

Marble | HTML5 | NOOKcolor | Ruby on Rails 3 | BlueZ | Android

LINUX JOURNAL

Since 1994: The Original Magazine of the Linux Community
MARCH 2011 | ISSUE 203 | www.linuxjournal.com

REVIEWED:
Barnes & Noble's
NOOKcolor

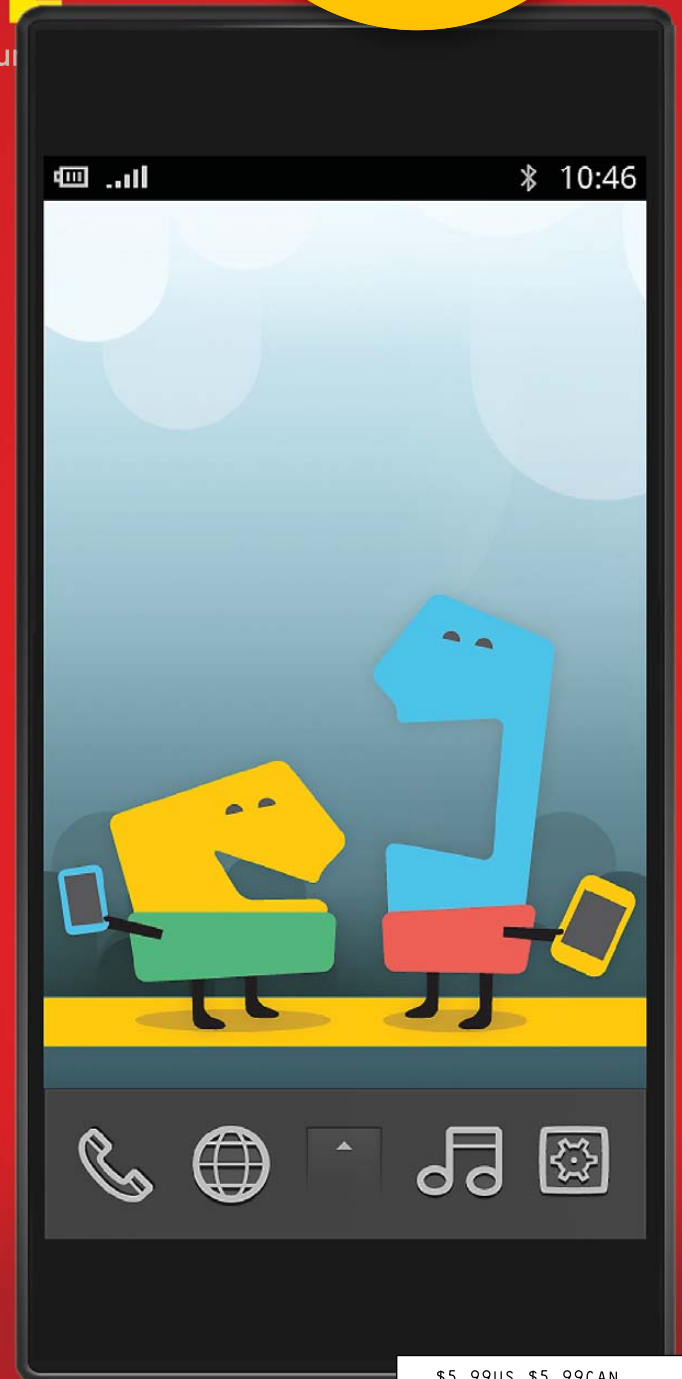
Radical Breeze's
Illumination
Software
Creator

Linux in Your Pocket

WHAT'S NEW IN MeeGo 1.1

Programming
Android with Python

Run Marble on
a Smartphone



PLUS

Write Mobile Apps
with HTML5

Ruby on
Rails 3

Build a
Personal Server



More TFLOPS, Fewer WATTS

Microway delivers the fastest and greenest floating point throughput in history

Enhanced GPU Computing with Tesla Fermi

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

New Processors

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

Configure your next Cluster today!

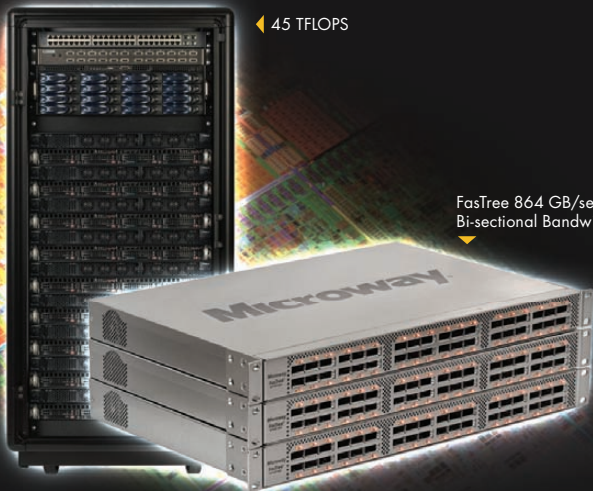
www.microway.com/quickquote
508-746-7341



2.5 TFLOPS

10 TFLOPS

5 TFLOPS



4.5 TFLOPS

FasTree 864 GB/sec
Bi-sectional Bandwidth

FasTree™ QDR InfiniBand Switches and HCAs

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

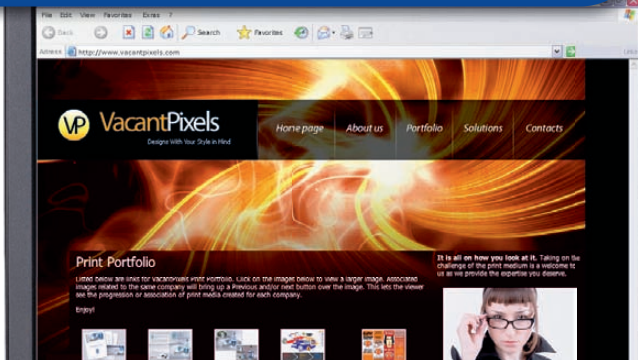
Achieve the Optimal Fabric Design for your Specific MPI Application with ProSim™ Fabric Simulator

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



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

Microway
Technology you can count on™



PROFESSIONAL WEBSITES

As the world's largest web host, we know the developer features you need in a hosting package!



Domains Included

All hosting packages include domains, free for the life of your package.



Unlimited Traffic

Unlimited traffic to all websites in your 1&1 hosting package.



Developer Features

Extensive language support with PHP 5/6 (beta) with Zend Framework and git version management software.



Online Marketing Tools

SEO tools to optimize your website. 1&1 Webstatistics makes it easy to monitor your progress.



Green Data Centers

We're committed to hosting your site with a minimal impact on the environment.

1&1® HOSTING PACKAGES
6 MONTHS
FREE!*
HURRY – OFFER ENDS
2/28/2011!

1&1® BUSINESS PACKAGE:

- 3 Included Domains
- Private Domain Registration
- 250 GB Web Space
- UNLIMITED Traffic
- **NEW:** Version Management Software (git)
- 2,500 E-mail Accounts
- 50 MySQL Database (100 MB)
- 25 FTP Accounts
- E-mail Marketing Tool
- 24/7 Toll-free Customer Support

~~\$9.99~~
per month*

Need more domains?

.info domain only \$0.99 first year*
.com domain only \$4.99 first year*

More special offers available on our website!



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

www.1and1.com

CONTENTS

MARCH 2011
Issue 203

FEATURES

48

Python for Android

SL4A brings scripting languages to the Android platform and provides a working alternative to Java development.

Paul Barry

54

Put the World in Your Pocket with Marble

Free maps, free software. Move freely.

Stuart Jarvis

60

Augmented Reality with HTML5

Just how far can HTML5 be pushed for writing mobile applications?

Rick Rogers

68

Run with MeeGo

An in-depth look at the current status of the MeeGo Project.

Ibrahim Haddad



ON THE COVER

- Reviewed: Barnes & Noble's NOOKcolor, p. 40, and Radical Breeze's Illumination Software Creator, p. 44
- What's New in MeeGo 1.1, p. 68
- Programming Android with Python, p. 48
- Run Marble on a Smartphone, p. 54
- Write Mobile Apps with HTML5, p. 60
- Ruby on Rails 3, p. 22
- Build a Personal Server, p. 32

SPEND LESS. STORE MORE.

Aberdeen's AberNAS Network Attached Storage appliances, featuring the high performance Intel® Xeon® processor 5600 series, deliver maximum efficiency while providing the most storage capacity available today.



DENSITY

Storage from 2TB to 132TB
in a single appliance

RELIABILITY

RAID 6, Automatic RAID rebuild,
Snapshot Data Recovery and
Industry Leading 5-Year Warranty

SCALABILITY

Expand to beyond 1 Petabyte
via XDAS and JBOD units

Who gives you the best bang for the buck?

ABERNAS 160 SERIES



1U NAS up to 8TB

- Quad-Core Intel Xeon Processor
 - 3GB 1333MHz DDR3 Memory
- Available with Windows or Linux OS
- 2TB Starting at..... **\$2,795**
8TB Starting at..... **\$3,795**

ABERNAS 260 SERIES



2U NAS up to 16TB

- Quad-Core Intel Xeon Processor
 - 3GB 1333MHz DDR3 Memory
- Available with Windows or Linux OS
- 4TB Starting at..... **\$4,295**
16TB Starting at..... **\$6,295**

ABERNAS 290 SERIES



2U NAS up to 24TB

- Quad-Core Intel Xeon Processor
 - 3GB 1333MHz DDR3 Memory
- Available with Windows or Linux OS
- 12TB Starting at..... **\$5,995**
24TB Starting at..... **\$7,995**

ABERNAS 360 SERIES



3U NAS up to 32TB

- 2x Quad-Core Intel Xeon CPUs
 - 6GB 1333MHz DDR3 Memory
- Available with Windows or Linux OS
- 16TB Starting at..... **\$7,995**
32TB Starting at..... **\$10,995**

ABERNAS 560 SERIES



5U NAS up to 48TB

- 2x Quad-Core Intel Xeon CPUs
 - 6GB 1333MHz DDR3 Memory
- Available with Windows or Linux OS
- 24TB Starting at..... **\$10,495**
48TB Starting at..... **\$14,495**

ABERNAS 860 SERIES



8U NAS up to 100TB

- 2x Quad-Core Intel Xeon CPUs
 - 6GB 1333MHz DDR3 Memory
- Available with Windows or Linux OS
- 50TB Starting at..... **\$18,495**
100TB Starting at..... **\$26,995**

ABERNAS 8960 SERIES



8U NAS up to 132TB

- 2x Quad-Core Intel Xeon CPUs
 - 6GB 1333MHz DDR3 Memory
- Available with Windows or Linux OS
- 128TB Starting at..... **\$33,995**
132TB Starting at..... **\$34,995**



**Powerful.
Intelligent.**

CONTENTS

MARCH 2011

Issue 203

COLUMNS

- 22** Reuven M. Lerner's At the Forge
Ruby on Rails 3
- 26** Dave Taylor's Work the Shell
Making a *Mad Libs* Generator
- 28** Mick Bauer's Paranoid Penguin
Interview with a Ninja, Part I
- 32** Kyle Rankin's Hack and /
Your Own Personal Server: the Network
- 80** Doc Searls' EOF
Freeing Up Level 7

REVIEWS

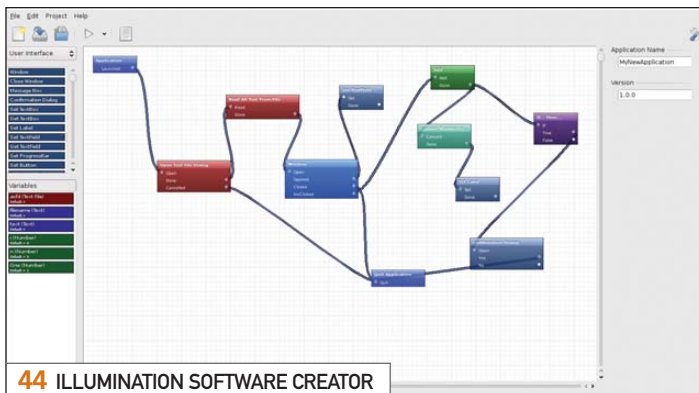
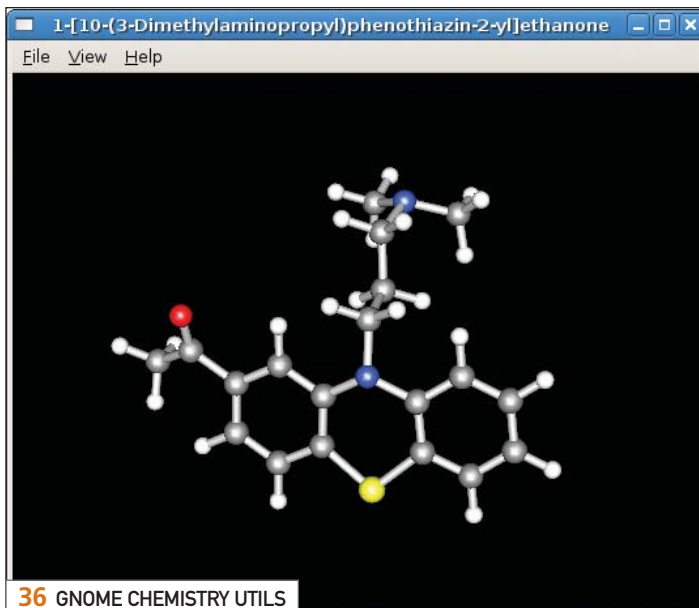
- 40** Barnes & Noble's NOOKcolor
Bill Childers
- 44** Radical Breeze's Illumination
Software Creator
Mike Diehl

INDEPTH

- 75** Is Your Personal Area Network Giving You the BlueZ?
Bridging or routing, which is best for your Bluetooth Personal Area Network? You might not have any choice!
Chuck Elliot

IN EVERY ISSUE

- 8** Current_Issue.tar.gz
- 10** Letters
- 14** UPFRONT
- 34** New Products
- 36** New Projects
- 65** Advertisers Index
- 79** Marketplace



USPS *LINUX JOURNAL* (ISSN 1075-3583) (USPS 12854) is published monthly by Belltown Media, Inc., 2121 Sage Road, Ste. 310, Houston, TX 77056 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 Pitney Bowes, P.O. Box 25542, London, ON N6C 6B2

MPLS for the masses

\$39⁹⁵



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

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

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

Features:

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

Benefits:

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

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

LINUX JOURNAL™

Since 1994: The Original Magazine of the Linux Community

**DIGITAL EDITION
NOW AVAILABLE!**

Read it first

Get the latest issue before it
hits the newsstand

Keyword searchable

Find a topic or name
in seconds

Paperless archives

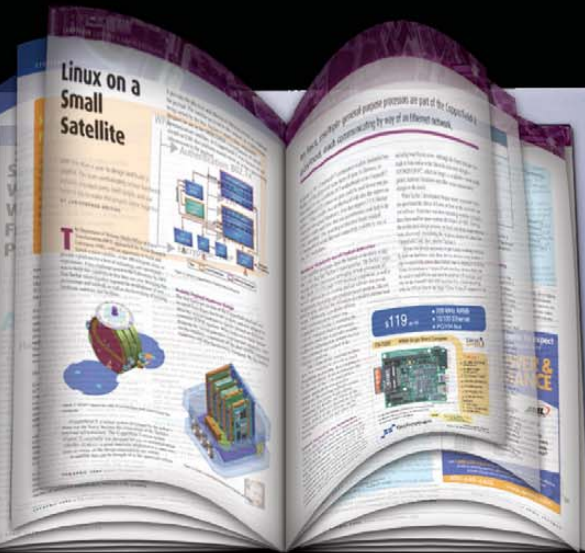
Download to your computer for
convenient offline reading

Same great magazine

Read each issue in
high-quality PDF

Try a Sample Issue!

www.linuxjournal.com/DLISSUE



LINUX JOURNAL

Executive Editor Jill Franklin
jill@linuxjournal.com

Senior Editor Doc Searls
doc@linuxjournal.com

Associate Editor Shawn Powers
shawn@linuxjournal.com

Art Director Garrick Antikajian
garrick@linuxjournal.com

Products Editor James Gray
newproducts@linuxjournal.com

Editor Emeritus Don Marti
dmarti@linuxjournal.com

Technical Editor Michael Baxter
mab@cruzio.com

Senior Columnist Reuven Lerner
reuven@lerner.co.il

Security Editor Mick Bauer
mick@visi.com

Hack Editor Kyle Rankin
lj@greenfly.net

Virtual Editor Bill Childers
bill.childers@linuxjournal.com

Contributing Editors

Ibrahim Haddad • Robert Love • Zack Brown • Dave Phillips • Marco Fioretti • Ludovic Marcotte
Paul Barry • Paul McKenney • Dave Taylor • Dirk Elmendorf • Justin Ryan

Proofreader Geri Gale

Publisher Carlie Fairchild
publisher@linuxjournal.com

General Manager Rebecca Cassity
rebecca@linuxjournal.com

Senior 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

Editorial Advisory Panel

Brad Abram Baillio • Nick Baronian • Hari Boukis • Steve Case
Kalyana Krishna Chadalavada • Brian Conner • Caleb S. Cullen • Keir Davis
Michael Eager • Nick Faltsy • Dennis Franklin Frey • Alicia Gibb
Victor Gregorio • Philip Jacob • Jay Kruiuzenga • David A. Lane
Steve Marquez • Dave McAllister • Carson McDonald • Craig Oda
Jeffrey D. Parent • Charnell Pugsley • Thomas Quinlan • Mike Roberts
Kristin Shoemaker • Chris D. Stark • Patrick Swartz • James Walker

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.



Developing for the iPhone or iPad?

Register Early AND SAVE!

Choose from Over 50 Classes!

- iPhone/iPad Developer Essentials
- Enterprise Apps Management
- Spotlight on iPad

Attend iPhone/iPad DevCon East



April 4-6, 2011 BOSTON

Hyatt Regency Cambridge

Learn from our expert speakers

Our top-notch faculty are the most established apps developers and marketers in their business. They're not talking heads or corporate drones: They're hands-on experts with real-world apps experience. They know what it takes to succeed in the mobile marketplace, and at iPhone/iPad DevCon, they'll share their knowledge, insights and best practices with you to give you a great technical conference experience.

"This conference will get you up to speed quickly. Good content, keynotes, and the support team was excellent."

—Dr. Joy Sargis, Manager, Development Unit, Geisel Library, UCSD

Registration Now Open!

www.iphonedevcon.com



Join Us for...
 More Classes!
 More Learning!
 More Networking!

