www.serversdirect.com	9
EmperorLinux www.emperorlinux.com	71	Silicon Mechanics www.siliconmechanics.com	27, 31
Gecad Technologies/Axigen www.axigen.com	79	StrayTats www.straytats.com	78
Genstor Systems, Inc. www.genstor.com	29	Technologic Systems www.embeddedx86.com	37
Gutsy Geeks www.gutsygeeks.com	78	Tuxera Ltd. tuxera.com	19
iXsystems, Inc. www.ixsystems.com	7	Ubiquiti Networks, Inc. www.ubnt.com	C2
Logic Supply, Inc. www.logicsupply.com	53	Utilikilts www.utilikilts.com	78
Lullabot www.lullabot.com	21, 23	ZaReason www.zareason.com	77
Microway, Inc. www.microway.com	C4, 3		

ATTENTION ADVERTISERS

March 2010 Issue #191 Deadlines
Space Close: December 28; **Material Close:** January 5

Theme: System Administration

BONUS DISTRIBUTIONS:
Annual Linux Users Group Promotion

Call **Joseph Krack** to reserve your space
+1-713-344-1956 ext. 118, e-mail joseph@linuxjournal.com

Listing 1. initcalls Taking More than Nine Milliseconds to Complete

```
root@8548cds:~# dmesg | grep initcall | egrep '[0-9][0-9] msecs'
initcall pty_init+0x0/0x43c returned 0 after 57 msecs
initcall serial8250_init+0x0/0x138 returned 0 after 20 msecs
initcall gfar_init+0x0/0x58 returned 0 after 60 msecs
initcall cp_init+0x0/0x34 returned 0 after 16 msecs
initcall ide_scan_pcibus+0x0/0x14c returned 0 after 4246 msecs
initcall of_flash_init+0x0/0x34 returned 0 after 43 msecs
initcall uhci_hcd_init+0x0/0x104 returned 0 after 445 msecs
initcall ip_auto_config+0x0/0xefc returned 0 after 4597 msecs
```

Listing 2. Using kd

```
$ ./linux/scripts/kd -n 10 kft_data.sym
Function                      Count Time     Average   Local
----------------------------- ----- -------- --------  --------
__schedule                     5208 22050824     4234  22046510
schedule                       1921 10828704     5637 -10478620
setup_arch                        1  6021110  6021110        29
tsc_init                          1  6021081  6021081        79
set_cyc2ns_scale                  1  6021002  6021002   6021002
kobject_uevent                  389  1659254     4265    813013
mem_init                          2  1223745   611872    111906
wait_for_completion             395  1192559     3019     14685
free_all_bootmem                  1  1109561  1109561        53
free_all_bootmem_core             1  1109508  1109508     74651
```

and IP auto configuration!

Using KFT for Boot Time Measurements

One of the more powerful tools for boot time measurement is Kernel Function Trace. KFT instructs the compiler to generate instrumentation for virtually all kernel function calls. When enabled and triggered, data is logged for each function call entry and exit, which allows you to identify functions that consume large amounts of time. To use this tool, you will need to apply the KFT patch to your kernel. Detailed instructions and links to KFT kernel patches can be found at **elinux.org/Kernel_Function_Trace**.

For each function call not specifically filtered out by your trace configuration, a line is generated that contains a timestamp at function entry, the function address, the address of the caller, process ID and a delta. The raw data is accessed by reading /proc/kft_data after the run has completed.

Several tools are available to post-process the raw data. The addr2sym converts the kernel virtual address in the raw data to symbolic addresses. It is simple to use:

```
# addr2sym <kft_data.raw -m System.map >kft_data.sym
```

Above, kft_data.raw is the raw data copied from /proc. System.map is produced by the kernel build and can be found in the top-level kernel directory, and of course, kft_data.sym is the output file.

The KFT dump (kd) utility can be found in the Linux kernel scripts directory after the KFT patch has been applied. Running kd on the raw data produces a statistical summary of the functions called. You can use kd to display the most time-consuming functions or functions with time greater than that specified in your configuration and several other useful filters. Listing 2 contains a partial listing (top ten) of the most time-consuming functions on a typical high-performance Power Architecture processor. The negative number associated with the schedule call is due to the fact that schedule changes context to another process—that is, it never exits in the traditional manner that a function call usually does.

Conclusion

Improving Linux boot time is moving from the obscure corners of R&D labs to mainstream product development. Driven by competitive pressures in a wide variety of markets, system developers are devoting an increasing amount of effort to making sure their systems are ready to use when users want them. Always a hot topic, we are sure to see many more developments in the near future aimed at further reducing Linux system boot time.■

Christopher Hallinan is the author of *Embedded Linux Primer* and a Field Applications Engineer for MontaVista Software, Inc. He has been engaged in Linux-related work and play since 2000. He currently resides in sunny southwest Florida.

Resources

Other tools are available to help reduce your system's boot time. Bootchart is a powerful tool useful for visualizing the post-kernel initialization processes. Details can be found at **www.bootchart.org**.

Readahead is designed to pre-fetch required boot files from disk so that when they are needed, they can be read from the buffer cache for faster boot. Readahead can be customized to read specific files in a given order. You can find more about readahead at **https://fedorahosted.org/readahead**.

For the ambitious, the Moblin distribution contains a host of optimizations, which taken together, aim to produce a five-second boot on a typical Netbook: **moblin.org/projects/fast-boot**.

Elinux.org (**elinux.org/Boot_Time**) maintains a very useful collection of data related to fast boot optimizations, including more analysis and profiling tools, links to other articles and much more.

The Mesh Potato

What do you call an 802.11bg mesh router with a single FXS port that automatically forms a peer-to-peer network and relays telephone calls without landlines or cell-phone towers? A Mesh Potato, of course. DAVID ROWE

The Mesh Potato is an 802.11bg mesh router with a single FXS port (Figure 1). Adjacent Mesh Potatoes automatically form a peer-to-peer network, relaying telephone calls without landlines or cell-phone towers. The Mesh Potato hardware and software is open. The power, Ethernet and FXS ports are robust to developing-world conditions like static, lightning, bad power and accidental abuse. The Mesh Potato comes in a weatherproof box for outdoor mounting and costs about the same as any other Wi-Fi router (less than $100).

An analog phone connects to the Mesh Potato via the FXS port. FXS (Foreign eXchange Station) is a telephone interface that supplies power, dialtone and generates ringing voltage. When you make a phone call, your Mesh Potato talks to the potato down the street, which talks to the next potato, and eventually to the destination. The mesh network can be augmented via backbone links and connected to the rest of the world using VoIP trunks.

This article describes the history of the Mesh Potato Project, including how it was conceived and its development so far. I also discuss the Mesh Potato's design and the technical challenges we have faced.

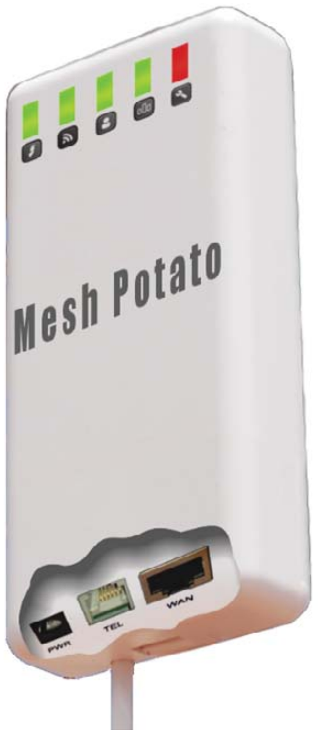


Figure 1. The Mesh Potato

History

In June 2008, I attended the Village Telco workshop in Cape Town, South Africa. The Village Telco (and I quote) is an easy-to-use, scalable, standards-based, wireless, local, do-it-yourself telephone company toolkit. Put simply, the idea is that "some guy in a village" can build a local telephone network and make a sustainable business by charging a nominal fee for calls to the PSTN (Public Switched Telephone Network) via VoIP trunks. We were in Cape Town to work out how to build the Village Telco software and hardware.