Sponsored By



iPhone/iPad DevCon™ is a trademark of BZ Media LLC. iPhone® and iPad® are registered trademarks of Apple Inc.



SHAWN POWERS

A Chicken in Every Pot and Linux in Every Pocket

The problem with mind-control devices is they generally work only in very close proximity to the host. In our ongoing quest to take over the world, we've come up with a way to keep our open-source mind probes tethered to our victims at all times: embedding Linux into the phones of the populace. With the help of Google and its vow against being evil (tee-hee!), we've taken over a significant percentage of the market. Soon, our free ideals and open standards will take over the world! MWAHAHAHA!

Okay, perhaps I should avoid watching vintage sci-fi films right before writing my column. Nonetheless, this month is an exciting issue. Our focus is "Linux in Your Pocket". Whether you're a fan of Android, Maemo, MeeGo or Moblin, the mobile computer market is busting at the seams with tablets, phones and Netbook crossovers.

Having armies of Android minions is pointless if you can't program them to bend to your will, and Paul Barry shows us that Java isn't the only way to make Android apps. Python is a very popular scripting language, and Python for Android opens up mobile device programming for a whole bunch of people. If Android isn't your cup of tea, you might be interested in MeeGo 1.1. Ibrahim Haddad describes the ins and outs of the project that sprouted from Intel's Moblin and Nokia's Maemo. Some people love it, and some people hate it, but it works on both Netbooks and smartphones, so trying it should be pretty simple.

Granted, merely running Linux on mobile devices isn't a recipe for world domination, but Stuart Jarvis explains how to keep tabs on the world itself with Marble. The KDE program, which is a virtual globe, has been ported to mobile devices that support Nokia's Qt framework. Although spinning a globe on your handheld certainly is entertaining, because it's a digital globe, the features are a bit more robust. We may have gotten ants dizzy on the globe in geography class back in school; however, with Marble, you get a smart globe capable of mapping destinations and planning trips. If you add augmented reality to the mix, you can take a complete vacation without ever taking your eyes off your phone! Rick Rogers shows how to use augmented reality with HTML5 for an experience reminiscent of the heads-up display used in every first-person-shooter game ever made. I don't recommend walking around with your cell phone strapped to your face, but it does

make quick order of unfamiliar surroundings.

If by some odd chance you're not a mad scientist bent on creating pocket-sized mind-control devices, fear not. This issue has many other great articles and columns to scratch that Linux itch. Kyle Rankin starts a series on personal servers, proving that everyone can be sysadmins, even if they never leave their home offices. Whether you're looking to host a local file server or manage Internet services from your basement, Kyle's series is an exciting one for those interested in taking charge of their data. If you do host your own server, you'll want to read Mick Bauer's column as well—part I of his interview with a system cracker.

If programming is more up your alley, check out Reuven M. Lerner's column this month. Now that Ruby on Rails is at version 3, you probably want to know what that means if you're a Ruby programmer. Reuven describes what's new, what's the same, and why you might care either way. Additionally, Dave Taylor teaches us more about shell scripting, again with a really fun project. If you've ever played *Mad Libs*, you know it can be fun. In fact, if you follow along with Dave, you might not even realize you're learning as you go, because *Mad Libs* can be wildly entertaining.

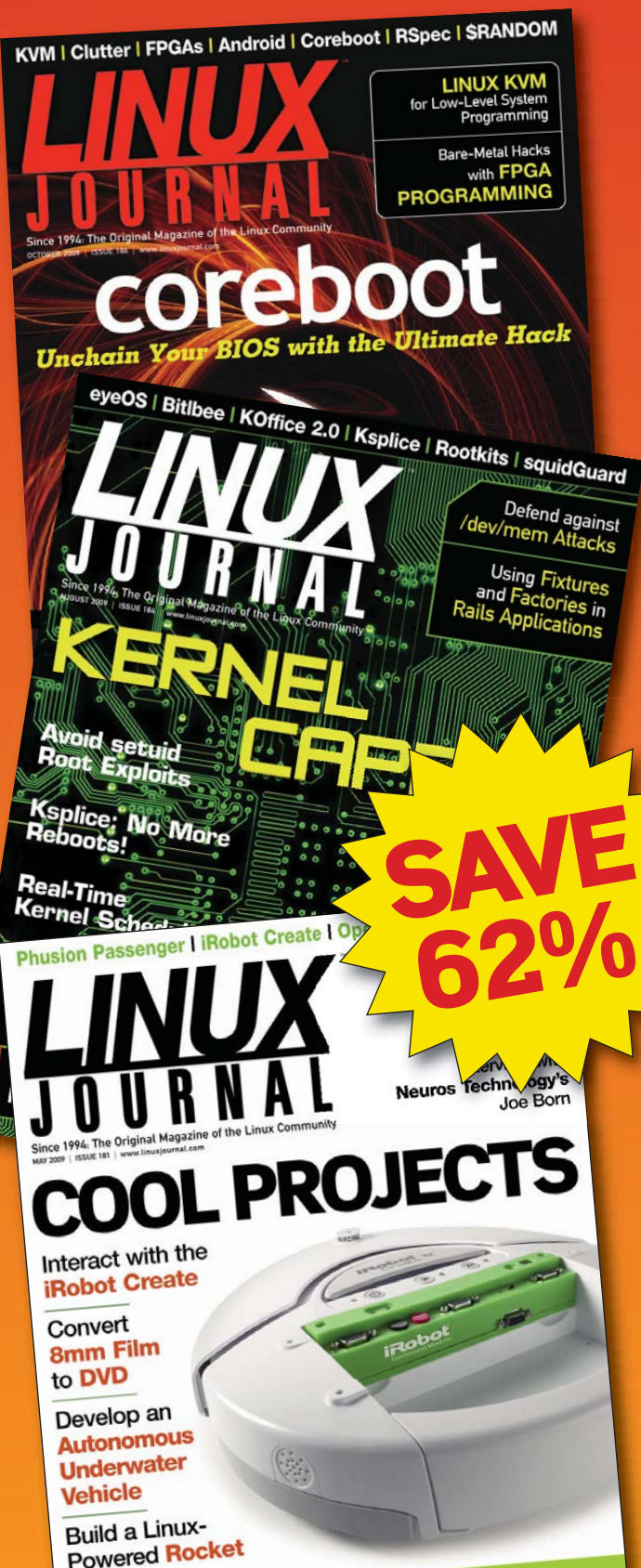
We've also got product reviews, like Bill Childer's review of the NOOKcolor (which he totally hacked and tells us how to do the same). It's a great way to get a decent tablet at a cheap price, and also read some books along the way. Mike Diehl follows, showing application creation with Radical Breeze's Illumination Software Creator. For those of you who don't actually like to write code, Mike has just the programming framework for you.

As for me, I'm going to read Chuck Elliot's article on programming Bluetooth devices. I see half the population walking around with Bluetooth earpieces attached to their heads. If I can just reprogram them to emit subliminal messages to their hosts, I can have a zombie army in no time! MWAHAHAHA!

Oh, right. Turn off the late night sci-fi and get this issue in the mail. Done and done. Enjoy! ■

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.

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



SAVE 62%

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

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

SUBSCRIBE NOW AT:
WWW.LINUXJOURNAL.COM/SUBSCRIBE

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

letters



Well Done!

I have only had broadband Internet recently, so am still discovering the best places therein. Chance brought me to your site today, and that was it. Hooked. Not only is the content interesting (if often essentially a pointer), but the site itself is excellent. Clean, *legible*, well structured—congrats.

--
Lewis Smith

We're glad to have you! There are many people in my area here in northern Michigan who still have dial-up. Once you have broadband, it's hard to imagine life without it! Have fun.—Ed.

Slackware Linux?

I've been using Linux for years now and just recently obtained a subscription to *Linux Journal*—great magazine! I started out using Ubuntu, but I have advanced my Linux arsenal and have been using Slackware for almost over a year! It's definitely not for the faint of heart, but it's a solid OS and very true to the Linux spirit. I wanted to suggest *LJ* possibly include more information on this Guru-targeted OS for some of us who are less fortunate! Regardless, keep up the good work!

--
Michael Griffith

We'll see what we can do. Slackware truly is an awesome distribution, but like you

said, it's not for the faint of heart. I can't promise we'll have more Slackware-specific stuff, but we'll try to remember Slackware users in our articles.—Ed.

Two Minor Corrections

I just wanted to write in to make a couple minor corrections that may have confused some beginners.

The first one is in Kyle Rankin's review of the Chinavision Pico Projector in the November 2010 issue. He stated that port 21 was open in the Linux operating system of the device and wondered if telnet actually had been enabled. Port 21 is actually for FTP; telnet is port 23. This could have been a typo or just an oversight, but I wanted to clear up any confusion readers may have had.

In Part IV of Mick Bauer's "Building a Transparent Firewall with Linux" series in the December 2010 issue, he states that iptables ignores inter-VLAN traffic and describes that as being traffic between different ports within the same VLAN. Traffic within the same VLAN is actually intra-VLAN rather than inter-VLAN. This is similar to the difference between an intranet and an Internet, of course.

Again, these are minor and pale in comparison to the enormous level of great content *LJ* publishes every month. Keep up the good work.

--
Brandon McCombs

Generating Turn-by-Turn Driving Directions

Today I got a chance to mess around with Dave Taylor's script in the December 2010 issue. Although I'm new to using curl, sed and lynx, the driving directions application in the article gave me some leads to study while using the Bash shell.

Thanks very much for a such a nifty script. After customizing the script as suggested by Dave, I've got it accessible from a directory structure I use for my scripts, so it's now a great addition to my listings. My other small change to the script was to send the output to a desktop.txt file on my desktop, so I could print it out before leaving the house en route to a new customer site.

The code in the article that starts with curl --silent, I finished up on the last section with the following change after the last \:

```
lynx -dump -stdin > $HOME/Desktop/directions.txt
```

Then I added a line that gets me set to print the listing sent by the script:

```
gedit $HOME/Desktop/directions.txt
```

--
Dave Mawdsley

Dave Taylor replies: Thanks for the feedback, Dave. It's always nice to hear that people have enjoyed some of my mad-scientist hacking.

Back to School for Linux

I've been wanting to go back to school for quite some time and complete my degree. As I've recently become interested in Linux and open source, and I'd like to be able to leverage my education with my love of open source—or at least open standards.

The only problem is that the more I look, the more I find degree programs centered around .Net and Microsoft operating systems. Are there any schools that offer UNIX- or (even better) Linux-centered curricula?

As I'm currently already established with family, friends and work, it would be incredible if I could get a great education with the flexibility to attend remotely. Is such a thing in existence now? Thanks for your help!

--
M. Miller

It's been quite a few years since I was in college myself, but my guess is that Linux-specific degrees are pretty rare. My recommendation would be to look toward programming, networking or Web development. They're not Linux-specific, but you'd be able to specialize within the field.—Ed.

Dave Taylor Is Doing a Good Job!

In the January issue of *LJ*, there was a letter from a reader that basically tore into Dave Taylor. I just wanted to say that I think Dave Taylor is doing a good job. When it comes to computers, there is always more than one way to do something.

There is the “Good Way”, the “Better Way” and the “Wrong Way”.

There may be more efficient ways of doing things than the ways that Dave shows in his articles, but that doesn't make them wrong. I think it would have been fine for the reader to point out better ways of doing things. But, the letter came across as just mean.

As someone who does many Bash tutorials myself, I know that you can have a 100 people like your tutorial, but it takes only one mean comment to ruin your day. So, to counteract the bad day that read may have given Dave with his harsh letter, I would like to give Dave some encouraging words.

Dave, you do a good job. I look forward to your articles each month. I thank you for what you do, and I'm sure many readers would agree with me.

--
Kristofer Occhipinti

Netflix Streaming

In the December 2010 issue, a reader asks for help watching Netflix streaming movies from Linux. I found a way to do that.

The problem is the Microsoft DRM, which is difficult to get around, but you can proxy it. By proxy, I mean you can use a Windows-intermediary computer. With a Windows PC running PlayOn software from MediaMall, you can make Netflix streaming video available to a Linux box. I describe this setup for watching Netflix with VLC on Linux in my blog (www.innate-ideas.net), but this method also works for watching with XBMC or other UPnP/DLNA-aware software under Linux. Hope that helps.

--
Earl

Thanks. You are correct, PlayOn (although not free) works fairly well, but unfortunately, it still requires Windows to work. Granted, you don't need to be sitting at the Windows computer if you use PlayOn, but it's still there. It's very frustrating that Netflix insists on using Microsoft's Silverlight DRM. Thanks for the recommendation. For those folks who don't mind a Windows install hiding in a closet somewhere, it's a great way to watch streaming video on Linux.—Ed.

More Ways to Count Files

Reading the letter “Another Way to Count Files” in the January 2010 *Linux Journal*, I find that Michael Eager's method improves on Dave Taylor's by reducing the number of external programs called (to a round zero) and also by actually removing a bug (introducing another though).

Dave Taylor's script does not handle directories correctly as the `ls` command would give the contents of matching directories instead of the names. This could easily be remedied though, by using `ls -ld` instead of `ls -l`.

But correcting that, Michael Eager introduces another problem instead. His script cannot handle files and directories with spaces in the names. I know spaces in names are frowned upon by die-hard Linux/*nix people, but despite being an old-hand myself, I use them regularly. I find that the Linux/*nix community probably should be more aware of this problem, and maybe Dave Taylor could focus one of his columns on this problem.

A solution that uses no external calls and without the pitfall of Michael Eager's (my own requiring Bash though) is:

```
#!/bin/bash
...
Matches=0
for Match in $Pat*
do
    if [ "$Match" != "$Pat*" ]; then
        Matches=$((Matches + 1))
    fi
done
```

I am aware that my solution could be relatively costly if there are a lot of matching files/directories. I guess there is still more than one way to skin a cat.

--
Torben Rybner

January 2011 Issue

To Shawn Powers: things you helped me with from this month's issue. I'm now listening to Pithos—most excellent, thank you! I have used BackupPC for about two years now—great to see it mentioned.

Tell Mick Bauer that his firewall series of articles is great. I thought about a Linksys WRT54GL router—might do that. I just stopped using IPCop (I think

that was last month's article), and now I'm using SmoothWall 3.0 (just to get a 2.6 kernel).

I've been using the Clonezilla live CD for about four years now [see Jeremiah Bowling's article “Clonezilla: Build, Clone, Repeat” in the January 2011 issue]. It has saved my behind more times than I can remember. It is an outstanding product and is developing well.

Keep up the great work everybody! Oh, and, let's see, this is issue 201, so let's see at least 201 more issues. That means, 13 issues per year, so 15 years from now, what will we be reading? Linux OS in world domination and some upstart OS trying to knock Linux down a notch! (Maybe Bill's nephew, trying to save what's left of MS. We can only hope.)

--
Bob Wooden

I'd like to keep my childhood fantasies alive and assume in 15 years we'll all be living in space. Linux Journal will be where everyone goes for current events and information, and our flying cars will all be running embedded Linux. Oh, and my hairline will stop receding!

Seriously though, thanks for the kind words. We love what we do here, and it's great to see others enjoy the fruits of our labor.—Shawn.

Linux for Science Column

Joey Bernard's new column, “Linux for Science”, is one of the reasons I keep subscribing to *Linux Journal*. Yes, I enjoy the system administration hints and industry news, but what makes Linux so wonderful is the thought that I, or anyone else, can use the same tools in basement science experiments that are used in research labs. I am eagerly awaiting future articles from Mr Bernard.

--
Kwan Lowe

Joey Bernard replies: Thank you very much for the encouragement. My day job involves helping university researchers do computational science, and I'm always amazed at all the things you can do. If there are any particular subjects or programs you would like to see covered, please let me know.

[LETTERS]

Welsh Command Prompt (PS1)

As a lover of Linux and the Welsh language, I thought it might be useful to post a Perl script that will give a Welsh language-based prompt. Edit the `.bashrc` to the following:

```
PS1='`/home/ed/cymraeg.pl`: `pwd` : '
```

```
cymraeg.pl:
#!/usr/bin/perl -w

%mis=(Jan=>"Ionawr",Feb=>"Chwefror",Mar=>"Mawrth",
->Apr=>"Ebrill",May=>"Mai",Jun=>"Mehefin",
->Jul=>"Gorffennaf",Aug=>"Awst",Sep=>"Medi",
->Oct=>"Hydref",Nov=>"Tachwedd",Dec=>"Rhagfyr");
%dydd =
%(Mon=>"Llun",Tue=>"Mawrth",Wed=>"Mercher",Thu=>"Iau",
->Fri=>"Gwener",Sat=>"Sadwrn",Sun=>"Sul");

chop ($mnth = `date +%b` , $day = `date +%a` ,
->$rhyf = `date +%d`);
foreach $mmonth(keys %mis) {if ($mmonth eq $mnt)
->{ $fy_mis = $mis{$mmonth}; }}
  foreach $dday(keys %dydd) {if ($dday eq $day)
->{ $fy_dydd = $dydd{$dday}; }}
print "Dydd_$fy_dydd-$rhyf-Mis_$fy_mis";
```

Assuming everything works okay, you should get the following:

```
Dydd_Mawrth-02-Mis_Tachwedd:/home/ed:
```

For anybody who is a Welsh speaker and spends considerable time at a `*nix` prompt, this is, for me at least, a cathartic experience.

--

Ed Williams

I'm always worried when we print things I don't understand. The little-geek-that-got-bullied-in-school part of me always assumes someone's making fun of me. Although I don't think your command prompt is calling me four-eyes, I'll leave it to our Welsh-speaking readers to figure that out.—Ed.

Networking on the Command Line

It was nice to see a command-line tool for network configuration (see Joey Bernard's Upfront piece in the December 2010 issue). Please do more! I do have one thing to add. I am currently running CentOS 5.4, and the configuration files Joey mentioned in his article are different. Below is an example with the path. Also, note the difference in the auto setting. There is a configuration file for each network interface:

```
[root@CentOs01:/etc/sysconfig/network-scripts#] cat ifcfg-eth0
# Advanced Micro Devices [AMD] 79c970 [PCnet32 LANCE]
DEVICE=eth0
BOOTPROTO=static
BROADCAST=172.16.175.255
HWADDR=00:0C:29:77:D0:7E
IPADDR=172.16.162.3
NETMASK=255.255.240.0
NETWORK=172.16.160.0
ONBOOT=yes
```

By the way, I'm really enjoying the 200th issue.

--

Mickey Craven

LINUX JOURNAL

At Your Service

MAGAZINE

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

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

LETTERS TO THE EDITOR: We welcome your letters and encourage you to submit them at www.linuxjournal.com/contact or mail them to Linux Journal, PO Box 980985, Houston, TX 77098 USA. Letters may be edited for space and clarity.

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

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.



DRUPALCON CHICAGO

MARCH 7-10 2011

Drupal is the industry-leading open source content management platform that's used to power millions of websites around the world.

This March 7-10, thousands of Web developers, designers, businesspeople, and everyday citizens will gather for DrupalCon Chicago, a one-of-a-kind conference that no Web professional can afford to miss!

Find out more at <http://chicago2011.drupal.org/>



MARCH 7-10, 2011 • SHERATON HOTEL & TOWERS • CHICAGO, ILLINOIS

diff -u

WHAT'S NEW IN KERNEL DEVELOPMENT

John Stultz submitted code to enhance the **real-time clock**. He didn't like the fact that if a process wanted to set an alarm, it would have to take a lock on the RTC and poll it, waiting for that alarm to go off. His new code created an intermediate layer that would keep track of alarms set by any user code and feed the alarms back to the programs that had set them, as they occurred. If his patch is accepted, it could be good news for user programs that no longer will have to deal with the complexities of holding and freeing an RTC lock.

John also noticed that the **Android RTC alarm driver** had a new approach to the RTC that he felt deserved to be included in the main kernel tree. But, because the Android driver depended on other parts of the Android system that were not included in the official kernel, John rewrote the driver to use native kernel interfaces and submitted it that way. The basic value of the Android driver, as he saw it, was that it gave the user a simpler interface, at the cost of losing some subtle abilities that didn't seem very useful anyway. So as far as John could see, it was a pure win. It'll be interesting to see whether the Android system adapts to use John's version of the driver as well, or whether it will continue to maintain its own version independently.

Vernon Mauery submitted code to support **IBM Premium Real-Time Mode (PRTM)** on supporting x86 hardware. This powerful operating mode can give user programs a wholesale control over the system's hardware, which normally is tightly regulated and secured by the kernel. PRTM has the potential to be a devastating security hole, but it also can provide very low latency action to applications that require real-time performance. And, it can provide very useful features to debuggers and other meta-

applications. Vernon's code provides a SysFS interface to enter and leave PRTM at will. I'm guessing there also will be security features to prevent applications from doing this unless they are absolutely trusted.

Documentation is always nice to see, and **Andres Salomon** recently submitted some for the **staging tree**. The staging tree exists in the main kernel source, under `drivers/staging`, and it acts as a repository for incomplete drivers that aren't ready to be relied on completely. In the old days, people wanting to test these drivers would have to download the driver's patch independently, apply it to their own copy of the kernel, and compile and install that kernel. This barrier to entry was sufficient to keep all but the most enthusiastic testers from ever testing a driver before it was included in the official tree. The staging tree allows any user to test a driver very easily, just by enabling it during the configuration process prior to compiling the kernel. This opens up testing to an immense and seething horde of potential users. The result is that drivers receive far better testing before being incorporated into the main source tree. Users are protected from potentially damaging code in these drivers, because they simply cannot enable drivers in the staging tree when they compile the kernel.

Of course, as with all aspects of kernel development, the rules constantly are being revised, so that the kernel development process can really be seen as a set of algorithms developed in an evolutionary way that has the ultimate goal of making the best possible behaviors also the most natural-seeming for any given kernel contributor. As with all evolutionary development, any given snapshot presents only the imperfect balance of that moment, but it's so great to watch.

—ZACK BROWN



Linux Journal Archive

The newly updated 1994–2010 *Linux Journal* Archive disc is here! In easy-to-use HTML format, the fully searchable, space-saving archive offers immediate access to an essential resource for the Linux enthusiast: *Linux Journal*. The archive includes all 200 issues of *Linux Journal*, from the premiere issue in March 1994 through December 2010.

- **Regular price:** \$34.95
- **Sale price:** \$29.71 (15% off)
- **Coupon code:** ACDNEW

Sale ends March 31, 2011.

They Said It

It is the framework which changes with each new technology and not just the picture within the frame.

—**Marshall McLuhan**

A computer is like an Old Testament god, with a lot of rules and no mercy.

—**Joseph Campbell**

They have computers, and they may have other weapons of mass destruction.

—**Janet Reno**

The technologies which have had the most profound effects on human life are usually simple.

—**Freeman Dyson**

All technology should be assumed guilty until proven innocent.

—**David Brower**

The newly updated **LINUX JOURNAL ARCHIVE** is here!



**NOW ON
SALE:
SAVE 15%***

The archive includes **all 200 issues** of *Linux Journal*, from the premiere issue in March 1994 through December 2010. In easy-to-use HTML format, the fully searchable, space-saving archive offers immediate access to an essential resource for the Linux enthusiast: *Linux Journal*.

*Use coupon code **ACDNEW** at www.linuxjournalstore.com, now through March 31, 2011, and save 15% off the standard retail price of \$34.95.

www.LinuxJournalStore.com

GUI from Afar

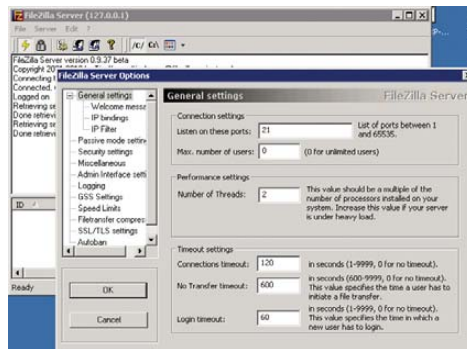
For most configurations, I prefer to use command-line tools. This is especially true on headless servers. There are a few things, however, that I find easier to configure with GUI tools. Webmin does a fine job for some of those things, but sometimes the native GUI tools are really nice. One good example is virt-manager. Although it's possible to run virt-manager on a workstation separate from the server running KVM, sometimes installing virt-manager also installs kernel modules that are incompatible with other virtual systems.

In my particular scenario, I like to run VirtualBox on my workstation, but KVM on my server. Although it is possible to run virt-manager on a workstation with VirtualBox installed, it can get messy with conflicting kernel modules. That's where `ssh -X` comes into play. Just install virt-manager on the server, run it remotely with a simple `ssh -X user@server virt-manager`, and the GUI program starts up right on your current desktop!

It's not a new trick, but it's one I find myself using often. If you haven't considered installing GUI tools on your headless server, just because it seemed silly, don't forget the GUI power of SSH. It might make your life a little easier—or at least more colorful.

—SHAWN POWERS

NON-LINUX FOSS



Almost every operating system has an FTP client, but they often are command line only or kludgy at best. If you're looking for a solid FTP (or SSH) client, FileZilla is a cross-platform program that does a few things and does them very well. Whether you need to transfer files over FTP, FTP/TLS or even SSH, it has a simple interface and a reliable code base.

The folks at the FileZilla Project went one step further, however, and provided a free FTP server program for Windows as well. FileZilla's FTP server fills a much-needed gap in the desktop Windows operating system and provides extensive customization. It runs in the background, so there's no need to have a window open all the time in order to have an FTP server running. If you need a reliable FTP client for Windows, OS X or Linux, check out FileZilla's client program. If you need an FTP server for Windows, be sure to check out the server program: filezilla-project.org.

—SHAWN POWERS

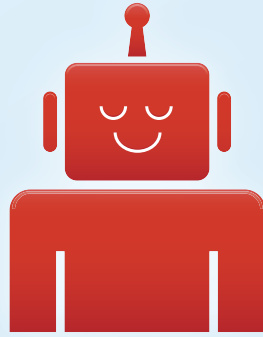
Is that LinuxJournal.com in your pocket, or are you just happy to see me?

I'm sure many of you carry around your monthly *Linux Journal* with you everywhere. Sure, the glossy paper does tend to get a little beaten up, but having important refer-



ence material with you at all times is worth a bit of wear and tear. On the other hand, we know some of you take meticulous care of your issues of *Linux Journal*, placing them gingerly on your coffee table for all your guests to marvel at. Either is perfectly fine with us, but we have a portability solution that will work for everyone. Visit LinuxJournal.com from your mobile device, and you'll notice you are now viewing a simplified version of the site that is optimized for a mobile experience. We hope this proves useful to you while you are off on all your geeky adventures, or even when you are just sitting around the dinner table and need to show your friend that awesome blog post by Shawn Powers you read last week. Happy surfing!

—KATHERINE DRUCKMAN



Lullabot™

Learn Drupal & jQuery

FROM THE COMFORT OF
YOUR LIVING ROOM



The Lullabot Learning Series includes everything you need to become a Drupal & jQuery expert from the comfort of your living room! The videos are available in both DVD format and high-definition video download.

Purchase the videos at <http://store.lullabot.com>

Linux on a Fingernail

This issue of *Linux Journal* is all about how to get Linux in your pocket. In this article, I go one better and tell you how to get Linux on your fingernail. Now, before you get too excited, I won't be discussing some new nano-computer being used by James Bond, unfortunately. Instead, I discuss how to put Linux on a micro-SD card (or any other USB drive, for that matter). Using this, you can run Linux on any machine that can boot off a USB device.

One of the first utilities to receive widespread attention and use is UNetbootin (unetbootin.sourceforge.net). This application is available under both Linux and Windows. It has built-in support for downloading and installing several Linux distributions, including Ubuntu, Fedora, Debian, PCLinuxOS, Linux Mint, Sabayon Linux, openSUSE, Gentoo, Arch Linux, MEPIS and many others.

UNetbootin also has the ability to load several different system utilities, such as the following:

- Parted Magic: a partition manager that can resize, repair, back up and restore partitions.
- SystemRescueCD: a system repair, backup and recovery tool.
- Super Grub Disk: a boot utility to restore and repair GRUB installations.
- Backtrack: a utility for network analysis and penetration testing.

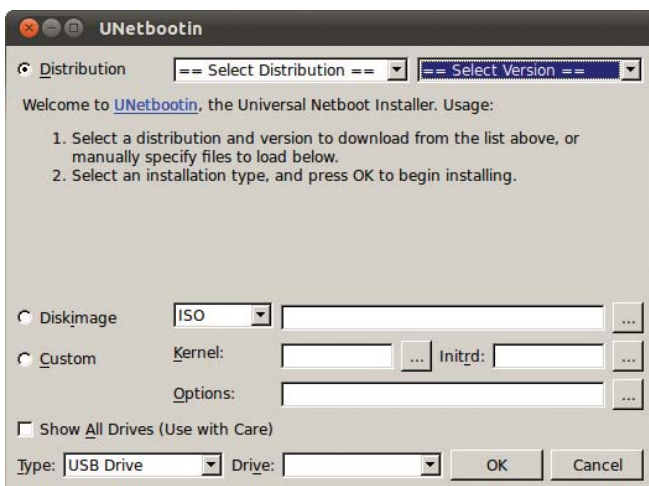


Figure 1. UNetbootin

- Ophcrack: a utility to recover Windows passwords.
- Smart Boot Manager: boots off of CD-ROM and floppy drives on machines with a faulty BIOS.
- FreeDOS: an open-source DOS to allow you to run BIOS flash utilities or just plain-old DOS.

UNetbootin can download the correct ISO image for all of these distributions automatically, or you can download (or create) your own ISO image and hand the filename to UNetbootin. In either case, the next step is to extract all of the files from the image so they can be copied to the USB drive. The USB drive you want to use needs to be formatted with a filesystem already on it. Once the files all have been extracted, UNetbootin uses some heuristics to figure out where the kernel and initrd files are hiding and places them into `/ubnkern` and `/ubninit`. It then goes through the boot configuration files from the ISO to try to figure out what boot options need to be set on the USB drive. Once it figures that out, it puts those options into the configuration file `/syslinux.cfg`. Then, UNetbootin uses Syslinux to make the USB drive bootable, and you should be good to go.

Another utility that has started garnering a lot of attention is `usb-creator` (<https://launchpad.net/usb-creator>).

This program is the official tool to create live USB versions of Ubuntu live CDs. This utility started with version 8.04 of Ubuntu. A KDE front end showed up in version 8.10, and a Windows version currently is in the works. This utility can do the same sort of work as UNetbootin. It has the built-in ability to create an Ubuntu live



Figure 2. `usb-creator`

USB, but that's not all. Like UNetbootin, `usb-creator` can take an arbitrary ISO image and copy that onto your USB drive. It even can take a CD-ROM from your CD drive and copy its contents over to your USB drive. It's a simple matter of selecting the source and the destination, and then running.

One big advantage of `usb-creator` is its ability to create a persistent live USB for you very easily. Any extra space on your USB drive, above and beyond what is required for the OS files, can be used as writable space for persistent files. This means any changes you make to your system will be written to the USB drive. You even have the option of clearing the persistent space on shutdown. That way, you can use your USB drive as a complete operating system, exactly as if it were on your hard drive. And, there you go, Linux on a thumbnail.

All of these techniques require you to have either a live CD or an ISO image of a live CD. But what if you want to go a bit more low-level, a bit more from scratch? One of the tools available to you is Syslinux (syslinux.zytor.com/wiki/index.php/The_Syslinux_Project). Syslinux actually is a suite of different programs that provide for booting from many different media, including:

- syslinux: booting from FAT filesystems.
- pxelinux: network booting.

- isolinux: bootable “El Torito” CD-ROMs.
- extlinux: booting from ext2/ext3/ext4 or btrfs filesystems.
- memdisk: a tool to boot legacy OSes from nontraditional media.

Sysexlinx installs into the boot sector of your device and puts a copy of the file LDlinux.sys into the root directory. It then loads the kernel and other OS files from the actual filesystem. Because the filesystem is just a simple FAT filesystem, all the files, including the kernel, can be manipulated using standard DOS tools. By default, Sysexlinx assumes the kernel is in the file named LINUX on the boot disk. This default can be changed in the config file. If you hold down the Shift or Alt keys during bootup, Sysexlinx displays a LILO-style “boot:” prompt where you can enter a kernel filename and options.

Sysexlinx searches for its configuration file in the following order: /boot/sysexlinx/sysexlinx.cfg, /sysexlinx/sysexlinx.cfg, /sysexlinx.cfg.

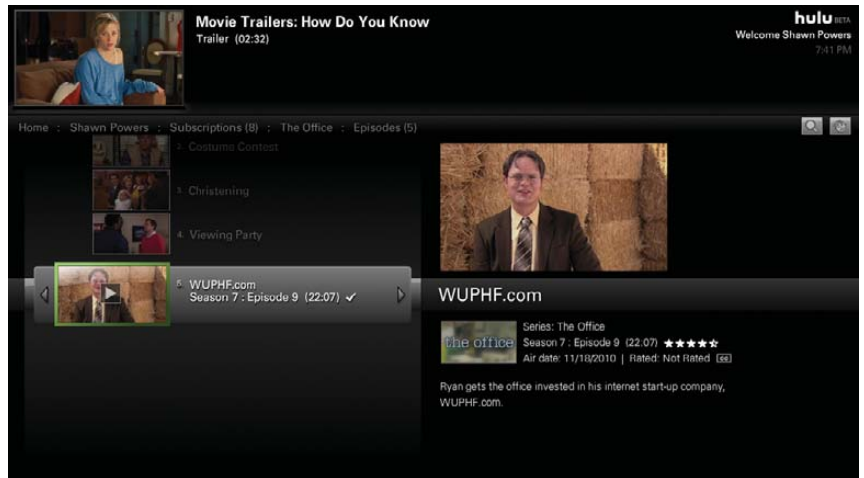
In this file, you can set parameters to change all the default settings. Any filenames in the configuration file are relative to the directory that sysexlinx.cfg is in, unless it is a full path. A basic example looks like this:

```
DEFAULT linux
LABEL linux
  SAY Now booting the kernel from SYSEXLINX...
  KERNEL vmlinuz.img
  APPEND ro root=/dev/sda1 initrd=initrd.img
```

The kernel types that Sysexlinx supports don't need to be a regular Linux kernel image. They can be a PXE bootstrap program, a boot sector or a COMBOOT image.

These techniques and utilities should give you a good start at putting your USB drives to their best use. You now can carry around your whole OS in your pocket. This is essentially what I've done for my old Eee PC. With Linux on USB, I can keep the same system there, and on my MacBook through Parallels. You also can use the same system on basically any machine that you can get to boot off of USB. Have fun and be creative.

—JOEY BERNARD

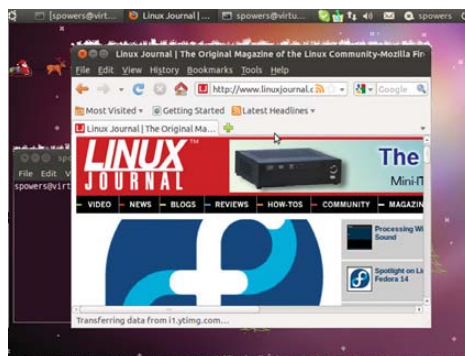


Hulu Hoop around the Desktop

I've been complaining about Netflix's lack of Linux desktop support for more than a year now. All my whining seems to be making little difference, so it seems only fair to look at alternatives. Granted, there aren't any direct competitors to the Netflix streaming service, but there is Hulu. Hulu has a history of not playing well with others and blocking developers like the Boxee team, but I must give credit where credit is due. It has a fully working Linux desktop program. Plus, the Web-based features work as well. If you add the \$7.99 monthly subscription for Hulu Plus, you happily can watch many seasons (sometimes the full catalog) of television programs directly on your Linux desktop. With Hulu's desktop application, you can send the whole experience full screen and have your own little home-entertainment system wherever you have Internet access. Sure, there still are ads. Sure, the catalog is different and less robust than the one from Netflix, but heck, at least Linux users are invited to the party!

—SHAWN POWERS

Silly Programs



Those of us who have been using Linux for a long time all know the joy of silly programs like xeyes. One of my favorites, however, is good old xsnow. Whether you love the cold weather or live in Florida and like to ski on occasion, xsnow will add some winter fun to your desktop. The xsnow program has been around forever and is surely available for your distribution.

If you're absolutely against snow and all its icy accomplices, you might want to check out a couple other oldies but goodies. Back before we had fancy computer games, we used to waste time with programs like xneko (you'll probably find it now as oneko), which was a little cat that chased your “mouse” around the screen. Or, perhaps you still enjoy the ever-staring eyes of xeyes (or a more modern tuxeyes). Finally, if your time-wasting tastes are a bit more on the macabre side, xroach (or groach) might tickle your fancy with bugs that scurry to hide under the windows on your desktop. Whatever your thoughts on silly time-wasting apps, you owe it to yourself to check out a bit of Linux/UNIX history.