Steve Song of the Shuttleworth Foundation pulled together a fascinating team of people from the development, VoIP, mesh networking and business communities. The team was small (about ten people) and very hands-on in its outlook and skill sets (Figure 2). The breakfast and dinner conversations were fascinating—funny stories about broken-down hotels in some developing countries and sad stories about the poverty of others.

Figure 2. Village Telco Workshop 2008. Top from left to right: Jason Hudson, Jeff Fletcher, Johann Hugo, Alberto Escudero-Pascual, Steve Song, Jeff Wishnie and Alan Levin. Bottom: David Rowe, Elektra, Rael Lissoos.

One of the outcomes was the decision to build a little box called the Mesh Potato. We started out thinking we would use off-the-shelf hardware, like wireless routers and ATAs. Suddenly, it dawned on us that we didn't have to accept non-optimal, off-the-shelf hardware. We had the skills to design and build *exactly the hardware* we needed for the project. We also chose to make the hardware design open, just like the software.

Since then, we have come a long way. Through a series of development projects funded by the Shuttleworth Foundation, we have designed, debugged and built about 20 Alpha Mesh Potatoes (Figure 3). The first phone calls over Mesh Wi-Fi were made in June 2009, almost exactly one year after the project kicked off. We currently are preparing for a Beta run of Mesh Potatoes, with full production scheduled for early 2010.

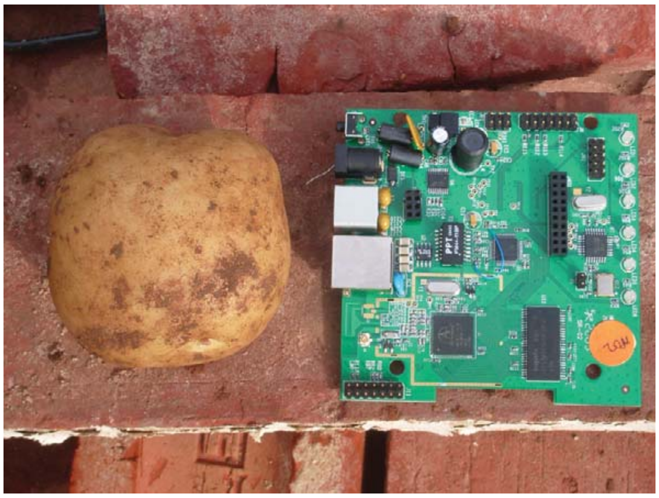


Figure 3. Prototype Mesh Potato (on the Right)

Why Not Mobile Phones?

We keep hearing how popular mobile (cell) phones are in the developing world. I have seen how well a humble cell-phone works, penetrating to the corners of some really remote areas of the world. So why do we need a Wi-Fi-based system like the Village Telco?

The answer is simple. The call costs for mobile phones are very expensive for many people in the world. In many cases, it's roughly the same cost as a mobile call in a developed country. If you are earning $1/day, a 50-cent mobile call is very expensive (Figure 4).

Figure 4. If cell phones could talk!

Although mobile phones have delivered remarkable benefits to developing countries, the mobile oligopolies that have emerged in the process have kept call charges artificially high. Worse, mobile operators tend to function as "walled gardens" in order to entrench their market share. Just compare the price of an e-mail message on the Internet (zero) and via a cell phone (20 cents for a text message), and you get some idea of the problem.

Communities in the developing world need an alternative. Hence the need for the Village Telco—a system that uses commodity Wi-Fi technology and unlicensed spectrum to provide low-cost phone calls.

Key Features

The Mesh Potato runs B.A.T.M.A.N. (see Resources) mesh routing software, Asterisk, the Speex voice codec and Oslec echo cancellation. No cell-phone towers, no landlines, no big Telcos are required. Local entrepreneurs can roll out their own Village Telco system using a modest server and a bunch of Mesh Potatoes—community-owned telephony.

The mesh network is self-organising and self-healing. If a node goes down, B.A.T.M.A.N. automatically re-routes the calls. We are building custom hardware specifically for developing communities using open hardware and software principles. I am intrigued by the idea of developing custom open hardware devices—no need to accept whatever is available off the shelf. Most of the value in any router-type product is delivered by the software, which these days is usually Linux. The idea of relying on closed, proprietary, not-quite-right hardware is obsolete.

The Mesh Potato is as open as we can make it. We have minimised binary blobs and deliberately chosen open over proprietary software. The Mesh Potato is Atheros-based, as this allowed the use of the MadWifi open-source WLAN driver. We use the Speex and GSM codecs instead of g729 and Oslec instead of a proprietary echo canceler. The hardware schematics are available on-line.

The Mesh Potato will be mass-produced in large numbers. Open projects like this will start to exert influence over future telephony systems. For example, if 1,000 Village Telco operators are trunking calls encoded in Speex, VoIP trunk operators will need to support Speex. This represents an important paradigm shift. The Open community now has a chance to set standards, rather than have to play along with "standards" based on closed hardware and software.

I have developed open hardware telephony products in the past, including the IP04, which is manufactured by Atcom (see Resources). So it was natural that we team with Atcom for the board-level PCB layout and volume manufacture of the Mesh Potato. Atcom is a VoIP hardware company from Shenzhen, China, that understands and embraces open hardware and open software. Atcom is handling the board-level PCB layout and volume manufacture of the Mesh Potato.

Technical Overview

Figure 5 is a mud map of the Mesh Potato hardware. The Mesh Potato uses an Atheros AR2317 System-on-a-Chip (SoC), which is a very low-cost router chip that combines an MIPS processor running at about 200MHz with 802.11bg Wi-Fi. It has built-in interfaces for LEDs, SDRAM and serial Flash. Best of all, it is well supported by OpenWRT and MadWifi. The FXS hardware, drivers and other firmware we have developed are generic. It is possible to port them to other router architectures. In very high volumes, it would make sense to integrate the FXS chipset functionality onto the SoC.

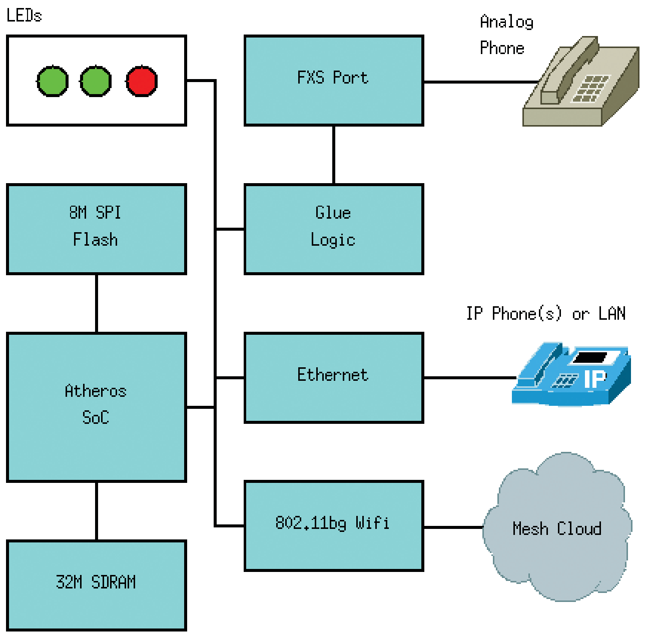


Figure 5. Mesh Potato Hardware Architecture

Development Story

Development of the Mesh Potato kicked off in September 2008. Along the way, we had a few design issues and many challenging bugs to fix. As part of the open design philosophy, we have documented the design and even some of the “bug hunts” on the Village Telco blog (see Resources).