Statistics with R

In the January 2011 issue, I looked at using Scilab to do some basic data analysis. Although you can do a lot with Scilab, sometimes it makes more sense to use a specialized tool. In the case of statistical problems, that specialized tool is R, developed at Bell Labs. R is actually a full programming language, designed for doing statistics tasks. It's very similar to another statistical programming language, S. Because R and S are programming languages, any actual programs are implementations of this language. Commercial implementations are available, like S-Plus. In this article, I discuss the open-source implementation by the R Project. References to R from here on refer to this open-source implementation.

R is broken up into a core engine and a full complement of libraries and functions that provide all its capabilities. Because of this, different downloads contain different prepackaged functions. However, an entire repository of extra functionality is available at CRAN (Comprehensive R Archive Network) where a strong community of people is willing to share the work. Thus, there usually is a function, or several, to handle almost any task you can imagine. If not, you can develop your own. Because R is broken down into a core engine and functions, any graphical interface simply is bolted on top of the core. Several graphical interfaces exist, such as RCommander or Rgui.

Graphical interfaces also are available for Windows and Mac OS X. Here, I'm looking at actual commands in R, so I'm using the text interface. To start it, simply type R and press Enter.

To quit R, type `q()`; and press Enter. In R, commands can end either with a semicolon or newline.

R has extensive help files for all the commands and functions, which are in the form of man pages. You can access these help pages in two ways:

```
> help(command)
```

Or:

```
> ?command
```

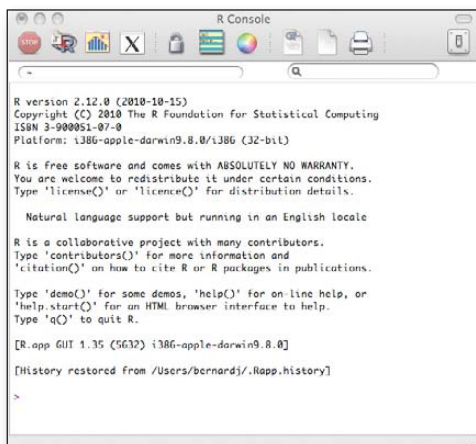


Figure 1. R on a Mac

You can run a whole series of commands with the source command:

```
> source("commands.R")
```

where the file "commands.R" contains all the commands you want to run as a single block. Think of it as if it were a shell script. You can send your output to a file with the sink command:

```
> sink("output.txt")
```

Once you are done writing the output, you can reset the output back to the console with:

```
> sink()
```

R can handle basic statistics out of the box. The available functions can do all the single variable statistics you're familiar with doing. The first step is to load data. Data is stored in lists, which can be arranged as arrays and matrices. The easiest way to create data arrays is to use the concatenation function `c`. It works well when you have only a little bit of data to work with:

```
> data1 = c(2,3,4,2,0,1,2)
```

This creates a single list and dumps these values into the variable `data1`. Then you can use this variable in other functions—for example, to find the average of these values, use:

```
> mean(data1)
[1] 2
```

The second line above is the output from the function `mean`. Output is labeled with a numeric label: `[1]`. The output from the first command in a session has the label `[1]`; the

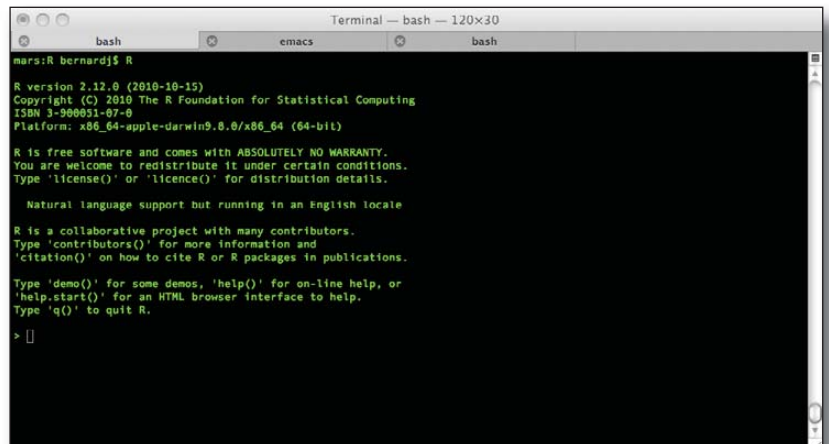


Figure 2. Starting R

output from the second command is labeled [2] and so on. If you have a larger amount of data to work with, use the read command to read in a file. If that data is in a table format, you can read the table in as a whole with:

```
> DataToUse = read.table("research.data")
```

You can access individual data elements by using the [] characters, so printing the second item is as simple as:

```
> data1[2]
[1] 3
```

If you want to set a value in a vector, use:

```
> data1[2] = 5
```

Let's say you were making some measurements on whales. You would start by saving the data into a variable:

```
> whale = c(74,122,235,111,292,111,211,133,156,79)
```

The first thing you would want to do is to run some basic statistical functions to find the mean, variance and standard deviation:

```
> mean(whale)
[1] 152.4
> var(whale)
[2] 5113.378
> std(whale)
Error: couldn't find function "std"
```

It looks like there is no standard deviation function. But, because R is a full programming language, you can write your own:

```
> std = function(x) sqrt(var(x))
> std(whale)
[3] 71.50789
```

This shows one of the shortcomings of R. Because it is such a large system, it does tend to have a relatively steep learning curve. More research would have shown that there is a built-in standard deviation function; it's just called sd instead of std:

```
> sd(whale)
[4] 71.50789
```

Sometimes it's worth taking the extra time to do some research before storming forward and writing all the code yourself.

R has very extensive graphing capabilities, as well. It generates very high-quality plots and graphs you can use in publications. The commands can be very simple if you just

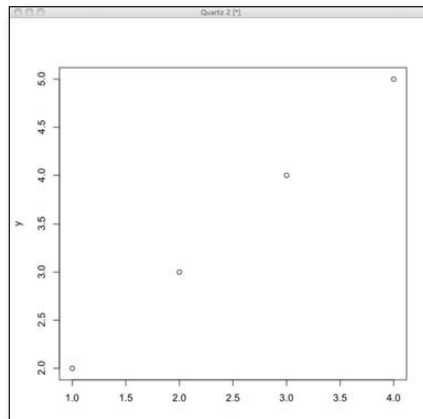


Figure 3. Example Plot

want a quick display so you can inspect your data visually. If you want very specific output, with lots of markup and customizations, R provides full control over all your graphs' details. The most basic plot command is simply plot. Let's say you have a series of 2-D data:

```
> x = c(1,2,3,4)
> y = c(2,3,4,5)
> plot(x,y)
```

This gives us the plot shown in Figure 3.

This looks like a straight line, so you probably want to do a linear regression on it:

```
> abline(lm(y~x))
```

This command does the linear regression of a straight line and plots it on the same plot (Figure 4).

It seems a pretty close fit, so you probably should look at the coefficients of the linear model:

```
> lm(y~x)
Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)          x
              1          1
```

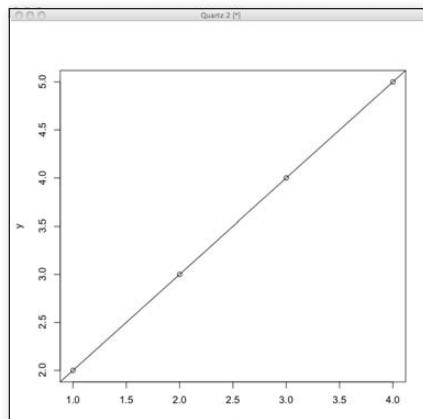


Figure 4. Linear Regression of a Straight Line Plotted on the Example Plot

So you end up with a straight line with a slope of 1 and an intercept of 1.

Once you've done all your analysis, you'll probably want to generate some nice plots for your publication. By default, when you call one of the plotting functions, the output is the graphical display. But, you can change this to an output file of a particular format. Let's say you want to generate some PNGs:

```
> png("filename.png")
> plot(x,y)
```

This will save your plot to the file "filename.png". You also can save to several other file formats (JPG, PDF, PostScript and so on). Many other options are available, so check out the help page.

This short crash course should give you a starting point for exploring R. See the R Project's main page for more information (www.r-project.org), and remember to check out the CRAN repository (cran.r-project.org) before you put too much work into it. If you'd like me to cover any scientific packages or computational science techniques, please let me know.

—JOEY BERNARD



REUVEN M. LERNER

Ruby on Rails 3

What's new in Ruby on Rails 3, and what has stayed the same? Reuven explores the latest version of this framework and what it means for you.

I still remember an IM conversation I had with a friend several years ago. He asked whether I'd heard of Ruby on Rails, and I replied that although I'd heard of it, I hadn't yet gotten the chance to work with it. Several months later, I had such an opportunity, and it wasn't long before I was hooked on both the Ruby language and the Rails framework.

It turns out that I wasn't the only one to be so smitten with Rails. During the past few years, Rails has changed the face of Web development. It has attracted many people (including myself) to the Ruby language, while also affecting, influencing and inspiring Web development frameworks in other languages.

When people ask why I like to work in Rails, I tell them there's no one answer, because I don't believe any one part of Rails is revolutionary. Rather, the developers have found hundreds, and perhaps thousands, of things that were traditionally difficult or annoying in other Web frameworks and have tried to sand down those sharp edges. For example, database migrations, which let you define and create your database schema in pieces, building it up over time, have saved me enormous headaches. The same is true for Active Record's automatic getter and setter methods for columns in the database. And for controller filters—the list goes on and on.

So, I have approached the development of Rails 3, which was released late last year, with some trepidation. After all, most things in Rails 2 worked well enough for me—why mess with a good thing? Besides, the merger with the development of Merb, a competing Web framework written in Ruby, didn't strike me as a particularly necessary thing. True, Merb was more open to alternative ORM, HTML and JavaScript libraries, but it wasn't too terribly painful to use such alternatives in Rails. Did we really need a major overhaul just to enjoy those benefits?

The answer, I'm happy to say, is that it was, indeed, worthwhile. Ruby on Rails 3 hasn't changed too much from its predecessor; it still looks, feels and acts like the Rails we know and love. Most changes are small and subtle, but where they aren't, you can understand the reasoning behind them and quickly adjust to the changes. I've been surprised to discover just how easily I've adjusted to the Rails 3 way of doing things.

This month, I give a whirlwind tour of Ruby on Rails 3. I describe where things have changed a lot, where they've changed a little, and where they haven't changed at all—at least, for a developer using it. Under the hood, Rails has changed quite a lot, adding a number of new

classes, modules and abstractions that have made it into a more modular framework than it was before. This means if you want to implement a new database-storage adapter, for example, Rails 3 will provide you with tools to make its implementation and integration fairly easy.

What Hasn't Changed

Before I go into what has changed in Rails 3, it's probably worth saying what's remained the same. First, Rails remains true to the model-view-controller (MVC) paradigm, with an emphasis on "fat" models—that is, the classes you define using models should contain the bulk of the business logic, as well as acting as an ORM and connection to the database. Controllers remain in the role as a go-between, accepting connections from the user and rendering the output in whatever format is appropriate. Views contain a combination of HTML and logic (with as little code as possible) that allow you to display everything from personalized menus and buddy lists, to XML and even PDF output.

Other classic areas of Rails, designed to support your main MVC design, also remain. Helpers, methods that allow you to remove code from your views, are still there. So is the lib directory, into which you can (and should) still put custom classes and modules, including those that will be used by multiple controllers or models. The public directory continues to be the repository for static assets, from JavaScript and CSS files to images and Flash that can be served up as-is.

Rails 3 certainly is more open to alternatives than was previously the case. When you create an application, you can specify that you want to remove Active Record, Prototype and/or test-unit if you prefer to use something else in their place. But these are options, and it means that for programmers who want to use the same systems they used in Rails 2, no changes will be necessary.

Although I haven't personally benchmarked the performance of Rails 3 in comparison with Rails 2, there's no doubt that it feels faster than Rails 2. This is especially true if you run Rails 3 with Ruby 1.9.2, the latest version of the language significantly faster than Ruby 1.8.7 in many respects. A great deal of attention has been spent on making Rails 3 more modular and much faster than Rails 2, and it shows.

Little Changes

That said, a bunch of little changes will become obvious almost immediately upon starting to work with Rails 3. For starters, Rails no longer has a bunch of

different commands in the script directory. Instead, you use the global rails command with a parameter indicating what you want to do. So, to create a new Rails application, you use `rails new appname`. But, to enter the console, you say `rails console` or, as a gesture to slow typists, `rails c`. Similarly, you can use `rails generate` to create a new controller, model, resource or scaffold (among other things). And, you can use `rails dbconsole` to open a client connection to the database defined in the current environment.

I must admit, I never had a problem with the old programs in the script directory. At the same time, this means your Rails application directory can be largely empty, because the scripts instead will come from the global Rails installation.

One small but significant change is the way values are displayed in views. In previous versions of Rails, the (default) ERb templating system allowed you to insert values into the resulting HTML as follows:

```
<h1>Welcome back, <%= current_user.name -%>!/</h1>
```

The `<% %>` brackets indicated that Ruby code should be evaluated, and the `<%= %>` brackets told Rails to insert the resulting value of that evaluation into the rendered HTML. The problem with this approach was that users potentially could enter not only text, but also `<script>` tags and JavaScript when they registered, leading to everything from annoyances to XSS (cross-site scripting) attacks.

In Rails 2, the solution was to use the “h” helper method, which turned potentially malicious tags and JavaScript into static text. This meant developers interested in removing XSS attacks had to use the “h” method in every set of `<%= %>` brackets that might contain user-generated input.

Rails 3 has reversed this default. Now, all strings are sanitized and defanged before they are displayed to the user. If you want to allow HTML tags (and potentially JavaScript) to be displayed, you need to use the “raw” helper method, which does the reverse of what “h” used to do:

```
<h1>Welcome back, <%= raw current_user.name -%>!/</h1>
```

Another change, but a bit more substantial, is the switch away from inline JavaScript from such helpers as `remote_form_for`. Instead, Rails sets attributes on the HTML tag; these attributes then can be used by callback functions to invoke JavaScript. The move to unobtrusive JavaScript is a welcome one, and it will go a long way toward uncluttering views with unnecessary JavaScript code.

Active Record

The jewel in the Rails crown continues to be Active Record, an ORM that allows you to think and code

with objects, while knowing (somewhere in the back of your mind) that each object instance is being stored to and retrieved from a relational database table.

The biggest change in Active Record is in the syntax you use to specify queries. In Rails 2, you would write something like this:

```
@people = Person.all(:conditions => "created_at > '2010-jul-14'",
                    :order => "last_name ASC, first_name ASC",
                    :limit => 10)
```

Rails 3 introduces “Active Relation”, a library used by Active Record, which changes the way you structure these queries. On the surface, the changes might seem annoying and petty. You would rewrite the above query as:

```
@people = Person.where("created_at => '2010-jul-14'")
                .order("last_name ASC, first_name ASC")
                .limit(10)
                .all
```

The first thing you’ll notice is that the query has been broken up into several methods. The second thing you’ll notice is that the `.all` call comes at the end. This is because until you invoke `.all`, nothing is executed in the database. This means you can mix and match method calls, adding additional conditions and complexity to your queries, until you finally decide to invoke it. So, you can build up a query over time, pass it as a parameter to a method, return it from a method or add to it conditionally—all before invoking the query on the database.

Aside from this, Active Record seems (again, on the outside) not to have changed very much. Validations now can be invoked with a slightly different syntax, foregrounding the name of the attribute you want to validate, rather than the validation itself. So instead of saying this:

```
validates_presence_of :email
validates_uniqueness_of :email
```

you now can say this:

```
validates :email, :presence => true, :uniqueness => true
```

I’m still not sure which of these syntaxes I prefer. Fortunately, at least for now, the old syntax still works and doesn’t raise a deprecation warning.

Under the hood, there have been a bunch of changes to Active Record. Active Relation is probably the biggest and most obvious, but another is the modularization of Active Record, such that validations now are in a separate module. This probably won’t affect most people, but it means if you develop a different ORM, or if you’re implementing an interface to a non-relational (NoSQL) database, such as MongoDB, you

now can use the validation system from Active Record without having to re-invent the wheel.

Overall, Active Record 3 continues to shine. It's one of my favorite reasons for using Rails, and I know I'm not alone in my appreciation for all the things it does to help my applications keep humming along.

Bundler

Perhaps the biggest change you are likely to encounter in Rails 3 is Bundler. One of the best things about programming in Ruby is the huge library of gems, or downloadable packages, available to the community. This is Ruby's answer to Perl's huge CPAN library, and although Ruby gems aren't even close to the size and scope of CPAN, they are being developed at an impressive pace.

Earlier versions of Rails attempted to handle the inclusion of gems inside the main configuration file, `config/environments.rb`. In this file (or any of its environment-specific files), you would put a line that looked like this:

```
config.gem 'will_paginate'
```

Doing so would ensure that the `will_paginate` gem is available and included. A number of Rake tasks were defined to automate the process of testing for installed gems, installing them on the system and installing them in the application.

Rails 3 changes all of this, in favor of a program called Bundler. The idea is that you create a file, called Gemfile, in the root directory of your Rails application. You then invoke `bundle install`, which goes through your Gemfile, ensures that all of the gems are indeed installed and creates a file called Gemfile.lock that specifies the precise versions that need to be installed for the application to run. Together, the Gemfile and Gemfile.lock files help ensure that the gems you need are installed on both your development and production machines, either in the system's gem directory or privately in the application.

I must admit that I'm still getting used to Bundler, and I haven't yet become convinced of how wonderful it is or how superior it is to the previous way we configured gems. I'm also more likely than not to forget to type `bundle install` before running my server.

That said, I do see an advantage to having all gem requirements and configurations, regardless of environment, in a single file. It certainly makes deployment easier on such hosting systems as Heroku, and it avoids some of the confusion I've experienced with various gems being required in different environments. This is perhaps the only feature of Rails 3 that has yet to overwhelm me with its advantages, but the pain of switching to Bundler is relatively small, and I have a feeling I'll become convinced over time that it's useful and even superior.

Routing

The last major change in Rails 3 that I cover here is routing. The router is the part of Rails that maps HTTP requests to controller methods. If a user sends a request of `GET /people/5`, it's up to the router to determine that the people controller's `show` method has an ID of 5.

For several years now, Rails has tried to simplify routing by encouraging the use of REST (representational state transfer). That is, each URL describes a particular object, and it is the HTTP verb (GET, POST, PUT, DELETE) that describes what you want to do to that object. If I just want to see person #5, I can say:

```
GET /people/5
```

But, if I want to update person #5, I would say:

```
PUT /people/5
```

along with name-value pairs describing the attributes that should be updated.

In Rails 2, you would describe a resource in the routes file (`config/routes.rb`) with a line like this:

```
map.resources :people
```

In Rails 3, things have gotten simpler. For starters, you can declare a resource as follows:

```
resources :people
```

In and of itself, that's not a big change. But, if you need to add additional methods beyond the standard seven (`index`, `show`, `new`, `create`, `edit`, `update` and `destroy`) Rails provides, you can do so in a block:

```
resources :people do
  post "send_feedback", :on => :collection
end
```

This was certainly possible with the older syntax, but it was a bit more convoluted.

If you simply want to add a single, hard-coded route, you can do it with:

```
get "home/faq"
```

Rails 3 is smart enough to know this means you want GET requests for `/home/faq` to be redirected to the home controller and to invoke the `faq` method.

I must admit that Rails routing always seemed like a bit of black magic to me, even after years of working with it. The simplification that Rails 3 offers is most welcome, in that I feel I have a much better understanding of what it does and how it accomplishes it. Best of all, my `routes.rb` file has become less complex and easier to

maintain, which is, as I mentioned previously, one of the biggest reasons for working with Rails.

Should You Upgrade?

After describing Rails 3 so enthusiastically, you might expect that I'll tell you to go out and upgrade whatever Rails 2 applications you might have written immediately. And, indeed, if the application is simple and small, such an upgrade is not a bad idea.

But, if you're like me and work with some large, complex applications, switching is not likely to be a quick or easy process. A large number of queries will need to be rewritten to use Active Relation. The routing table will need to change, the Gemfile will need to be rewritten, and you'll have to double-check your helpers for XSS attacks. This doesn't mean upgrading is a Herculean task, but it's not something to attempt in one evening.

The first thing I would suggest to anyone considering an upgrade is to ensure that your tests, and especially your integration tests, are in place, passing and providing you with adequate test coverage. Once you have such tests in place, you can start the upgrade process, checking at every stage to see what you might have broken.

A plugin called `rails_upgrade` is available; it provides a Rake task that looks through your code and configuration, tries to determine places where you might encounter problems and points them out to you. If you have good test coverage and run `rails_upgrade`, you're likely to know where potential problems lie and whether you've broken anything when you try to align

your code with Rails 3.

Finally, upgrading to the latest version of Rails 2 (2.3.10, at the time of this writing) is a good way to prep for an upgrade to Rails 3. If you read through your logs, you'll find a number of deprecation warnings that are meant to nudge (push?) you in the direction of Rails 3 compatibility. Even if you don't plan to upgrade right away, getting your code closer to Rails 3 standards won't be a bad thing.

Conclusion

Rails 3 is a terrific upgrade to an already great Web development framework. While it was under development, I followed the blog postings and explanations with some trepidation, wondering how different Rails 3 would be from previous versions. I needn't have worried. As Rails has improved my experience as a Web developer in many small ways, so too has Rails 3 improved on its predecessor with many small changes. I generally like these changes a great deal and believe this shows that Rails still has a lot going for it. Upgrading might be tricky in some cases, but it's worth working in that direction, even if it takes a while. The application's execution speed will improve as a result, but so will your development speed and the maintainability of your code. ■

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

Resources

The home page for Ruby on Rails is rubyonrails.org. That site has pointers to downloads, documentation and many resources offered by the community.

One terrific community resource is the "Rails Guides" documentation, which can walk you through a large number of features in Rails. These guides are great reading for Rails developers, new and old alike. They are current for Rails 3 and often point to differences with Rails 2 for those planning to upgrade.

The weekly "Railscasts" screencast produced by Ryan Bates continues, as always, to be a source of knowledge and inspiration. A large number of Railscasts in 2010 were devoted to Rails 3 and to what it means to upgrade. I viewed these several times before working with Rails 3, and they made a big difference in my work.

The `rails_upgrade` plugin can be downloaded and installed from its GitHub home page at github.com/rails/rails_upgrade.

You can learn more about Bundler at gembundler.com. Yehuda Katz also has a blog posting about Bundler and what it means for Rails developers, at yehudakatz.com/2010/09/30/bundler-as-simple-as-what-you-did-before.

Finally, a few books about Rails 3 have started to emerge. The best-known book from the Pragmatic Programmers, *Agile Web Development with Rails* written by Sam Ruby, Dave Thomas and Rails creator David Hannemeier Hansson, remains a solid introduction to the framework. A less-dense introduction, *Beginning Rails 3*, published by Apress and written by Cloves Carneiro Jr. and Rida Al Barazi, probably is a good starting point for people new to Web development.

And, although it is not yet published at the time of this writing, *Rails 3 in Action* by Yehuda Katz and Ryan Bigg, published by Manning, has the look of a real winner. I particularly like the fact that it uses Cucumber and RSpec. It also mentions third-party gems and services that a production Rails application is likely to use, such as Git, Hoptoad and paperclip.



DAVE TAYLOR

Making a *Mad Libs* Generator

Using shell scripts to re-create this classic grammar game.

My son is at the age when he's decomposing sentences, diagramming them and learning about the parts of speech. Me? I couldn't differentiate between an adverb and an adjective if a wet, smelly red ball smacked me in the head. That's why I have an editor!

There are games for everything, however, and one of the best games for learning the parts of speech is a simple one that's been around since I was a kid: *Mad Libs*. You know what I'm talking about, it takes simple sentences like: "When my dog is happy, he jumps and barks, his tail wagging a mile a minute." and transforms them into: "When my [noun] is [adjective], he [verb] and [verb], his [noun] wagging a mile a [noun]."

The question is, can we write a shell script that can perform this sort of transformation? The answer, of course, is yes.

Identifying Parts of Speech

There are two challenges with this project: figuring out which words to replace with their parts of speech and figuring out the part of speech of a given word. Let's tackle these in reverse order.

It turns out that a number of different Web sites let you look up a word and offer its definition and part of speech. The one I use for this exercise is from Princeton, because it's fast, easy to parse and easy to submit queries.

To look up the part of speech of, say, "dog", the URL to invoke is simply wordnetweb.princeton.edu/perl/webwn?s=dog.

The result highlights the part of speech as an h3 line, so isolating that element is a breeze:

```
curl --silent "lookup$word" | grep '<h3>'
```

This particular word demonstrates one of the nuances of the problem: many words have more than one part of speech, demonstrated by the difference between a pet dog and someone who is dogging your every footstep. Sure enough, the result:

```
<h3>Noun</h3>
</ul><h3>Verb</h3>
```

For simplicity's sake, let's just take the first match, easily done by adding `| head -1` to the pipe. Next, let's drop it all into lowercase and strip out the HTML:

```
| tr '[:upper:]' '[:lower:]' | sed 's/<h3>///;s/<\/h3>///'
```

Both of these are worth a bit of explanation. You might well have seen `tr '[A-Z]' '[a-z]'` as the more common way to transliterate uppercase to lowercase, and that works just fine, if you're working in English. Using the character sets `"[:upper:]"` and `"[:lower:]"` is a more portable alternative that's preferred.

The `sed` command also lets you specify more than one command argument to apply by simply separating them with a semicolon. What we have here is a substitution of `<h3>` to a null string (for example, removing it), followed by the same thing for `</h3>`.

That's all we need to get the part of speech. For example:

```
$ lookup="http://wordnetweb.princeton.edu/perl/webwn?s="
$ word="happy"
$ curl --silent "$lookup$word" | grep '<h3>' |
  tr '[:upper:]' '[:lower:]' | sed 's/<h3>///;s/<\/h3>///'
adjective
```

And, the hard part's done!

Choosing Words to Replace

For this article, let's use a replacement density constant to figure out whether any given word should be replaced. The higher the density, the more likely a given word in the input stream will be replaced by its part of speech.

This is lazy and not a great solution, because it can match "is" or "the" just as easily as "dog" or "tail", but let's go with it for now to get a sense of how it'll all fit together. We'll come back to it and improve the sophistication of the selection criteria later. With me? Good!

For a given word, deciding whether to substitute its part of speech can be calculated as follows, assuming we have a variable called `density` that has a nonzero integer value:

```
if [ $(( $RANDOM % $density )) = 1 ] ; then
```

`$RANDOM` is one of those cool magic variables in the Bourne shell that has a different value each time you reference it—handy!

Putting Things Together

Let's put these together and see what we get. We'll

use an initial density of 5, which theoretically should mean that if we have a properly random \$RANDOM, each word should have a 1:5 chance of being replaced.

The script needs to read the input word by word, testing each word as it goes. This can be done easily with the following loop structure, assuming that the text input comes from stdin:

```
while read sentence ; do
  for word in $sentence ; do
```

Now, we add the random conditional and have a skeleton ready to test:

```
while read sentence ; do
  for word in $sentence ; do
    if [ $(( $RANDOM % $density )) -eq 1 ] ; then
      echo "($(word))"
    else
      echo $word
    fi
  done
done
```

You can see that at this stage we're going to output the words we're planning on replacing with "(())". Here's a quick test:

```
echo this is a test mad-lib input | sh make-madlib.sh
this
is
((a))
test
((mad-lib))
input
```

One tiny tweak before I wrap it up for the month—how do we get the words to appear on the same line? It's easy. Remember that each of the code loops is essentially a little script of its own, so this task can be accomplished by adding four characters to the very end of the outermost loop:

```
done
done | fmt
```

That's all you have to do—add the | fmt after the second done statement. Now when it's run:

```
echo this is a test mad-lib input | sh make-madlib.sh
this is a ((test)) ((mad-lib)) input
```

Next month, we'll add the part of speech lookup code into the conditional and then spend some time exploring a more sophisticated word choice algorithm. Clearly, random isn't as beneficial. ■

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



Linux - FreeBSD - OpenSolaris - etc.

Proven **Technology**.

Proven **Reliability**.

When you can't afford to take chances with your business data or productivity, rely on a Genstor server customized to your specifications.

POWER

PERFORMANCE

Fly into the Cloud with Genstor Systems



- Up to 48 cores in a 1U.
- AMD Opteron 6100 series.
- Single high-efficiency power supply.
- Up to 512GB DDR3 memory.
- Ideal as front end processing servers.



- Up to 12 cores in a 2U
- Dual redundant high efficiency power.
- Up to 96GB DDR3 memory.
- Server Power Capping via Intel Intelligent Power Node Manager.
- Ideal as front end processing and/or storage.



- Up to 4 GPU cards.
- Dual redundant high efficiency power.
- Up to 192GB DDR3 memory
- Up to 24 cores using 2 CPUs.
- Up to 8 3.5" disks.

Genstor Systems, Inc.



Powerful.
Intelligent.

1501 Space Park Drive
Santa Clara, CA 95054
www.genstor.com

E-mail: sales@genstor.com
Phone: 877-25 SERVER
408-980-0121

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



MICK BAUER

Interview with a Ninja, Part I

Mick chats with a highly skilled, highly ethical system cracker.

It's a strange world we live in now, isn't it? We communicate through machines as often, and with more different people, than we do in person. Some of us have careers that didn't exist when we were in college. Some of us even have careers that involve doing things we used to do secretly, possibly even illegally.

Consider the professional penetration tester, who breaks into his clients' systems and applications for the purpose of testing and improving their security. To me, this is one of the most remarkable professions a person can have in information security.

Some people are skeptical of the value of this line of work. What does it prove? If penetration testers succeed, have they really learned more than, or even as much as, they would have learned by performing a careful analysis of system and application configurations, and maybe by analyzing some source code? If penetration testers fail, did they simply have a bad day, or is the system in fact secure?

Although I'm sure there is such a thing as a redundant or otherwise unhelpful penetration test, I can assure you that in my own role as a network security architect, there are plenty of situations where a rigorous, professional penetration test is essential. Source code isn't always available, configurations don't always tell the full story of system behavior, and some things are too new for there to be any "conventional wisdom" of how secure they are, or how they can be made secure. A good penetration test can help me determine which of a vendor's claims hold up, and which system administrator practices are being adhered to.

Therefore, I work closely with and proudly count as friends some outstanding penetration testers. Luckily for you, gentle reader, this month, one of them, who for our purposes here I'll simply call Ninja G (for reasons that will be revealed shortly), graciously agreed to a *Linux Journal* interview, in which we discussed his singular career, what he finds interesting in computer security, and what he sees as notable trends in the current state of Linux/UNIX security.

MB: Thanks so much for taking the time to chat. By way of an introduction, can you explain what your official duties are and how you spend a typical day?

NG: I work in the finance industry for a Fortune 500 (or less) company. The duties of those in my group include penetration testing, enterprise-wide network

security scanning and real-time security event response. My days vary greatly from one week to the next, which is a wonderful thing about this sort of work. One week I am testing a small embedded Linux device, the next week it could be a remote server or Web site, corporate Wi-Fi or VoIP network.

I believe this classifies me as a generalist, so it probably isn't surprising that a large part of my days are spent coming up to speed on the target platform. This can include securing logical or physical access, reading all the vendor-supplied documentation, researching all currently known and historical vulnerabilities and then actually using the platform or device (as intended) to gain a holistic view of how things normally work. Only then can I start to ask myself all of those evil little questions concerning how the designer decided to handle situations that (under normal circumstances) were never intended to happen.

MB: How did you arrive at this kind of a job? I'm assuming you didn't major in penetration testing in college!

NG: College? Heh, no. I was an illicit computer/phone hacker during the 1980s and early 1990s. I don't have a criminal record, but I can't say that I was never caught; rather, at one point I was the focus of a multi-agency investigation. I never was prosecuted, but my "close call" officially ended my desires to intrude illegally into systems as a hobby.

It was at that point that I started looking into jobs in the Information Security field. I was hoping to find an outlet where I could continue the passion of my past hobby, yet without the risk of prosecution. The problem was trying to find just the right company that not only had an Information Security department, but also found value in beating up proposed solutions to look for potential security problems. I knew that large financial organizations had an interest in this, but unfortunately, I lived in small town, which was exactly the wrong area for this type of work.

So I "followed my bliss" and ended up in a large city, working for a financial company whose products are offered globally. I wouldn't say that my first job in this industry was *exactly* what I wanted, but often your job is what you make it, so I slowly started revealing my skill level in the arcane "hacking" arts. Word gets out, rumors spread, and very slowly more and more of the work that interested me started to come my way.

Fast-forward 16 years, and I'm working for a larger financial company in an even larger city. Now the bulk of my time is spent on penetration testing, so I can't complain at all.

MB: What do you find most interesting in your job and your career? What sorts of problems are the most fun, and why?

NG: Learning new things every day. Difficult problems are always the most rewarding.

In this line of work, you are only as good as your last major find, and people tend to forget quickly about past (now-mitigated) vulnerabilities. It isn't uncommon to go for days without stumbling upon new vulnerabilities of any significance, which is *horribly* depressing. Perseverance at these times often cuts deeply into my own personal time, but it almost always pays off.

All it takes is discovering one new zero-day (vulnerability), and I'm back on top of the world—for a day or so. Then, the process starts all over again with something else.

As a generalist, I really don't have any preference on what types of devices, software or systems I test. Each presents their own lessons and potential for discovery.

MB: WLAN, as a target, seems to be especially popular in the criminal community. Does WLAN stand out, one way or another, in your own work, or is it just the same types of problems and vulnerabilities, only over radio waves?

NG: No, I would say that wireless presents its own unique set of security ramifications and even heavily trained corporate WLAN administrators sometimes don't comprehend concepts as simple as malicious server certificate re-use. If you're unfamiliar with this style of "insider" attack, it can be summed up in one sentence. An authorized Web server administrator creates a malicious (evil twin) wireless access point that mimics the corporate WLAN (and which they load with the valid certificate from their Web server from the same name-domain); however, they configure it to merrily accept (and log) all authentication credentials provided, regardless of what they are.

For those wanting to play with this sort of attack, Josh Wright's WPE (Wireless Pwnage Edition) patch to FreeRadius is a nice shortcut to RADIUS server impersonation. Of course, back when I was first playing with this, that didn't exist, so I manually patched HostAPd to do the same thing. In more sophisticated attacks,



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



As the Chief Financial Officer for Silicon Mechanics, Steve is an Expert where value is concerned. That's why he's pictured here with the Rackform iServ R143.

The R143 is a flexible and affordable 1U server. It features an Intel® Xeon® Processor 3400 Series, with powerful features like Turbo Boost, Hyper-Threading, and DDR3 memory. This processor is also available in a low-voltage version, which can optimize power usage and help contain energy costs. With 6 DDR3 DIMM sockets, 2 Gigabit Ethernet adapters, a PCIe expansion slot, and 4 hot-swap SAS/SATA drive bays, the R143 can handle a lot more than entry-level workloads. With a price that starts around \$1250, you don't have to be a CFO to understand the value.

When you partner with Silicon Mechanics, you get more than a flexible, affordable entry-level server — you get an Expert like Steve.

For more information about the
Rackform iServ R143
visit www.siliconmechanics.com/R143

Expert included.

Silicon Mechanics and the Silicon Mechanics logo are registered trademarks of Silicon Mechanics, Inc. Intel, the Intel logo, Xeon, and Xeon Inside, are trademarks or registered trademarks of Intel Corporation in the US and other countries.

I have “Frankensteined” a modified HostAPd to Xsupplicant, and successfully performed full man-in-the-middle attacks against WPA2/AES with hardware one-time password tokens.

There is also a bunch of older attack code floating around that doesn’t really work well anymore, as it was written before QoS was a common feature in access points. I have updated a lot of this code for my own use, plus I have written several of my own wireless attack tools. My favorite sort of wireless exploit to write would probably be best categorized as “blended surgical attacks” where the real action happens at layer 7, and forensics after the fact is next to impossible, as the attackers aren’t connected to any WLAN, nor do they ever expose their own MAC address.

MB: I’ve noticed that a lot of guys on your team study ninjutsu, the martial art practiced by real-life (not metaphorical) ninjas. Is this a coincidence? What’s the ninja-hacker connection?

NG: Oh, so you noticed that! Probably just a coincidence. To be fair, I would say that I study a system of Japanese martial arts that contains nine different budo taijutsu koryu (old-school combat arts), only three of which are classified as ninjutsu.

The connection? Well, I can’t speak for any other martial artist, as everyone has his or her own reasons for pursuing such a hobby, but I do have a few theories on why people who are interested in computer hacking or penetration testing may be attracted to specific martial arts, such as ninjutsu.

First, both hackers and ninjas benefit from the mythology spun around their mystique. Often our labels for these individuals come along with a lot of misconceptions, which usually don’t exactly hurt the people being labeled, as often they are attributed with near super-human abilities and skills. (Win! Unless these labels are used in a court of law, then: fail!)

Second, as my teacher often says, “We must learn the techniques and tactics of the bad guys, if we are to defend against them.” The same holds true for anyone who is skilled at breaking or securing systems. Knowledge of what the other guy is likely and capable of doing at any point can make a huge difference.