CPU Load

A key question was CPU load. Could a humble router CPU support Asterisk, a speech codec, an echo canceller and route several other phone calls over the mesh at the same time? To answer this question, we designed a test with all of these software modules running at the same time. As this was in the early days, and we didn’t have any FXS hardware, we simulated the speech samples coming from the FXS port.

To model the maximum load of the system, we thought about a worst-case scenario of one mesh node routing 15 phone calls for its peers. This means the node would have to receive, then re-transmit, voice packets for 15 simultaneous phone calls. At the same time, the node had a phone call of its own, which meant the speech codec, echo canceller and Asterisk were all running. To test this scenario, we set up some Asterisk boxes to generate calls and used commodity Atheros Wi-Fi hardware to run the prototype Mesh Potato firmware.

The test passed. Call quality was maintained, provided we used 80ms voice packets to reduce the overhead of many small VoIP packets.

Listing 1. Small, Fast Serial ISR to Minimise Instruction-Cache Thrashing

```
static irqreturn_t serial8250_interrupt(int irq, void *dev_id)
{
    struct irq_info        *i = dev_id;
    struct list_head       *l;
    struct uart_8250_port  *up;
    unsigned int           lsr;
    unsigned char          ch;
    unsigned int           iir;
    int                    count = 0;

    spin_lock(&i->lock);

    l = i->head;
    up = list_entry(l, struct uart_8250_port, list);

    lsr = serial_inp(up, UART_LSR);

    while (lsr & UART_LSR_DR) {
        ch = serial_inp(up, UART_RX);
        ch = mp_buffertxrx(ch);
        serial_outp(up, UART_TX, ch);
        lsr = serial_inp(up, UART_LSR);
        count++;
    }

    spin_unlock(&i->lock);

    return IRQ_RETVAL(1);
}
```

Stuck Beacons and Ad Hoc Wi-Fi

The MadWifi driver had a nasty “stuck beacon” problem that was specific to ad hoc mode, which is required for mesh networking. Nodes attempt to adjust their internal clocks based on reception of beacons from other nodes. Under certain situations, this caused a race condition, which locked up the driver’s transmit queue. This means the driver would stop working for about 30 seconds.

Elektra worked hard with the MadWifi developers to establish and test a workaround. The driver is started in access-point (rather than ad hoc) mode, and then we create a virtual ad hoc access point that does not transmit beacons:

```
$ wlanconfig ath0 create wlandev wifi0 wlanmode adhoc nosbeacon
```

Beacons are unnecessary for our mesh network, and B.A.T.M.A.N. broadcasts its own packets at regular intervals. In access-point mode, there is no attempt to adjust the MAC clock, so the race condition is avoided.

FXS Interface

Low-cost Wi-Fi SoC devices are highly integrated. They have minimum hardware interfaces—just enough for their core purpose.

To support an FXS interface, we normally use a CPU that has some sort of Time Division Multiplex (TDM) bus to support one or more 64kb/s bit streams of speech samples. The TDM bus hardware then handles DMA of the speech samples, presenting them as buffers to the device driver.

Every chip I have worked on in 20 years of telephony hardware design had a TDM bus. The AR2317 has many other features we need (low cost, Wi-Fi, OpenWRT support), however, there's no TDM bus!

So, I worked out a scheme to send and receive speech samples via the RS-232 console port, using some external glue logic and a small microcontroller to buffer the speech samples. It's not elegant, but it's reasonably low cost and it works, allowing us to use a low-cost AR2317 SoC for a very different application (VoIP) than it was designed for.

As the speech samples arrive in the RS-232 port, they are buffered by the SoC serial FIFO. When the FIFO fills, it interrupts the CPU every 1ms. Now, 1ms is a very high interrupt rate. What this means is every 1ms, the CPU must stop what it's doing and run the Interrupt Service Routine (ISR). This can cause a big performance hit, as the instruction cache contents must be flushed and replaced every 1ms.

To minimise this performance hit, I rewrote the Linux drivers/serial/8250.c interrupt handler to be as small as possible (Listing 1). This is small enough to consume only a tiny amount of valuable instruction cache and is likely to stay resident in the cache between interrupts. The functions it "calls" are either macros or very short inline functions.

Small Team, Big Company and Calibration

Being a small team on a modest budget, we have experienced difficulty obtaining all the data we require from the SoC vendor, Atheros. This is a problem with some chip vendors. Unless you can pay big up-front fees or have a one-million chip order pending, it is hard to get support or even basic data. This is a pity, as I feel many chip vendors rely increasingly on the Open Source community for their firmware and build tools and hence their chip sales. Also, we are trying to design a mass-market product that will sell more of their chips.

This is not true of all chip vendors. Analog Devices has

Powerful: Rhino

Rhino M6400/E6500
- Dell Precision M6400/ Latitude E6500
- 2.2-3.0 GHz Core 2 Duo or 2.5 GHz Core 2 Quad
- Up to 17" WUXGA LCD w/ X@1920x1200
- NVidia Quadro FX 3700M
- 80-500 GB hard drive
- Up to 16 GB RAM
- DVD±RW or Blu-ray
- 802.11a/g/n
- Starts at $1330

- High performance NVidia 3-D on a WUXGA widescreen
- High performance Core 2 Quad, 16 GB RAM
- Ultimate configurability — choose your laptop's features
- One year Linux tech support — phone and email
- Three year manufacturer's on-site warranty
- Choice of pre-installed Linux distribution:

Tablet: Raven

Raven X200 Tablet
- ThinkPad X200 tablet by Lenovo
- 12.1" WXGA w/ X@1280x800
- 1.2-1.86 GHz Core 2 Duo
- Up to 8 GB RAM
- 80-320 GB hard drive / 128 GB SSD
- Pen/stylus input to screen
- Dynamic screen rotation
- Starts at $2200

Rugged: Tarantula

Tarantula CF-30
- Panasonic Toughbook CF-30
- Fully rugged MIL-SPEC-810F tested: drops, dust, moisture & more
- 13.3" XGA TouchScreen
- 1.6 GHz Core 2 Duo
- Up to 8 GB RAM
- 80-320 GB hard drive
- Call for quote

EmperorLinux
...where Linux & laptops converge

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

Model specifications and availability may vary.

been very helpful with the IP04-based open hardware embedded IP-PBX (see Resources) that uses the Analog Devices Blackfin CPU. The company provided comprehensive data, time from many support engineers and even funded some of the IP04's development.

One particular problem area with the AR2317 has been RF calibration. Each AR2317 chip is slightly different and must be calibrated on the production line using automated test equipment. The software to perform this calibration and even the calibration registers' documentation is very closed. This has made it hard to calibrate our prototypes to obtain optimum RF performance.

However, where there is a will, there is a way. With Atcom's help, we have teamed with other Wi-Fi router vendors who have the required calibration equipment on their production lines.

Testing the Alphas

In early 2009, we started design of the actual custom Mesh Potato hardware. By the middle of 2009, we had built approximately 20 prototype Mesh Potatoes, and in July, we held the second Village Telco Workshop to test the alphas and plan the next phase of the project.