During martial training we learn to do some rather nasty things effectively, and then we explore the areas of space between attacker and defender, which, if occupied, could lead to either an optimal or sub-optimal outcome for the defender. We “play” in this way until the mind starts to form almost a three-dimensional model of what space leads to which outcome. My teacher often says, “Move through the ‘safe shaped’ space!”, which is bewildering to those unfamiliar with this concept. A lot of budo taijutsu training is counter-intuitive like that. Which brings me to my last point.

I once read that people who play mahjong, or other

puzzle-like games can keep their mind sharp well into their autumn years—provided that the puzzles are *hard*. I’m pretty certain that those who exercise their minds by hacking are probably naturally drawn to other activities that are equally stimulating and challenging on a mental level.

Of course, hacking problems are way different from budo taijutsu problems, as they use completely different sections of the brain. When hacking, you have the luxury of being able to stop and think for a bit. Stopping occasionally to scratch your head, ponder and plot can go a long way to moving you forward to your end goal. Now imagine that your problem isn’t breaking into a system, but rather surviving a knife attack. There simply is no time to stop to think; you just have to move and trust that your mental map of interactive space, balance taking, weapon usage, striking and so on is sound.

Training is often chaotic, with the teacher varying the conditions (such as weapon type or length, distance, number of attackers and so on) frequently. At higher levels of training, things start moving and changing so quickly that there really is never a time set aside to stop mentally to comprehend your experience. So instead of having a cognitive understanding of exactly what happened each time, you have only a general “feeling” of what is happening.

MB: I know that you work with Linux in at least two different contexts in your day job: as a platform from which to conduct attacks and also as a target, since so many of the things we ask you to test nowadays seem to run on Linux! What sorts of trends, good and bad, do you see in Linux security?

NG: On embedded Linux systems, I often find the sorts of problems that plagued larger UNIX systems decades ago, except now they can’t be blamed on a clueless or lazy system administrator. Instead, it is often an embedded Linux developer who doesn’t understand things like file ownership and permission bits, temporary file name prediction, proper usage of hard and soft links, command chaining, race conditions and exactly why it may be a good idea to update that prehistoric version of BusyBox.

One trend I see is that some embedded developers have gone to great lengths to attempt to enhance the security of the embedded Linux environment (to varying degrees of success). Their efforts range anywhere from a secondary password challenge-response scheme for root access, to neutering root completely, to only executing signed binaries, to removing all login services and shells entirely. A lot of these same things hold true for non-embedded Linux appliances, where users are not generally traipsing through the filesystem with shell access.

Of course, all newer Linux platforms have benefited greatly from some of the more recent developments in memory protection. There is still more work to be done, but I definitely would classify things like executable space protection as a good trend.

MB: By saying “neutering root completely” are you referring to SELinux and other role-based approaches to security? The “root is omnipotent” aspect of Linux’s security model has always been its soft spot, hasn’t it?

NG: SELinux is sometimes used (usually in full-size appliances); however, you probably would be surprised at some of the one-off, homegrown solutions that some vendors have implemented to attempt to rein in the almighty “root” in embedded Linux solutions.

Some vendors simply decide that root has no real purpose as a user account, so they cut it off at the knees (by giving it an invalid shell and a locked-out password). The system still will function perfectly, but whenever there is a need to upgrade system files, it usually involves completely reflashing the firmware for the device.

Other vendors choose to keep the root account, and then build their own mechanisms where upgrades are handled by root, but practically everything else runs as a nonprivileged user. In theory, this is a good approach, but again, even a simple misunderstanding of how hard and symbolic links work can spell disaster for upgrade routines that expect only vendor-provided (properly formed) upgrade files. If you can predict ahead of time where a temporary file used in the upgrade process is

going to exist (say, /tmp), and you get there first with a link, many times you can end up clobbering the target of your link as root. If you get lucky, the script also may have a more-permissive umask setting, which would end up modifying and maybe lowering the file permissions to something less than before.

To Be Continued

Dear readers, at this point in the interview, I’ve got good news and bad news. The bad news is, I’m out of space for this month’s column and will have to wait for next time to share the rest of this fascinating and fun conversation with you.

The good news is, in Part II, you’ll learn about Ninja G’s opinions on the role of firewalls, the corrupting influence of hacking skills, the importance of responsible disclosure and his predictable yet surprising answer to the eternal question of who is more elite, pirates or ninjas. I hope you look forward to it! ■

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

Small, Portable Devices with Ubuntu Linux

Small Form Factor Intel® Atom™ Platform

No fans, no moving parts. Just quiet, reliable operation. Incredibly compact and full featured; no compromises.



VESA-Mountable NVIDIA® ION/ION2 System

Compact, lightweight system with GeForce® Graphics. Flexible storage options (dual HDD support) and WiFi.

Value only an **Industry Leader** can provide.

Selecting a complete, dedicated platform from Logic Supply is simple: Pre-configured systems perfect for both business & desktop use, Linux development services for greater system customization, and a wealth of online resources all within a few clicks.

[Learn More > www.logicsupply.com/linux](http://www.logicsupply.com/linux)

LOGIC
SUPPLY



KYLE RANKIN

Your Own Personal Server: the Network

This first installment of Kyle's personal server series will walk you through some common network hangups that can get in the way of hosting your own server at home.

These days, it seems everyone is talking about the cloud. Now, what exactly someone means by "the cloud" seems to vary, but typically, the cloud refers to some sort of service, such as e-mail, Web, DNS, file storage and so on, that is managed for you by a third party. Many people love how easy it can be to outsource their e-mail service, blog or image site to someone else. Like oil changes, home repair and cooking, server administration is yet another task you can pay (either with money or with marketing data) someone else to manage for you.

Alongside this trend to outsource work is a growing movement that values doing things yourself. Some examples include "Makers" involved in designing their own electronics, do-it-yourself home improvement,

It turns out it is quite rewarding, educational and not terribly difficult to manage your own services at home instead of outsourcing them to the cloud.

gardening, amateur cheese making, baking and even home brewing. The fact is, many of these so-called chores actually are rather rewarding and even fun to do yourself. I think we as Linux users should apply this same idea to server management. It turns out it is quite rewarding, educational and not terribly difficult to manage your own services at home instead of outsourcing them to the cloud. This is on top of the fact that when you manage your own server, you are in full control of your server, what's installed on it and who can see it.

In this series of columns, I'm going to discuss how to set up various types of services at home and how to make them available to the Internet at large. In this first column of the series, I discuss some things you should consider about your network before you set up your first server at home.

Your Internet Connection

When it comes to hosting servers at home, all ISPs (Internet Service Providers) are not created equal.

Before I even discuss bandwidth, first you should look into your ISP's terms of service. It turns out that some ISPs discourage, disallow or sometimes outright block home users from hosting their own services on the Internet. Take a large dose of caffeine and try to read through your ISP's terms of service (or just call and ask them) to see whether they have any sort of restrictions. These days, at the very least it's common for even server-friendly ISPs to block outbound e-mail traffic (SMTP port 25) by default to prevent spam. Although I'll discuss this more in a future column about e-mail, some ISPs will lift this restriction and some won't. The bottom line is that if hosting your own server is important to you, you will want to make sure you use an ISP that allows it. For me, this policy is more important when choosing an ISP than even speed or price.

Static IPs vs. Dynamic IPs

No matter what type of Internet connection you have, ultimately you are assigned at least one publicly routed IP address. If this address changes each time you connect to the Internet (or each time your DSL or cable modem resets), you have a dynamic IP. If this IP stays the same, it's static. Although people historically have run servers on both static and dynamic IPs, with a dynamic IP, you will have to go through the additional trouble of setting up some sort of dynamic DNS service so that each time your IP changes at home, everyone trying to access your service on the Internet will get the new IP. Unfortunately, due to the nature of how DNS works, you can't always guarantee (even with low TTLs) that everyone will see your changed IP in a timely manner, so if you are serious about running servers at home, I recommend you spring for one or more static IPs.

Connection Speed

Typically when you rate the quality of your Internet connection at home, you first look at your download speed. Average home users rely much more on their download bandwidth than their upload bandwidth as a metric of how "fast" their connection is, and many home Internet connections have much higher download bandwidth than upload. Once you start hosting servers at home, however, you'll find that their performance is governed more by your upload bandwidth. If

you want to host bandwidth-hungry services at home, like streaming audio or video or image-heavy Web sites, you might want to upgrade or change your Internet connection to get more upload bandwidth. On the other hand, your personal DNS or e-mail server probably is going to be fine even with somewhat low upload bandwidth. Although upload bandwidth can be slower at home than at a data center, most connections at home (at least in the US) are unmetered so you don't have to worry about bandwidth caps.

Modems and Gateways

These days, most people who would want to host a server at their homes tend to access the Internet through some sort of DSL or cable modem. This device connects to either a phone line or some other cable on one end and provides a network port (or sometimes a USB port) on the other. More sophisticated modems actually can act as a gateway, and even a DHCP server, and hand out internal IPs to computers in the home while the public IP resides on the modem itself.

If you plan on having multiple computers inside your home network, I recommend getting the modem configured so it acts more like a bridge, so that the publicly routed IP address is assigned to a device that is under your control, whether it's a home router or a computer on your network. Most home routers these days (including DSL and cable modems, if your ISP gives you the ability to configure them) have the ability to do port forwarding so that incoming traffic intended for your Web server (ports 80 and 443) can be redirected to the internal IP address. The more control you have over your gateway, the more flexibility you will have in how you set up your servers and your network. If you do opt to use a consumer router instead of turning a home computer into the gateway, you might want to choose a router that can be reflashed with custom Linux firmware (like OpenWRT or DD-WRT), so you can have some of the same flexibility you would have if a Linux server acted as the gateway.

Security, Firewalls and Virtual IPs

Of course, any time you open up a service to the Internet, you are opening yourself up to attack. These days, it doesn't matter if you just have a lone server on the Internet; attacks are automated, so your obscurity doesn't ensure security. Be sure that any service and server you make available on the Internet is kept up to date with the latest security patches. If you have the ability to configure a firewall on your gateway router, block all incoming ports by default and allow in only ports you know need to be open. If you are going to open up an SSH server to the public Internet, be sure to audit your passwords, and make sure they are difficult to guess (or better, disable passwords altogether and use key-based authentication). These days, more home (and enterprise) Linux servers are hacked due to

bad passwords than just about anything else.

While I'm on the subject of firewalls, here's a quick tip if you happen to use a Linux device as your router with iptables. Even if you are granted multiple public IPs, you may find you prefer to have all Internet traffic come through a central router so it's easier to monitor and secure. To accomplish that, you likely will need to have your gateway device configured to answer on all of the public IPs and assign private IPs to the computers inside your home. Let's assume I have a few static IPs, including 66.123.123.63 and 66.123.123.64, and a gateway router that is configured to answer to both of those IPs on eth0. I have an internal server on my network with an IP address of 192.168.0.7. Because it has an internal IP, I want to forward traffic on my gateway destined for 66.123.123.64 to 192.168.0.7. The first way I could do it is to forward traffic only on specific ports to this host. For instance, if this were a Web server, I might want to forward only ports 80 and 443 to this server. I could use these iptables commands on my gateway router for the port forwarding:

```
iptables -t nat -A PREROUTING -d 66.123.123.64 -i eth0 -p
➤tcp -m tcp --dport 80 -j DNAT --to-destination 192.168.0.7:80
iptables -t nat -A PREROUTING -d 66.123.123.64 -i eth0 -p
➤tcp -m tcp --dport 443 -j DNAT --to-destination 192.168.0.7:443
```

This is also a common solution if you have only one public IP but multiple servers in your network, so you can forward Web ports to an internal Web server and e-mail ports to a different e-mail server. This method works; however, I'll have to be sure to add new firewall rules each time I want to forward another port. If I simply want to have the router forward all traffic destined for 66.123.123.64 to 192.168.0.7, I could use these two commands:

```
iptables -t nat -A PREROUTING -d 66.123.123.64 -i eth0 -j
➤DNAT --to-destination 192.168.0.7
iptables -t nat -A POSTROUTING -s 192.168.0.7 -o eth0 -j
➤SNAT --to-source 66.123.123.64
```

Note that because these commands forward all traffic to that internal host, regardless of port, I will want to make sure to lock down the firewall rules on that internal server.

This should be enough information to get you started on your network setup at home so that by next month, you'll be ready to set up your first service. In my next column, I'll focus on DNS, including how to register a domain and how to set up your own home DNS server. ■

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.

WIN Enterprises' PL-80260

The latest offering from WIN Enterprises is the PL-80260 family of desktop networking platforms. The devices are designed to support routing, firewall/threat management, database and NAS networking applications in small- to mid-size businesses and remote enterprise offices. The 9.1"-wide devices are powered by energy-efficient Intel Atom processors and feature up to six Intel GbE LAN ports. The device pairs either the Intel Atom D410 single-core or D510 dual-core processor with the Intel 82801HM I/O Hub. Intel hyperthreading technology increases logical CPU threads, resulting in more efficient use of processor resources. The unit is RoHS-compliant. OS support includes Fedora 13, Debian 5.0.6, openSUSE 11.3, Windows XP Pro and Windows 7.

www.win-ent.com



Digi's ConnectPort X3 H Cellular Gateway

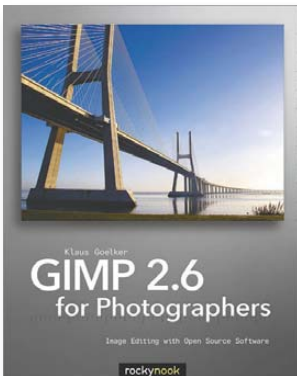
If you have far-flung assets that require monitoring, a potential solution might be Digi International's recently released ConnectPort X3 H programmable cellular gateway. The product is built for monitoring remote assets in harsh environments, including pipelines, agriculture, utility assets, research and others where exposure to volatile liquids, gases and severe temperatures is common. It also integrates with the iDigi platform, a cloud computing service that enables remote management and integration of devices and device information into a company's back-end systems. Key features include advanced battery power and a NEMA 4X/IP66 enclosure to protect the gateway from water, dust and dirt. Global connectivity is via GSM GPRS cellular networks.

www.digi.com/x3

TestOut Corporation's LabSim Linux+ Powered by LPI

Is your New Year's resolution to be a certified Linux guru? If so, browse on over to TestOut Corporation and check out its new on-line training course LabSim Linux+ Powered by LPI. The all-new browser-based course is designed to meet the revised and standardized certification objectives from CompTIA and the Linux Professional Institute (LPI), such that one can pass both of the new CompTIA Linux+ exams, LX0-101 and LX0-102. TestOut says that its on-line labs simulate a physical lab, letting students experiment with realistic, real-world scenarios from their own computers. The resources—labs, videos, demonstrations, and informational and self-exam materials—cover system architecture, Linux installation and package management, GNU and UNIX commands, devices, Linux filesystems and the Filesystem Hierarchy Standard.

www.testout.com



Klaus Goelker's GIMP 2.6 for Photographers (Rocky NOOK)

Kiss Photoshop goodbye and go GIMP with Klaus Goelker's new book *GIMP 2.6 for Photographers: Image Editing with Open Source Software*, published by Rocky NOOK. The GIMP is an open-source image editing tool for Linux, Mac OS X and Microsoft Windows that provides a free alternative to expensive programs, such as Photoshop. Goelker's book for beginners will take the reader from a sorry, unenlightened state directly to GIMP nirvana, covering topics, such as the basics of image editing, layers and masks, stitching panoramic images and preparing high-quality black-and-white images. The book follows a workshop format and has evolved from classroom materials that the author developed and taught in courses on image editing with the GIMP.

www.rockynook.com



Sarath Lakshman's *Linux Shell Scripting Cookbook* (Packt Publishing)

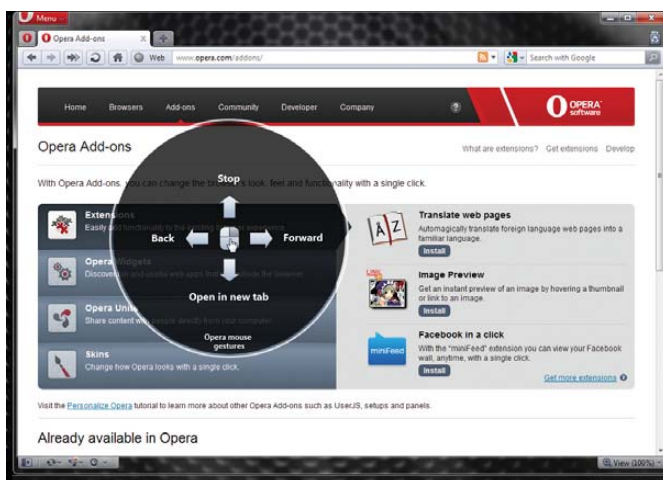
Scripting on a Linux/UNIX box is like playing an instrument. Okay, Sarath Lakshman says its more like cooking, and now that he has a new book out, called *Linux Shell Scripting Cookbook*, he gets to choose the metaphor. Regardless of whose metaphor you like, you may want to pick up Lakshman's book to master the powerful Linux shell scripting language, your tool for controlling the entire operating system. Written in a cookbook style, the aim of the book is to boil the lengthy man pages down into essential command-line recipes that cover most Linux commands and offer a variety of use cases and examples. Some of the many complex data manipulations covered include text processing, file management, backups and more. Utilities, such as sed, awk, grep and cut, also are covered.

www.packtpub.com

ADLINK Technology's Ampro CoreModule 745

The newest small-form-factor, rugged single-board computer from ADLINK is the Ampro CoreModule 745. This stackable PC/104-Plus SBC allows OEMs in military, avionics, transportation and other rugged markets to add a state-of-the-art Intel architecture controller to their systems without the need for a custom carrier board. The module supports a range of Intel Atom processors, from the power-efficient N450 running at 1.66GHz to the performance-oriented dual-core D510. The Atom's two-chip solution architecture with integrated memory and graphics controllers permit excellent performance with very low power requirements. With a TDP as low as 9W, the CoreModule 745 simplifies cooling requirements and enables conduction-cooled solutions for small sealed enclosures in space-constrained applications.

www.adlinktech.com/ampro-extreme-rugged



Opera Browser

Could the longtime cult-favorite browser Opera finally make the big time with its new version 11? The company says that Opera 11 is big—a browser that, besides adding a layer of polish to features users have known and loved for a decade, will “change everything you know about browsing”. The key new features include tab stacking, numerous new extensions and visual mouse gestures. Tab stacking is a novel way to organize open tabs. Extensions personalize the browser and add functionality. Visual mouse gestures allow browsing commands via a flick of the wrist. Opera 11, with its 30% smaller footprint, is available for download on Linux, Mac OS and Windows platforms.

www.opera.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

Sweet Home 3D— Interactive Home Design

www.sweethome3d.com

This nifty program caused quite a storm at SourceForge.net some time ago, but now that the publicity has died down, I'd like to put it back in the spotlight. As an unusually easy-to-use 3-D design program, this is a project I've wanted to cover for some time.