Server-less Asterisk

One nice demo was a simple Asterisk dialplan, placed on each Mesh Potato:

```
exten => _XX,1,Dial(SIP/4000@10.130.1.${EXTEN})
```

This allowed us to construct a serverless, peer-to-peer voice network. This dialplan takes a two-digit number XX and lets you dial another Mesh Potato with the IP of 10.130.1.XX. It's an instant local phone network on just a few watts/node, no server required (let alone a cell-phone tower or central office exchange). Simply switch on your potato, and you can make calls. Imagine the applications for Katrina-style disaster situations where you need instant communications.

Testing and Lessons Learned

We spent some time setting up mesh networks and testing the limits of the system by listening to voice quality. Using the B.A.T.M.A.N. debug modes, we could see the mesh hops go around corners and through windows to relay calls from one node to another.

We still have a lot to learn about everything that affects call quality. There are many factors, such as Wi-Fi propagation, antennas, speech coding, jitter buffers, interference and system load. We are planning a small R&D project to study and optimise call quality in marginal conditions.

We need effective ways to instruct people on how to set up a reliable mesh network (like a picture book or videos or real-time metrics of quality such as a GUI or dialtone).

Wandering around in the South African winter sunshine with a Mesh Potato and a battery, I had an "ah-ha" moment that frankly sent shivers down my spine. This thing really works! You sometimes lose track of the big picture when you are engineering all the details.

Our big goal now is to simplify the installation and configuration as much as possible. At the workshop, we spent some time trying to get a Mesh Potato connected to an Asterisk server, and it was the usual time-consuming Asterisk conf file and command-line frustration. It's hard the first time, but gets easier as you gain experience. However, we want to make Village Telco setup easy for thousands of first-time users. This experience drove the point home: we need to make configuration as straightforward as possible.

Production Potatoes

It has been a pleasure to work with the Shuttleworth Foundation, Elektra and Atcom on this project. We also have had amazing input from the participants in the two Village Telco Workshops and members of the Village Telco Google Group. And, we still have a lot to do. By early 2010, we plan to resolve the remaining calibration issues, perform Beta trials and obtain type approval for the Mesh Potato. At the higher levels of the Village Telco Project, we need to integrate a billing system and the Afrimesh GUI, and integrate into a simple one-click installation.

I am confident we will achieve this and more. We have shown that a small, talented team can develop custom Wi-Fi hardware specifically for their needs. Community-based product development for community-based telephony—how cool is that!■

David Rowe has 20 years' experience in the development of DSP-based telephony and sat-com hardware/software. In 2005, David founded the Free Telephony Project (www.rowetel.com/ucasterisk), which has pioneered the field of open hardware embedded VoIP products. His open-source contributions include the first open telephony hardware drivers in 1999 and the Oslec echo canceller (www.rowetel.com/ucasterisk/oslec.html). David's other interests include building and advocating electric vehicles and VoIP technology for the developing world.

Resources

Village Telco: **villagetelco.org**

Mobile Phones and Walled Gardens: **manypossibilities.net/2009/01/why-wifi-in-africa**

B.A.T.M.A.N.: **open-mesh.org**

Oslec Echo Canceller: **rowetel.com/ucasterisk/oslec.html**

Atcom: **atcom.cn**

IP04 Open Hardware IP-PBX: **rowetel.com/ucasterisk**

Afrimesh Mesh Network GUI: **code.google.com/p/afrimesh**

Village Telco Google Group: **groups.google.com/group/village-telco-dev**

Multisession Workstations

Press F1 for bash, F2 for Windows, F3 for Ubuntu, F4 for Mac OS and F5 for Citrix. Linux makes it all possible, and you don't even need a hard drive! JORGE SALGADO

The Linux Terminal Server Project (LTSP) has been around for years now, and it gets better with each new release. In the beginning, it was targeted at providing schools with a means to use low-cost computers as Linux terminal clients. It was a huge success; so much so, that LTSP now is included in several Linux distros, such as Edubuntu, as a regular package.

LTSP lets you tailor it to deliver multiple OS desktops to every workstation on the LAN, using just PXE-bootable desktops or thin clients. With this type of setup, users simply have to press Ctrl-Alt-Fn to access different desktops. The following shows an example menu you could present to LTSP users:

- Ctrl-Alt-F1: Linux shell.
- Ctrl-Alt-F2: Windows desktop for Internet browsing and e-mail.
- Ctrl-Alt-F3: Ubuntu with development tools.
- Ctrl-Alt-F4: Mac OS for graphics work.
- Ctrl-Alt-F5: Remote Citrix access for corporate ERP and CRM.

To set up an environment that supports the above options, the following steps are required:

- Set up an LTSP environment.
- Install the required client connection tools.
- Create scripts to use the client connections.
- Configure LTSP files to enable one or several screens to use the new client connection.

Setting Up an LTSP Environment

The first step is installing the LTSP packages on the Linux distro of your choice. Many recent distro releases have ready-to-install LTSP packages available in their repositories, so you probably can use your favorite package manager out of the box. This way, you should have your LTSP server up and running in a matter of minutes. On Ubuntu use:

```
$ sudo apt-get install ltsp-server-standalone
$ sudo ltsp-build-client
```

For detailed install instructions for other distros, check the LTSP Web site (see Resources).

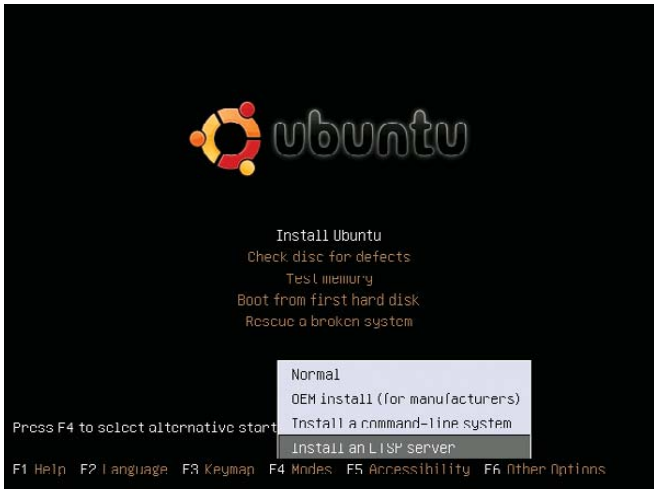
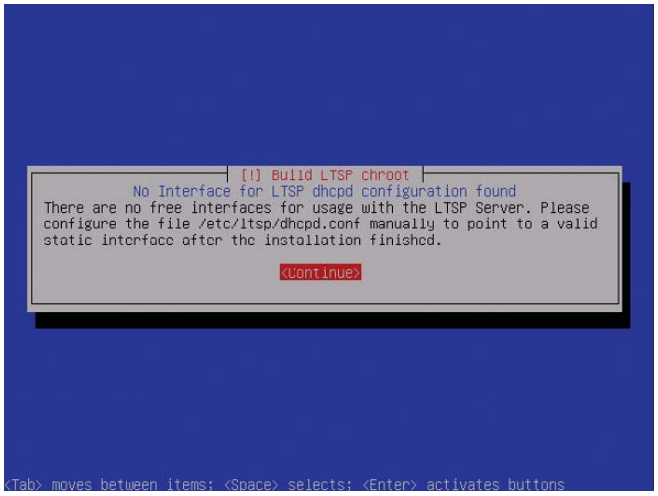

ubuntu
Install Ubuntu
Check disc for defects
Test memory
Boot from first hard disk
Rescue a broken system
Normal
OEM install (for manufacturers)
Install a command-line system
Install an LTSP server
Press F4 to select alternative start
F1 Help F2 Language F3 Keymap F4 Modes F5 Accessibility F6 Other Options

Figure 1. Ubuntu Alternate Install (Install an LTSP Server)



[!] Build LTSP chroot

No Interface for LTSP dhcpd configuration found

There are no free interfaces for usage with the LTSP Server. Please configure the file /etc/ltsp/dhcpd.conf manually to point to a valid static interface after the installation finished.

<Continue>
<Tab> moves between items; <Space> selects; <Enter> activates buttons

Figure 2. Ubuntu PXE Boot Client LTSP Session

For an easy out-of-the-box experience, download and boot from an Ubuntu Alternate CD, press F4, and choose Install an LTSP server (Figure 1).

From there, you install Ubuntu as usual. The only difference is that near the end of the install, you will see a warning about a second Ethernet card (Figure 2).

The error occurs because the installer defaults to using a second Ethernet card dedicated to boot terminals. You may use this setting if you like, but if you already have a DHCP server in your environment, you can use it.

Once you finish the LTSP install, log on to your server and build the LTSP image with the following command:

```
$ sudo ltsp-build-client
```

This takes a couple minutes. While it builds, you will see a text progress bar—get used to it, because you will need to rebuild this image several times.

If you are lucky and your DHCP server is a nice Linux box, edit your /etc/dhcpd.conf file to point your network boot options to the LTSP box. Add the following lines, and restart the DHCP service afterward:

```
option tftp-server-name  "mynew.ltsp.server";
option bootfile-name     "/ltsp/i386/pxelinux.0";
```

If you're unlucky and you have to set this up with a Windows server, as an administrator, open the DHCP configuration screen and add the configuration options below:

```
017 Root Path:              /opt/ltsp/i386
066 Boot Server Host Name:  <LTSP Server ip address>
067 Bootfile Name:          ltsp/i386/pxelinux.0
```

In addition, most Linux/BSD-based firewall software appliances, such as pfsense and endianFW, have options for this on their DHCP configuration screens.

If everything goes well, your LTSP environment is ready to boot network clients. Reboot one of your desktops and select network boot. You should see your desktop receive an IP address from your DHCP server, a large stream of dots when the boot image downloads from the TFTP server and then a regular boot splash screen from your distro. Then finally, your LTSP session will start (Figure 3).

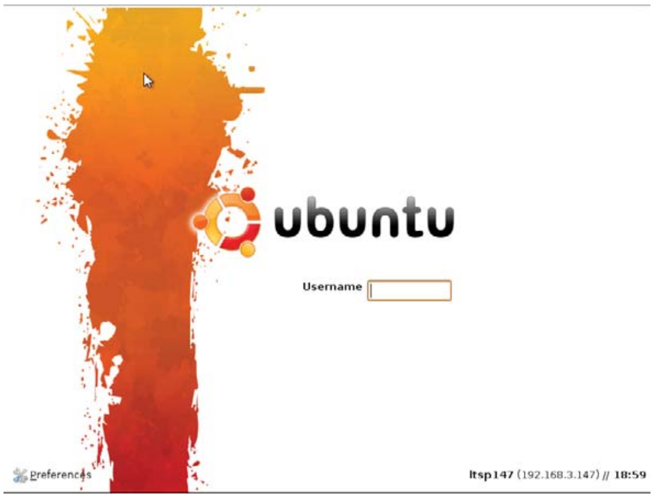


Figure 3. Ubuntu PXE Boot Client LTSP Session (note the lower right-hand corner of the screen)

You're now about a third of the way there. Next, let's go to the core of multisession setup and start installing the connection tools you'll need to connect to other types of sessions.

Installing Connection Tools

You need to set up your workstations to allow multiple remote sessions, connected to different servers and different OSes, each using the required specific connection protocol. LTSP includes only three types of connect scripts: shell, graphical Linux and Windows remote desktop. This is great, because out of the box, you are able to set up a shell session, a full diskless Linux client and a Windows remote desktop session.

LTSP lacks other interesting protocols, like VNC, NX and Citrix. For those, you'll need to install some tools and client applications.

Installing Additional Client Connection Tools

One of the features I like most about LTSP is that each time you update your ltsp-image, you actually "build" a small footprint distro. This means you can install packages, startup scripts or anything you want, into the LTSP image "distro", and then simply update the image as needed. Don't confuse your "server" distro, with the LTSP image distro; they are completely different. Your server is mainly a building environment for the LTSP image, which is why images built from Ubuntu, look like Ubuntu. Images built on OpenSUSE, look and taste like OpenSUSE, and so on.

This hasn't always been the case though. In earlier releases, LTSP produced images that weren't related to the server distro. Earlier LTSP images were more like a distro of their own. Now, you can choose your server flavor and produce images of the same flavor. This allows you to keep using the distro you are familiar with on the server and on your thin clients.

Remember: things you install on your server are not installed onto your LTSP images. Within your server lives a mini-distro that is used to build the LTSP images. To access your building distro, you must chroot into it from your server. Simply type the following:

```
$ sudo chroot /opt/ltsp/i386
```

You are now working on your LTSP image distro root directory. This environment is what images will be built from. From here on, all you need to do to install software is type your standard distro shell commands. Let's try a small shell picture viewer as an example (this will be useful later):

```
$ apt-get install zgv
```

To exit from the chroot session and build the new image that now will contain the zgv command, do the following:

```
$ exit                    # to exit the chroot session
$ sudo ltsp-update-image  # to build the update image
```

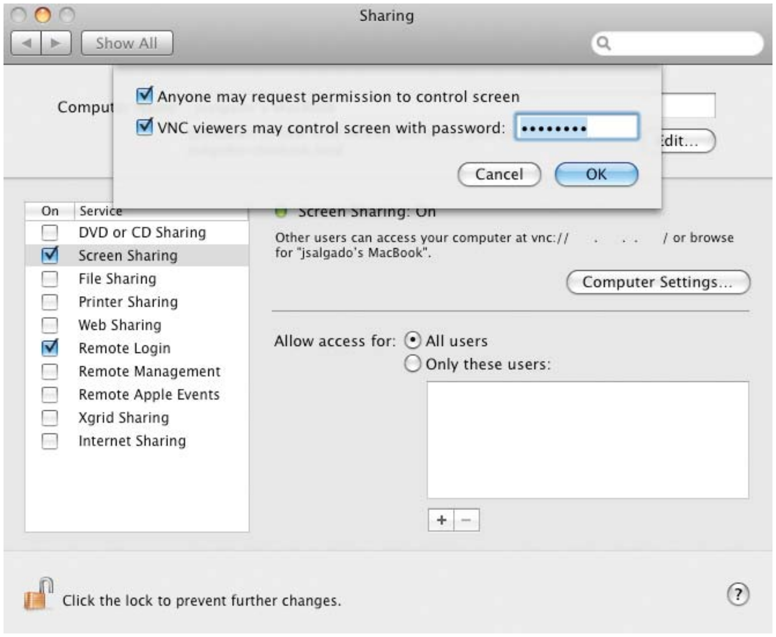
After the image is built, reset your PXE boot client, press Ctrl-Alt-F1 to go to the shell session and check that zgv is available. In my setups, Ctrl-Alt-F1 always is a shell session, regardless of the settings in the lts.conf file.

Building the image is a time-consuming task, so you probably will want to wait until all needed packages are installed and built, and test once rather than multiple times.

For the objectives in this article, you will need to install a VNC client so you can connect to a Mac OS X session and the Citrix XenApp client.