To quote the Web site: "Sweet Home 3D is a free interior design application that helps you place your furniture on a house 2-D plan, with a 3-D preview."

Installation As far as library requirements go, I couldn't find much in the documentation aside from Java and working OpenGL drivers. However, even Java might not really be needed, as there's a Java Runtime Environment folder contained in the given tarball, so perhaps Sweet Home 3D will run without it being locally installed.

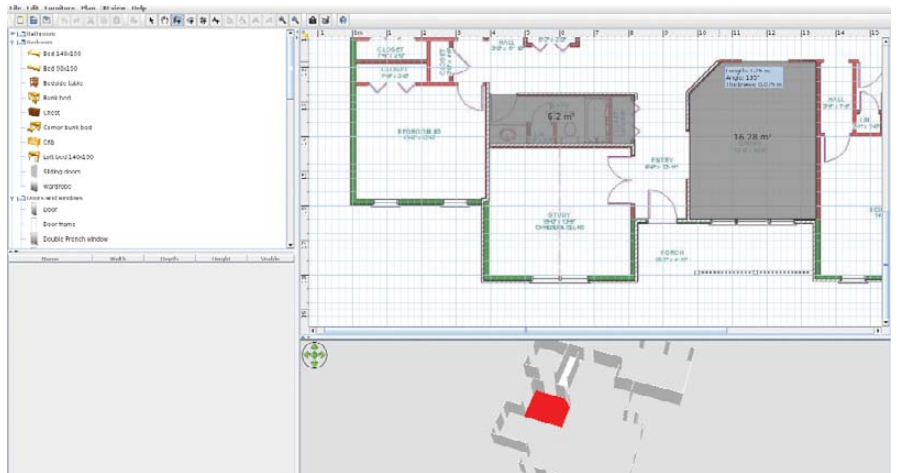
When it actually comes to downloading and running Sweet Home 3D, this is one of the easiest programs I've come across. Go to the Web site and head to the Download section. There you'll find a choice of 32-bit and 64-bit tarballs, with a source tarball also available at the bottom of the page.

Download the appropriate tarball for your system and extract it. In most modern file managers, you should be able to enter the new folder and click on the SweetHome3D file to run the program. However, if that doesn't work, open a terminal in the new folder, and enter:

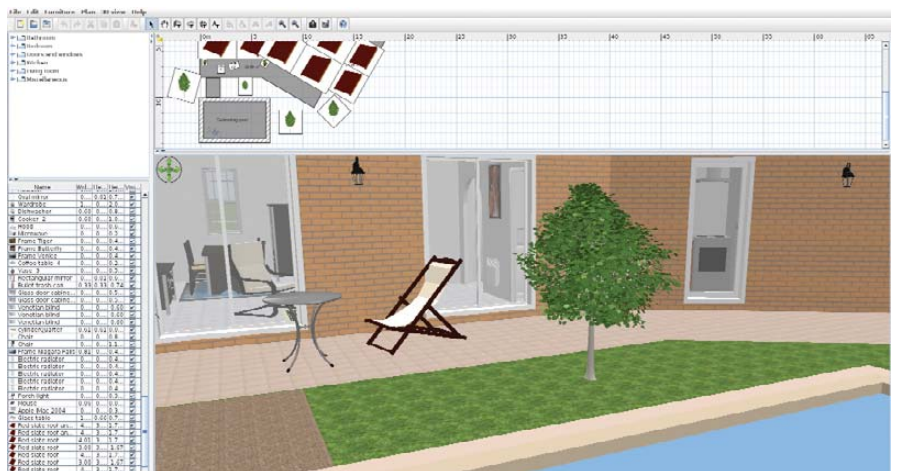
```
$ ./SweetHome3D
```

Usage Before I continue, I should mention that the Web site has a fantastic video tutorial that will unlock the working ideals of this program very quickly. The first thing it recommends is starting off with a blueprint image, and you then can draw over the top of it. You don't *need* one, but seriously, get one if you can.

To use the blueprint, from the main menu, choose Plan→Import background image. Choose the picture you want to import, and next you need to define the physical size from one end of the house to the other. You need the house's measurements for this, and there's a blue line below, which you drag onto one end of the house to the other. Although the



As soon as you start drawing walls on your blueprint, the house starts taking shape in the pane below it.

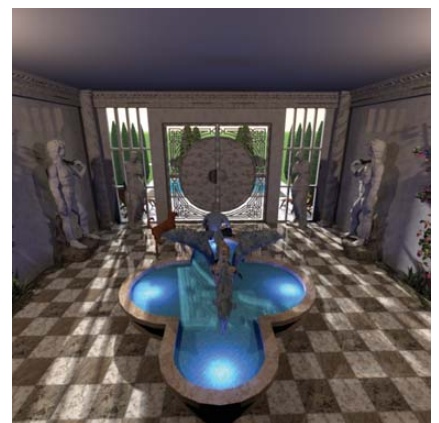


An example file for new users shows off some design possibilities, such as trees and a swimming pool.

default choice is in centimeters, it can be changed either to meters or millimeters, or you can set it to use the imperial system.

Next, you need to "Define the origin of the image, by clicking in its location below". See that blue thing in the top-left corner? Drag it over to the top-left corner of the house (you may want to maximize this window to place it more accurately).

Once you've made your way through these dialogs, choose the Create walls button, and click on the edge of a wall in the blueprints to start drawing. Each click you make defines a corner and allows the wall to change direction. When you reach the end of a wall, such as an archway for instance, either double-click or press the Esc button. Click the mouse again at the



If you export as a picture, Sweet Home 3D can enhance the image greatly. Here's a spectacular design, copyright Whippetsleek.

start of the next wall to resume the process. As you draw walls over the 2-D image, you'll see a 3-D house begin to take shape in the pane below it.

Doors and windows can be dragged from the pane on the left to the main plans on the right. Sweet Home 3D is exceedingly clever at working out how long a door or window should be, as well as which way it should open—the program actually works it out on the fly as you hover your mouse over different sections of the house. Even if you get the placement wrong, each object has its own mini-UI for readjusting the placement.

The Create rooms button allows you to define individual rooms within the house (cleverly independent of whether walls are there to define the area). This function's main use is really for things like texture and color, where one room can have different walls and flooring from another.

The left pane also has a selection of household items, such as chairs, washing machines and so on. If you're copying the layout of your own house, this is a great tool for testing out how a new item is likely to fit or look in your room before you actually buy it.

The coolest feature is that the bottom pane on the right actually lets you fly around your house using the mouse wheel or arrow keys, and you even can take snapshots or record a video.

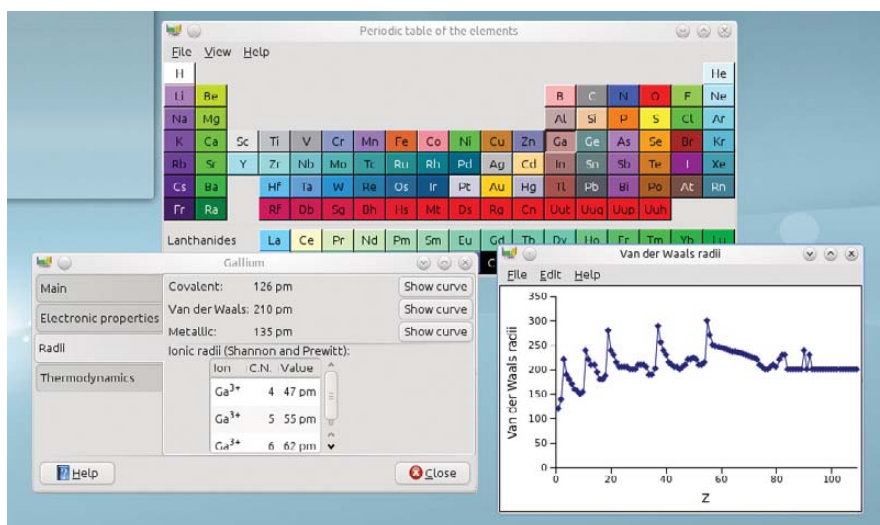
Ultimately, Sweet Home 3D was designed by passionate people who knew what they were doing. The UI takes certain liberties in 3-D design controls, fast-tracking you with areas that will make a house and leaving out some unnecessary clutter that often makes 3-D design a complete headache. This is one cool project.

GNOME Chemistry Utils

gchemutils.nongnu.org

Unlike the majority of projects I review, this one actually is a set of multiple programs, presented as one suite. Whether you're into crystalline structures or designing chemical molecules, this may be the project you're looking for.

Installation Available at the Web site are source tarballs for both the stable and development versions, and older versions are available in distro repositories. I ran into compiling problems with the development version, so I went with the stable version. Library requirements are quite extensive. According to the documentation,



An excellent tool for any scientist, GChemTable provides not only a table of elements, but also a whole swag of info about each element.

“to compile and use the GNOME Chemistry Utils, you need libglade-2, goffice, GtkGLExt, OpenBabel, BODR and their own dependencies. Everything except OpenBabel is available from the GNOME repository or one of its mirrors.”

I found a few more needed libraries, from both the configure and make stages. Although listed above, libgoffice,

Usage As mentioned before, this is a suite of utilities rather than just one, so I take a brief look at each utility here. Below each utility's title, I provide the shell command to run the program. Unfortunately, partway through reviewing this suite, I managed to “break” several utilities, seemingly beyond repair. No matter which files I wiped, or how many

Click on the Isotopic Pattern tab, and you can see the monoisotopic mass as a direct number and also displayed in a line graph, which can be exported as an image.

libopenbabel and libgtkglext1 all needed their development (-dev) packages installed to continue. Other needed packages were libgsf-1 and its -dev, libgcu0 and its -dbg, and chemical_mime_data.

Assuming you have the tarball, extract it and open a terminal in the new folder. From here, it's the standard fare of:

```
$ ./configure
$ make
```

If your distro uses sudo:

```
$ sudo make install
```

If your distro uses root:

```
$ su
# make install
```

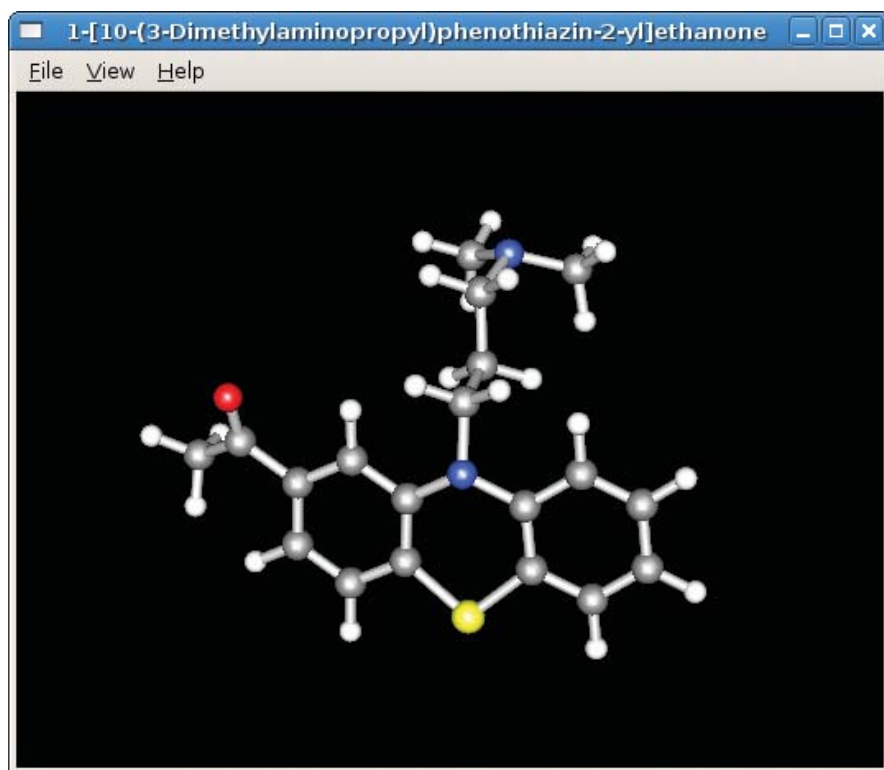
re-installations I tried, I couldn't get the programs working again. Therefore, reviews will be brief (and possibly theoretical).

GChemCalc:

```
$ gchemcalc
```

“GChemCalc is a simple calculator for chemists”, according to the About page. Below the main menu is a field for entering formulae, and GChemCalc will try to interpret the symbols you enter, either by atom or nickname. Once the formula is entered, its raw formula form will be written below, as well as its molecular weight. In the main field, the composition will be displayed according to each element along with its percentage of mass.

Click on the Isotopic Pattern tab, and you can see the monoisotopic mass as a



From the Web site, a great example of using the chemical viewer.

direct number and also displayed in a line graph, which can be exported as an image.

GChemPaint:

\$ [gchempaint](#)

I got far enough to realize that structures you build in this program can be used by most of the other programs. I'd just attempted to re-create an LSD molecule from the Wikipedia diagram, but the program crashed when I saved it as an .mdl file.

To start designing molecules, pick an object from the tool window, such as Add a four membered cycle, and you now can "draw" in the white workspace. Click and *hold* the mouse, and if you move the mouse around, the still-colored object will rotate until you release the mouse button, then the object will go black and stay in place. Although its operation seems somewhat arbitrary, moving forward and backward with the mouse will make the object larger or smaller (although you might need some perseverance to get it working).

As far as symbols and their placement, something like C or N will update

its appropriate symbols as you continue to expand on the molecule, balancing itself out (as far as I can tell anyway, I'm not a chemist).

Once you have a structure you're happy with, you can open it in GChem3D, allowing you to see it as an actual 3-D structure, instead of bland 2-D lines.

Something I'd have liked to explore more was a button marked: "Export for Wikipedia publication". I'm guessing that gets used a lot.

GChem3D:

\$ [gchem3d](#)

Unfortunately, I killed GChem3D when opening a bad file (the first of the casualties) and didn't get a chance to take it for a test run. The man file says: "gchem3d is a small chemical viewer application, which can show several chemical file formats". From what I can tell from the other documentation, this takes place in an OpenGL 3-D window, letting you rotate and interact with each structure, as well as customize the visual atomic representation.

Gnome Crystal:

\$ [gcrystal](#)

According to the man file, "gcrystal is a light model visualizer for crystal structures, based on the GNOME Chemistry Utils, that displays models of all sorts of crystal microscopic structures using OpenGL". Although I didn't break this program, I couldn't make sense of it as a layman; however, the concept appears to be similar to that of GChem3D, but for viewing and editing crystalline forms rather than viewing molecular structures.

GChemTable:

\$ [gchemtable](#)

This gives you a periodic table of elements that has a great database of information available for each element. Click on an element, and you'll be given four tabs. The first, Main, consists of the element's name (which is also translated into several other languages on the right), the atomic number, atomic weight and electronic configuration. The other tabs give you the electronic properties, radii and thermodynamics.

GSpectrum:

\$ [gspectrum](#)

According to the man file, "gspectrum is a simple spectrum visualizer. It is only able to display files in the JCAMP-DX format". This was another casualty of mine, but it appears to display JCAMP spectra on a basic X-Y scale.

This was the unluckiest month I've had at LJ. I think I went through a dozen programs that wouldn't work before I came across these two projects. Although half the programs in this suite died on me (in a way I've never encountered before), I'm sure distro-stable versions will be a lot more reliable. Bad luck aside, the GNOME Chemistry Utils suite seems to provide viewing, access and editing to some file types that are used in important scientific circles. That alone is worth this project's perseverance. ■

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

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

DON'T MISS **MAR**
THE WORLD'S LARGEST **9TH**
PYTHON GATHERING **THROUGH**
17TH
PYCON 2011, ATLANTA

us.pycon.org



PyCon

Connecting The
Python Community

PyCon is a volunteer-run conference by and for the worldwide Python community. The ninth annual PyCon is organized and hosted by the Python Software Foundation.



HARDWARE

Barnes & Noble's NOOKcolor

NOOKcolor: E-reader or Internet tablet? BILL CHILDERS

The march of technology never ends. Not long ago (in the December 2010 issue of *Linux Journal*, actually), I reviewed the Android-powered NOOK e-reader from Barnes & Noble. That product was an E Ink-based product, running Android 1.5. What a difference a few months can bring. Barnes & Noble has just released its second-generation e-reader, the NOOKcolor. As one can gather from the name, the NOOKcolor's immediate distinguishing factor is a 7" color touchscreen, while the original NOOK had a 6" black-and-white E Ink screen.

NOOKcolor—an Overview

Although the original NOOK was a decent enough e-reader and could be hacked to add additional functionality, the NOOKcolor borders on being an actual Android-powered tablet. Table 1 compares the specifications for the two units.

Looking at the specs, you can see that the NOOKcolor is a much more capable device. Unlike the original NOOK, the NOOKcolor is much hungrier for battery

life—that excellent color IPS screen gobbles a lot of power. The NOOKcolor can run for 8–10 hours on a charge, while the original NOOK was good for a couple days if I kept the wireless radios off. The NOOKcolor charges via the Micro USB connector on the bottom edge of the unit, much like the original NOOK did. However, unlike the original NOOK, the NOOKcolor *must* be plugged in to its wall charger to charge fully. A computer's USB port will supply only 500ma, which isn't enough to charge the NOOKcolor under operating conditions and will only slightly trickle-charge the unit when it's asleep. The NOOKcolor's wall charger puts out 2000ma, which is enough to charge the unit fully in about three hours. There is a little "n" logo embossed in the Micro USB cord end that glows with the charging status of the NOOKcolor (yellow for charging, green for fully charged), which is a nice touch.

As with the original NOOK, the NOOKcolor can read several media types:

- EPUB e-books (with and without DRM).



Figure 1. The NOOKcolor

- Mobi/eReader e-books (with and without DRM).
- PDF, XLS, DOC, PPT and other office-related formats.
- MP3 and AAC audio files (can be played in the background while reading).
- JPEG, GIF, PNG and BMP image types.
- MP4 video playback (new to NOOKcolor).

The NOOKcolor also includes a fairly capable, much improved Web browser over the original NOOK, and it ships with Pandora Internet Radio, a chess game, a sudoku game and a crossword puzzle game.