Connecting to Mac OS X

First, set your MAC to allow incoming connections. Go to your Mac OS X server and open System Preferences. Under Internet and Network, click Sharing, then select the Screen Sharing check box. Next, click the Computer Settings button, check Anyone may request permission to control screen, and click VNC viewers may control screen with password, and set a password. If you don't set this option, you will receive a "No matching security types" error each time you try to connect to your Mac OS X system (Figure 4).



Sharing

☑ Anyone may request permission to control screen
☑ VNC viewers may control screen with password: ••••••••

Cancel | OK

On	Service
☐	DVD or CD Sharing
☑	Screen Sharing
☐	File Sharing
☐	Printer Sharing
☐	Web Sharing
☑	Remote Login
☐	Remote Management
☐	Remote Apple Events
☐	Xgrid Sharing
☐	Internet Sharing

Screen Sharing: On

Other users can access your computer at vnc:// . . . / or browse for "jsalgado's MacBook".

Computer Settings...

Allow access for: [✓ All users] Only these users:

Click the lock to prevent further changes.

Figure 4. Mac OS X Enable Remote Access

Now, back to the LTSP server. Install the VNC client and VNC server (I used xvnc4viewer, because it was available in the Ubuntu 9 repositories, but you may use any VNC client that is available for your distro):

```
$ sudo chroot /opt/ltsp/i386
$ apt-get install xvnc4viewer vnc4server
```

I know it doesn't make much sense to install a VNC server on your "client" image; however, you will need the vncpasswd command from it later.

Because your LTSP image and your main LTSP server are siblings, you are free to test software behavior on your server first. Bear in mind that they are closely related, but not identical twins. Just because the server is able to do something, doesn't ensure that the LTSP image also will be able to do it. However, if something doesn't work on your server, don't waste your time trying to make it work on the LTSP image either.

To test from your server, install xvnc4viewer directly on your server, and start a connection to your Mac. You must use the -FullColor option; if you don't, your MAC will not allow the connection:

```
$ xvnc4viewer -FullColor -FullScreen your.mac.ip.address
```

You will get a VNC authentication window asking for the password you set earlier on your Mac. Next, you should see a beautiful Aqua Mac OS desktop on top of your Linux screen!

This is great, but the password window doesn't work correctly on the LTSP image. Because you didn't load a window manager before the VNC viewer, the authentication window pops up with no mouse pointer visible, and you need to click on the box to start writing. So, let's avoid that problem. Go back to the chroot session for your LTSP image, and type `vncpasswd`. This will create the .vnc/passwd file, which you will use as a parameter to your `xvnc4viewer` command. Move the newly created file to /usr/share/ltsp/vnc-passwd.

Now, you need to create a screen script for this VNC session. Those scripts are in the /usr/share/ltsp/screen.d folder. Change directories to that folder, copy the rdesktop screen script and modify it to be a VNC script. Call this new script vnc1, and make it look like this:

```
#!/bin/sh

PATH=/bin:$PATH; export PATH
. /usr/share/ltsp/screen-x-common

VNC_OPTIONS=${VNC_OPTIONS:-"-FullScreen"}
VNCVIEWER_OPTIONS="${VNC_OPTIONS} -FullScreen $* ${VNC_SERVER}"

if [ -x /usr/share/ltsp/xinitrc ]; then
    xinitrc=/usr/share/ltsp/xinitrc
fi

xinit $xinitrc /usr/bin/xvnc4viewer ${VNCVIEWER_OPTIONS} \
               -- ${DISPLAY} vt${TTY} ${X_ARGS} -br >/dev/null
```

Now, exit your chroot session, and edit the /var/lib/tftpboot/ltsp/i386/lts.conf file. You should add two new parameters, and set your screen_04 to use your new script. It now should look something like this:

```
[default]
```

```
VNC_OPTIONS  = "-FullColor -passwd /usr/share/ltsp/vnc-passwd"
VNC_SERVER   = your.mac.ip.address
...
SCREEN_04    = vnc1
```

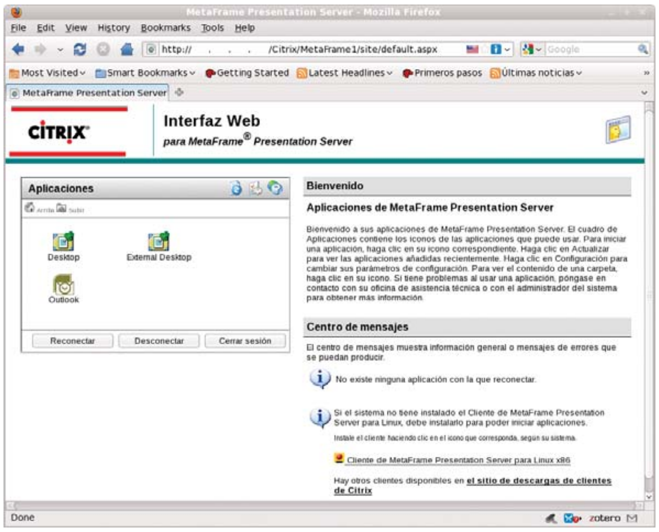
Next, rebuild your LTSP image with the `ltsp-update-image` command, wait for the process to end and test it on your PXE boot client. Press Ctrl-Alt-F4, and you should see a Mac remote session.

Because the objective is thin-client corporate infrastructure, you most likely will want to have multiple connections to your Mac OS "server". For this, there is Aqua Connect Terminal Server. Using it, you can connect several remote users to a single Mac OS X server (remember to double-check Apple's licensing terms).

Connecting to Citrix

Go to the Citrix Web site, look for the XenApp Linux client and download it. Copy the downloaded file into your chroot system. In your chrooted session, untar the Citrix client file. After decompression, you should have a new folder named linuxx86 and a few extra files, including the install script called setupwfc. To install, as root, execute `./setupwfc`, and answer the text wizard questions. You may have to fill in some dependencies for your distro, but after a few moments, your LTSP image will be Citrix-enabled.

The Citrix server configuration is beyond the scope of this article. You should start with a working Citrix XenApp Server. The good news is that you don't even need to be one of the Citrix server administrators at your company, you just need to have the user name and password for an account with published applications on the server. In other words, if you already have access to a desktop or an application via Citrix, you can set up that connection as one of the screens on your multisession terminal server. Simply log in to your Citrix session as a regular user and download



Bienvenido

Aplicaciones de MetaFrame Presentation Server

Bienvenido a sus aplicaciones de MetaFrame Presentation Server. El cuadro de Aplicaciones contiene los iconos de las aplicaciones que puede usar. Para iniciar una aplicación, haga clic en su icono correspondiente. Haga clic en Actualizar para ver las aplicaciones añadidas recientemente. Haga clic en Configuración para cambiar sus parámetros de configuración. Para ver el contenido de una carpeta, haga clic en su icono. Si tiene problemas al usar una aplicación, póngase en contacto con su oficina de asistencia técnica o con el administrador del sistema para obtener más información.

Centro de mensajes

El centro de mensajes muestra información general o mensajes de errores que se puedan producir.

No existe ninguna aplicación con la que reconectar.

Si el sistema no tiene instalado el Cliente de MetaFrame Presentation Server para Linux, debe instalarlo para poder iniciar aplicaciones.

Instale el cliente haciendo clic en el icono que corresponda, según su sistema.

Cliente de MetaFrame Presentation Server para Linux x86

Hay otros clientes disponibles en **el sitio de descargas de clientes de Citrix**

Figure 5. Citrix ICA File Download from Web Interface Icon

the session definition ICA file (Figure 5). ICA files are actually text files that contain the information and settings to establish a connection to a XenApp server. The easiest way to download this file is to right-click on one of the icons displayed on your Citrix server Web interface and select Save link as.

Once you have your ICA file, copy it to the Citrix client install directory on your chroot session:

```
$ cp my-ica-file.ica /usr/lib/ICAClient/desktop.ica
```

Now, let's create the screen script for the Citrix session in /usr/share/ltsp/screen.d. We'll call this script citrix1:

```
#!/bin/sh
# Copy the ica file to a temp file because
# wfica deletes the file on execution.

cp /usr/lib/ICAClient/desktop.ica \
   /usr/lib/ICAClient/temp-file-desktop.ica

sudo xinit /usr/lib/ICAClient/wfica \
           /usr/lib/ICAClient/temp-file-desktop.ica
```

Notice that XenApp is the new name for the Citrix presentation server, so any Citrix server XenApp or presentation server will work.

Finally, exit your chroot session and add the new screen parameter for the citrix1 script in your lts.conf file. It should look like this:

```
[default]
    ...
    SCREEN_05          = citrix1
```

Now you can rebuild your LTSP image with the `ltsp-update-image` command, and test the Citrix session on your PXE boot client when you press Ctrl-Alt-F5.

Connecting to a Windows Terminal Server

The rdesktop client and script are included in the LTSP install package, so you won't have to create scripts or install new packages. All you need to do is include their screen parameters in the lts.conf file. Your final file should look like this:

```
[default]
    VNC_OPTIONS  = "-FullColor -passwd /usr/share/ltsp/vnc-passwd"
    VNC_SERVER   = your.mac.ip.address
    RDP_OPTIONS  = "-a 16"
    RDP_SERVER   = your.windowsTS.ip.address
    SCREEN_01    = shell
    SCREEN_02    = rdesktop
    SCREEN_03    = ldm
    SCREEN_04    = vnc1
    SCREEN_05    = citrix
```

This time, you don't need to run `ltsp-update-image`. When you use the /var/lib/tftpboot/ltsp/i386/lts.conf file, it's read directly from the server and not from the ltsp-image. Be aware that there is another lts.conf file inside the chroot directory; avoid using that one.

Stopping the Screen Takeover Problem

If you've tested each step of your progress, you surely know by now that sometimes different "screens" suddenly take over the monitor output. They seem to be fighting each other to be top dog. This is not a bug. It happens when a remote session login screen timeouts. Windows and Citrix wait patiently for your login credentials, but after some inactivity time, they drop your connection. When this happens, X dies. Then your LTSP terminal restarts X and restarts the connection. This pulls the visible screen to the newly started X screen, taking over the monitor output.

To avoid this effect, you need to log in to all of your available sessions. Logged-in sessions also have timeouts, but they are much longer.

The simplest solution is based on an idea I found in an older version (from LTSP 3.0) of the rdesktop script. The script included a "read" statement just before the xinit call. That way, users had to press a key to start their rdesktop session. You can use that same approach. It's not fancy, but it works.

A more stylish solution is to use zgv to show a picture just before the session start line. zgv closes when users press the Enter or Esc key. Remember to add a "Press Enter to start" banner to your image.

Controlling Session Access

LTSP also lets you provide settings for groups or workstations or for individual workstations, identified by IP address, MAC address or hostname. This allows you to set which sessions are seen on which workstations and even to configure specific hardware for that workstation. The following lts.conf file shows an example of how this can be done:

```
[default]
    SCREEN_01       = shell

[LINUXER]
    SCREEN_03       = ldm

[12:34:56:78:9a:bc]
    XSERVER         = ati
    X_MOUSE_DEVICE  = /dev/ttyS0
    SCREEN_03       = ldm
    SCREEN_05       = citrix

[192.168.0.4]
    SCREEN_04       = vnc1
```

Conclusion

You could fill a book with examples of uses and configurations for LTSP. I've been working with it since version 1.0, going on ten years now, and with each new version, there are useful new features. For further technical information on the project, go to the LTSP Wiki, and if you get a chance, please support the project with a small donation.■

Jorge Salgado is a Senior Infrastructure Consultant. He holds MCTS, NCLP9 and LPIC1 certifications and spends most of his time pushing companies to get the best from Citrix, VMware and Linux technologies. He lives in the Queretaro area and can be contacted at jsalgado@smart4lan.com.

Resources

LTSP: **www.ltsp.org**

LTSP Downloads and Install Guides: **wiki.ltsp.org/twiki/bin/view/Ltsp/DownLoads**

Citrix Client Download: **www.citrix.com**

Mac OS Terminal Server: **www.aquaconnect.net**

10.2" 1024x600 Matte LCD with LED backlight

- Intel Atom N270 1.6 GHz CPU
- Intel GMA950 integrated video
- Optional internal 3G cellular modem for high-speed Internet connectivity virtually anywhere
- Draft-N WiFi
- Webcam
- Ubuntu, Ubuntu Netbook Remix, Kubuntu, Jolicloud, Moblin, and others
- Up to 3.5 hrs battery life
- Slim 10.5" x 7.3" x 1.1"
- 2.8 lbs.

zareason.com

ubuntu Get your Ubuntu case badge here

zareason

Gord spent too many years learning clusters

Learn what Gord knows

Free Clusters for Dummies book and HPC white paper

www.platform.com/gord

Platform Computing

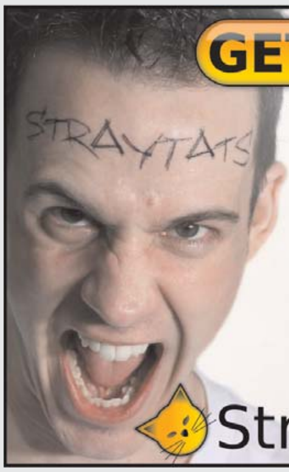
Texas' Oldest Craft Brewery

WWW.SAINTARNOLD.COM

Archie McPhee

Supplying the world with impractical weirdness for over 25 years!

mcphee.com



GET NOTICED!

With custom, temporary tattoos! Any image!

5% Off With This Code:

linuxjournal

Great for conventions!

StrayTats.com

The first and only radio show broadcast in the USA dedicated exclusively to spreading the word about the LINUX OPERATING SYSTEM and FOSS.

GUTSY GEEKS COMPUTER SHOW

gutsygeeks.com

BEAGLE BOARDS & PERIPHERALS

LOW-COST, NO FAN, SINGLE-BOARD COMPUTER

AVAILABLE EXCLUSIVELY AT DIGI-KEY

beagleboard

only $149^{00}

Digi-Key CORPORATION

www.digikey.com

axigen MAIL SERVER

The Mail Server for IT Professionals

www.axigen.com

Natuba™

HAS AN API THAT EVEN SHAWN POWERS IS COMFORTABLE USING. IT'S CALLED "EMAIL."

Shawn Powers
Linux Journal Associate Editor

Follow Shawn on Natuba!
Natuba.com/shawnp0wers

Share your photos on Natuba.com today! Expose yourself™

© 2009 Hush Labs, LLC. All rights reserved.

Polywell Mini-ITX PCs

ITX with VESA / Wallmount

NVIDIA® ION™

The World's Small, Greenest, Fanless PC with Blu-Ray Ready

ITX-40A with NVIDA® ION™ Graphics $199 Barebone system
4GB RAM, 1.6GHz Intel® 4W Processor $599 with Blu-ray, 500G HD

ITX-10A 1.4" ThinPC

ITX-20A with SlimDVD

ITX-30A with PCI Riser

Fanless Slim PC with Intel® 1.6GHz 1W Mobile Atom™ Processor, DC12V Power-in, 1GB DDR2 RAM starts at $199
Supports SATA Hard Drive or Solid State Drive (SSD), Optional PCI RISER Slot for TV Tuner or other Add-on Device on ITX-30A

ITX-1000C with 4LAN and WiFi Option

Over 250 Mini-ITX Models Available:
- NVIDAI® GeForce 8200/8100 with AMD® Athlon/Phenom Processor
- NVIDIA® GeForce 9300/7100/7050 with Intel® Core 2 Duo Processor
- PCI, PCIe, MiniPCIe Slot for TV Tuner or Industrial Add-on
- Custom Design Chassis for Small to Mid Size OEM Project

888.765.9686
linuxsales@polywell.com
polywell.com/us/Lx

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

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

POLY POLYWELL

NVIDIA, ION, nForce, GeForce and combinations thereof are trademarks of NVIDIA Corporation. Other names are for informational purposes only and may be trademarks of their respective owners.

Is “Open Phone” an Oxymoron?

Will we ever get a truly open phone? DOC SEARLS

Linux and embedded converge at the technical level, but they clash at the cultural one. At the technical level, Linux is like an element in the periodic table, or a pile of 2x4s. You can use it to build all kinds of stuff. Sure enough, all kinds of stuff does get built, including embedded stuff that most hard-core Linux hackers would never develop, or that they would develop in a very different way. We’re talking here about stuff that is typically closed as a vault. Examples include electronic picture frames, security cameras, airplane avionics, vacuum cleaners, assembly-line robots and medical equipment. And, of course, mobile phones.

Linux has been at the heart of millions of phones made and sold during the last few years. Few of these, however, are what you would call “open” to the degree that you can do what you please with them—that is, as you would with a PC. At the low end, phones are dumb devices that do little more than telephony. At the high end, they’re smart devices that do lots of different things. But alas, not in the same ways—meaning, there is no “write once, run everywhere”.

Right now, the Linux smartphone market is split between Android, LiMo and Palm. You write apps for Android in Google’s own version of Java5 (described by one developer as “not J2ME, but not quite Java5 either”). You write apps for LiMo in C or C++. You write apps for Palm Pre in JavaScript. Of those three, Palm is the only standalone vertically integrated platform, kind of like the iPhone. The LiMo vs. Android situation is a lot more complicated. The LiMo Foundation has a pile of big names behind it, not including Google. Google’s Open Handset Alliance (all developing for Android) includes its own pile of corporate heavyweights, some of which also are involved with LiMo. The differences are easy to play down. For example, LiMo focuses exclusively on middleware, while Android focuses on applications (but covers the whole stack). But, Linux for phones is beyond forked. Which brings me to another F-word.

Freedom.

Phone makers and phone companies don’t come from freedom. They come from control—specifically, of whole markets. Take one example from page 12 of the LiMo Foundation’s Introductory .pdf (dated July 2009). There, it explains where LiMo’s middleware platform fits in the scheme of things. Its middleware sits below three other layers: 1) UI/Applications, “As selected by the handset maker/operator”; 2) General Content, “As selected by the mobile consumer”; and 3) Applications, “As selected by the mobile consumer”. In other words, LiMo primarily serves markets that are run by handset makers and phone system operators. Of course, so do Android and Palm. (Not meaning to pick on LiMo here; they just provide a handy illustration.)

Why, as we enter the second decade of the 21st century, should we be forced to choose only from applications provided by handset makers and mobile phone network operators? Can’t we finally have an open phone marketplace—one where users and developers are free from control by both makers and operators, where you don’t need to get your apps from a “store”, where you can make any app you want, for any purpose you want, without confining your innovations to what phone makers and systems operators allow? Hey, that’s what we’ve had in the PC marketplace for going on 30 years. It’s what we’ve had with the Internet for 15 years. Can’t we have it in the phone marketplace too?

In a word, no.

Or, in two words, not yet.

The problem is that the phone business has never been open or free. It’s one of the most tightly closed and highly regulated businesses on the planet. True, the Internet started cracking open landline phone systems nearly two decades ago, but it’s still just getting started with mobile phone systems and the devices that run on them.

Mobile phones today are a bridge across a chasm between protocols, technologies and business categories. To see the big problem writ small, consider the natural conflicts between SIP (Session Initiation Protocol) and SS7 (Signaling System 7). SIP is a peer-to-peer protocol defined by the IETF (Internet Engineering Task Force). It arose from the obvious need for IP-based calling that looks and feels like what we’re used to with phones, but without all the other hassles, such as centralized switching and billing for everything. SS7 is defined by the ITU-T, or the Telecommunication Standardization Sector of the ITU (International Telecommunications Union), which began as the International Telegraph Union in 1865. SS7 is the heart and soul of every phone system, including all the mobile ones. While the IETF values “rough consensus and running code”, the ITU values top-to-bottom definitional completeness. Yet to various working degrees, SIP and SS7 coexist in our lives, devices and applications.

Trends favor the IETF and the Net, but I wouldn’t bet that way in the short run, which may be a decade or more. We see positive signs with smartphones and handhelds for which telephony is one application among many. But the telephony space is a billing space, and it still rakes in the cash. The Net may be a “world of ends” (**worldofends.com**), but its means include a vastness of phone company cabling, routing and switching. And, most of all, billing.

It is essential to recognize that billing is the core competency of telephony. The Net threatens that. I’m betting the Net will win. But it will be a long, hard fight. And Linux will be right in the middle of it, serving both sides.■

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.

WEB HOSTING BACKED BY FANATICAL SUPPORT®

Hosting is all we do.
And we do it better than anyone else in the world. We've set the standard for **Managed**, **Cloud** and **Email Hosting** with our support, expertise and diverse solutions for customers of all sizes, kinds and needs.

MANAGED HOSTING · CLOUD HOSTING · EMAIL HOSTING

*experience **fanatical support**®*

- Unlimited live support available 24x7x365 from your dedicated team
- Hundreds of IT experts including Red Hat Certified Engineers
- Scalable solutions to efficiently meet growth needs
- On-demand, low-cost computing from The Rackspace Cloud™
- Highest level of protection with an array of security services

rackspace.com/linuxjournal | 1.888.571.8976

rackspace® HOSTING

More GFLOPS, Less WATTS

Intel® Nehalem is here!

Higher Memory Bandwidth with DDR3 and QPI
Clusters and Servers Consume Less Power

Four Servers in a 2U Chassis with all Hot-Swap:
- 1200 Watt 1+1 supply, 12 Drives, and Server Modules!

FasTree™ ConnectX® QDR and DDR InfiniBand Switches and HCAs

Intel Professional Compiler Suite and Cluster Toolkit
- Version 11 with Nehalem Enhancements
- Academic Pricing Available

intel
Channel Partner
Premier
Member 2009

Configure your next Cluster today!
www.microway.com/quickquote

GPU Computing

WhisperStation™
With 1 to 4 Tesla GPUs

Tesla C1060 GPU Performance:
- 1 TFLOPS per GPU
- 4 GB DDR3 per GPU
- 102 GB/Sec Bandwidth
- CUDA SDK

Run MATLAB® on Tesla with "Jacket"

Clusters With Tesla™
S1070 - 4 GPU Servers
- 36 GPUs + 36 CPUs + 24 TB in 24U
- 40 Gbps FasTree™ InfiniBand
- InfiniScope™ Network Monitoring

FREE 15-day trial available at **microway.com**

Microway®
Technology you can count on℠

508-746-7341
microway.com

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