Using the NOOKcolor

I've never covered an "unboxing" ritual before, but I feel compelled to mention it with the NOOKcolor. Your first impression of the NOOKcolor is when you open the box, and Barnes & Noble really did a great job on the packaging. It's a cardboard box that utilizes a magnetic closure to hold the lower one-third of the box closed. You pivot the lower one-third back, and it'll bend back and is held against the back of the box by another magnet, where you can reach inside the box and pull the NOOKcolor out of its recycled cardboard sarcophagus. It's wholly unnecessary, but absolutely awesome in its execution, and it gives you a clue as to what Barnes & Noble is thinking: the NOOKcolor is no device to be trifled with.

Table 1. NOOK vs. NOOKcolor

	NOOK	NOOKCOLOR
CPU	Samsung S3C6410 (533MHz)	TI OMAP 3621 (800MHz)
RAM	256MB	512MB
Storage	2GB int., ext. MicroSD	8GB int (5GB usable), ext. MicroSD
Network	Wi-Fi (Wi-Fi + 3G on 3G NOOK)	Wi-Fi only
Screen	600x800 E Ink 6", 480x144 color	600x1024 7" color IPS
OS	Android 1.5	Android 2.1
Kernel	Linux 2.6.27	Linux 2.6.29
Size	7.7" x 4.9" x 0.5"	8.1" x 5.0" x 0.48"
Weight	12.1 oz	15.8 oz
Audio	Mono speaker and headphones	Mono speaker and headphones



Figure 2. Unboxing the NOOKcolor

The initial boot of the NOOKcolor drops you into a wizard that walks you through your Barnes & Noble store account and initial Wi-Fi access point setup. The unit also displays a short video that shows you the NOOKcolor's features. The NOOKcolor doesn't have much in the

way of hardware buttons—only a power button on the left edge, volume rocker on the right edge and an n button at the bottom center of the screen. If you push the n button, you'll be taken back to the home screen, where you can select any books on the NOOKcolor's desktop.

There's also a little arrow on the screen, right above the n button. Its normal mode is up, but when pressed, it rotates down, and a pop-up menu is exposed. This is the NOOKcolor's main menu. From here, you can select Library where you can choose your reading material. The Shop button connects you to the Barnes & Noble on-line store. The Search button is, well, a search. Extras is where the other NOOK programs are, such as Pandora, sudoku and chess. Web launches the Web

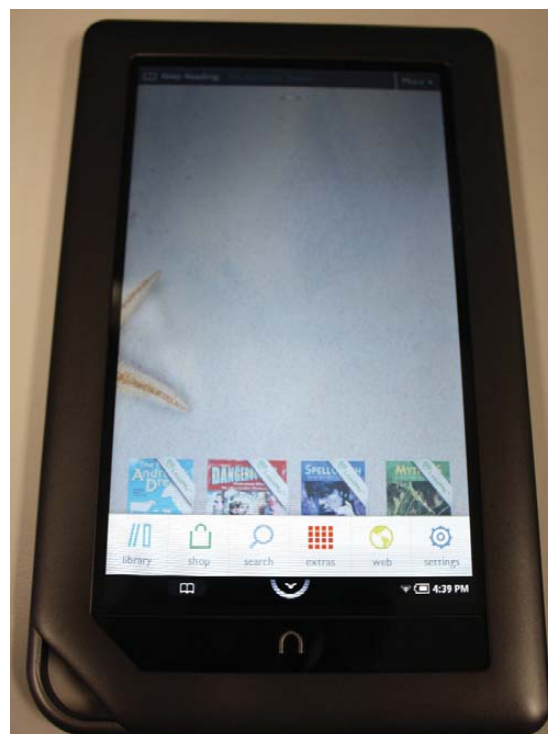


Figure 3. NOOKcolor's Main Menu

SILICON MECHANICS

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



Charles heads the web development team here at Silicon Mechanics: he's responsible for the configurators and power calculator on our site, among other things. As a software expert, he offers a useful perspective on our server and storage products.

When asked what he would do if he had a Rackform iServ R413 configured with 4 8-core Intel® Xeon® Processor 7500 Series CPUs and 32 DDR3 DIMMs, he said, "32 virtual machines . . . one per core . . . in one rack unit." But he didn't stop there.

Charles knows that to make the best use of a server with that kind of processing horsepower in a virtualized environment, he needs I/O to match. He paired the 4P server with a Storform iServ R516 storage server, configured with 24 2.5-inch Intel X25-E solid state drives. Think of it as a developer's dream team: multi-core processing and high memory counts for blistering performance, and high-performance storage for blazing I/O speed.

Need a "dream team" of your own? Talk to the Experts at Silicon Mechanics for the perfect match.

When you partner with Silicon Mechanics, you get more than just the power and performance of the latest Intel technologies—you get an Expert like Charles.



**Powerful.
Intelligent.**

For more information about the
Rackform iServ R413
visit www.siliconmechanics.com/R413

Expert included.

Silicon Mechanics and the Silicon Mechanics logo are registered trademarks of Silicon Mechanics, Inc. Intel, the Intel logo, Xeon, and Xeon Inside, are trademarks or registered trademarks of Intel Corporation in the US and other countries.

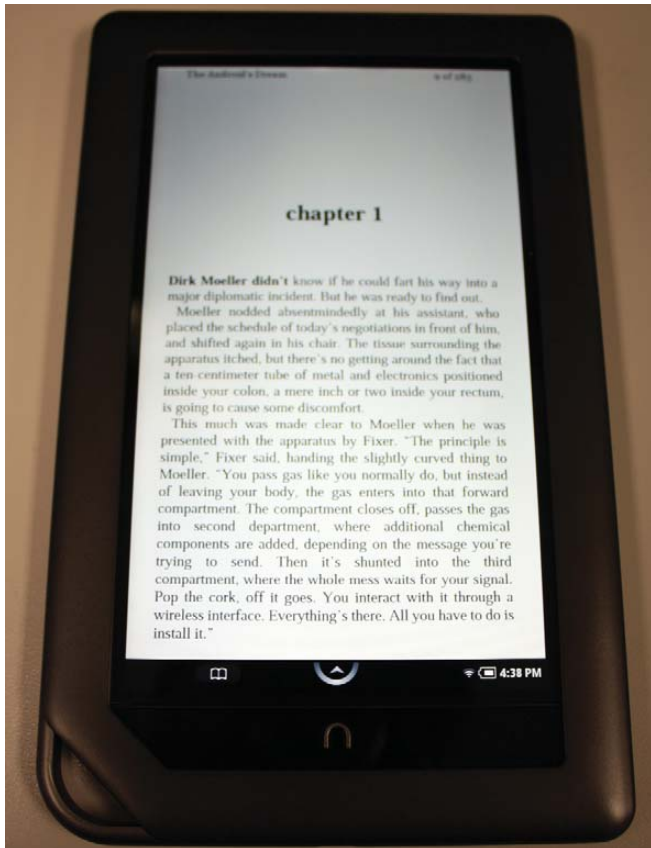


Figure 4. Reading *The Android's Dream*



Figure 5. *Linux Journal* on the NookColor

browser, and Settings is where you can adjust system settings like Wi-Fi, screen brightness and so on.

As good as a reading experience as I found the original NOOK, the NookColor is better in almost every way. The instantaneous response of the IPS color touchscreen is a treat after the perceptible wait of the

E Ink screen of the original NOOK. Although I found the original NOOK acceptable, not having the flash of the screen and the page flipping delay is quite nice. I'm fortunate in that I'm not prone to eye strain, so looking at the NookColor's screen wasn't an issue for me, but if you're prone to that, you might want to test-drive a NookColor at

your local Barnes & Noble for a bit. The one place the color screen is a liability is in bright light or sunlight. It just can't compete with the E Ink screen of the original NOOK under those conditions. Personally, I prefer the vivid, backlit screen of the NookColor. You can see it in Figure 4, with John Scalzi's excellent book *The Android's Dream* displayed (get it if you haven't already!). All the original NOOK features for highlighting, bookmarking and instant dictionary lookup are present and accounted for.

Page flipping on the NookColor is a completely different experience from the original NOOK too, as the unit has no hardware page forward/reverse buttons. Instead, you use the touchscreen to swipe across the screen, like you are turning the pages of a book. Alternatively, tapping the right side of the screen advances the page, and tapping the left side of the screen flips a page back. Unfortunately, there is no "page-flipping" animation—the screen simply displays the next page. I'd like to see a page-flipping animation while using the swipe gesture and have the current behavior on screen taps, as the shorter gesture implies a faster page turn.

Dangers of Rooting your NookColor

Obviously, anything you do to modify the NookColor voids your warranty instantly. Do not try to return it or pass off anything you've done as a factory issue. If you break it, you keep both pieces. You should be technically adept and have a good working knowledge of Linux before attempting any rooting processes. If you have any data on the unit, back it up, as you run the risk of erasing that data.

Having said all that, 90% of all the things that can go wrong with rooting a NookColor can be fixed by doing a factory restore of the firmware. To do that, simply turn the unit off, then press and hold the Volume Up key, n button, and power button until the NookColor turns on. You'll then be prompted with a Factory Reset wizard. Follow the steps to confirm the reset, and the NOOK will restore its firmware from a .zip file on a hidden partition of the internal Flash.

Please be sure to read the current instructions at NOOKDevs *before* attempting anything, as things are likely to change dramatically by the time this article is printed.



Figure 6. *Angry Birds* Running on the NOOKcolor

The NOOKcolor's on-line store experience is similar to the original NOOK, however, the vivid color screen really helps make the act of shopping a breeze. Again, Barnes & Noble built on its earlier technology and polished the edges of the on-line store, while keeping all of the good things about it, like automatic downloading of content in the event you replace your NOOK.

The Web browser is one area in which the old NOOK needed lots of work, and Barnes & Noble delivered. In fairness, it's the stock browser from Android 2.1, but the old NOOK's browser was so bad, the new browser beats it hands down. The new browser also scores 93/100 on the Acid3 test—not fully compliant, but it works a lot better than the browser in the original NOOK. It's this browser that makes you think the NOOKcolor may have more to it than meets the eye.

More than Meets the Eye—Rooting the NOOKcolor

If you've been following along, you'll see that the NOOKcolor's a very capable

device, hardware-speaking, and it runs Android. The folks at NOOKDevs thought the same thing, and they started poking at the NOOKcolor. They decided to attempt to root, or gain control of the hardware outside any blocks put on it by the manufacturer. They found that the device can be booted off a properly formatted MicroSD card. If they put their own OS on it, they could, in theory, boot off the card, mount the NOOKcolor's internal disk and make changes to it, allowing for installation of other software and modification of the unit. This is exactly what they did, and it turns out the NOOKcolor makes a very capable Internet tablet. Because this is a highly moving target at the time of this writing—literally, advances are being made almost on an hourly basis—I'm not going to do a step-by-step account of the method. Rather, I touch on the concepts here, and for current details, check out the NOOKDevs portal Web site (see Resources).

The method to root the NOOKcolor at the time of this writing is to use a specially

prepared image of a MicroSD card. Once you get that image, you'd write it to a MicroSD card, then boot the NOOKcolor off the card. The NOOKcolor will not have anything on its display during the boot, so you need to have a bit of faith and wait a while, then pull the card and reboot the unit. If all goes well, once the NOOKcolor reboots, you should be able to talk to it using the `adb` command found in the Android SDK. If the result of `adb devices` shows a serial number, you're talking to your NOOKcolor via the debugger, and you can sideload programs in the Android .apk format to it! Anything you add to the NOOKcolor appears in the Extras menu, so it's easy to get to and launch those new programs.

At this point, you can start pushing software to the NOOKcolor. Quite a few sites host Android freeware on the Internet (one of them is in the Resources section of this article). You even can get the ad-supported free version of the hit game *Angry Birds* running on the NOOKcolor. It runs flawlessly, and it's an excellent time killer. At this point, the sky's the limit. You can swap out launchers for alternate launchers and modify the system further (see NOOKDevs for examples of what you can do).

So there you have it—the NOOKcolor is a good e-reader and a very hackable device. If you're in the market for something fun to play with that has good build quality and lots of potential, check out the NOOKcolor at your local Barnes & Noble. It's a very nice device. ■

Bill Childers is an IT Manager in Silicon Valley, where he lives with his wife and two children. He enjoys Linux far too much, and probably should get more sun from time to time. In his spare time, he does work with the Gilroy Garlic Festival, but he does not smell like garlic.

Resources

Barnes & Noble's NOOK Portal:
www.nook.com

NookDevs NOOKcolor Portal:
nookdevs.com/Portal:NOOKcolor

Android SDK Download Page:
developer.android.com/sdk/index.html

Android Freeware:
www.freewarelovers.com/android

BREAKING NEWS

At the time of this writing, many things still are being discovered about the NOOKcolor. It appears that the chipsets inside the unit have Bluetooth functionality, and that it's simply disabled in the kernel and OS. There's strong evidence of that, but I was unable to get the reported Bluetooth operating, although others have claimed success with basic pairing to other Bluetooth devices. Check in on the crew at NOOKDevs for the latest news on what's going on with the NOOKcolor.

SOFTWARE

Radical Breeze's Illumination Software Creator

It runs on Linux! It runs on Flash! It runs on Android! It's Illumination Software Creator!

MIKE DIEHL

Wouldn't it be cool if you could create a program without having to write any code? And, wouldn't it be cool if your program could run under Linux, Mac, Windows, Android, Windows Phone 7, iPhone and Flash? It would be very cool, and with Illumination Software Creator from Radical Breeze (radicalbreeze.com), you can do exactly that.

Illumination Software Creator (ISC) is a graphical IDE that allows you to drag code blocks and connect them in order to create an application program. Figure 1 shows what the user interface looks like. It's pretty straightforward. To create a program, you simply drag code blocks onto the canvas and configure the block. Then, you connect outputs to inputs. The code blocks are categorized to make them easy to find. Clicking on a code block allows you to set various parameters for the given block. Variables, which appear in the lower-left panel, come in three types: text, number and text file.

Once an application program has been created with ISC, it can be targeted toward any of the architectures mentioned previously.

By default, ISC creates a program targeted toward a Python/GTK environment. This is the most feature-full target and allows programs developed with ISC to run on Linux, Mac and Windows. This target allows windows to have various dimensions and supports file I/O and shell script access where available on the host system.

The Android target creates an Android project, with complete Java source code, suitable for importation into the Eclipse IDE. As I'm not a regular Eclipse user, I had some difficulty getting the Android SDK properly configured. However, I was able to look at the Java code that ISC produced and then get the Android phone emulator working. I'm sure that a regular Eclipse user would be up and running in a matter of minutes, once the software was downloaded.

On the other hand, the iPhone target creates an iPhone project with native Objective C source code. However, because of Apple's licensing policies, this project directory has to be moved to actual Apple Macintosh hardware for compilation and linking against the Apple SDK.

Finally, the Flash/Flex target produces a directory that contains an .swf file as well as an HTML file that embeds the Flash object. The command-line Flash player, `swfdec-player`, ran my test applications without error, subject to the limitations of the Flash architecture. For example, Flash doesn't support file I/O, so my application's attempt to open a file selection window silently failed. However, the example Flash applications ran flawlessly in my Web browser.

As you can see, re-targeting ISC causes it to produce native code for the given target as opposed to simply linking against a proprietary runtime environment. Also, applications produced with ISC should run identically on whichever target they are deployed. Each code block, though written in different languages, produces identical results on each target. However, application windows you create with ISC default to a size appropriate for each target architecture; this prevents large windows from being clipped on mobile devices smaller screens. You could think of this as object-oriented programming on steroids—not only is the actual implementation of a code block encapsulated, but the target architecture is as well.

So far, I've been talking about code blocks in fairly generic terms. So what kinds of blocks do you have to work with? Of course, the most important block is the Window block. This block creates an application window in which you can put buttons, labels, form fields and text fields.

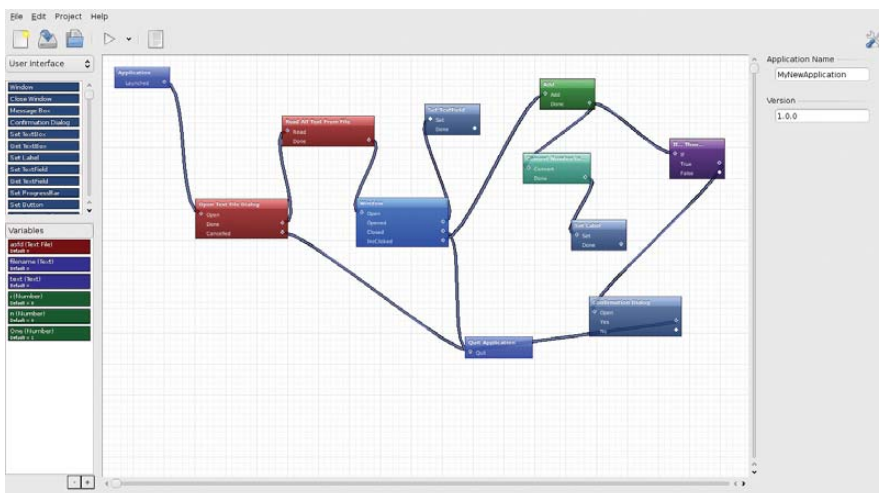
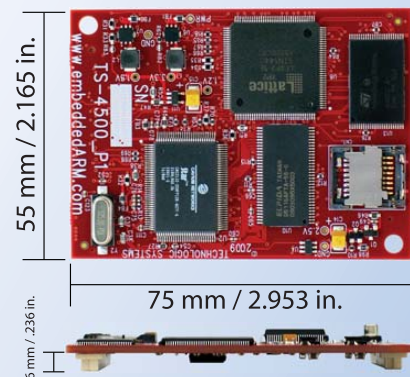


Figure 1. Illumination Software Creator's User Interface

TS-SOCKET Macrocontrollers


Jump Start Your Embedded Design

Series starts at **\$92** qty100



TS-SOCKET Macrocontrollers are CPU core modules that securely connect to a baseboard using the TS-SOCKET dual connector standard with common pin-out interface. COTS baseboards are available or design your own baseboard for a custom solution with drastically reduced design time and complexity. Start your embedded system around a TS-SOCKET Macrocontroller to reduce your overall project risk and accelerate time to market.

- TS-4200: Atmel ARM9 with super low power
- TS-4300: Cavium ARM11 with dual 600 MHz and FPU
- TS-4500: Cavium ARM9 at very low cost
- TS-4700: Marvell PXA168 with video and 1.2 GHz CPU
- TS-4800: Freescale iMX515 with video and 800 MHz CPU
- Several COTS baseboards for evaluation & development

 Design your solution with one of our engineers

- Over 25 years in business
- Never discontinued a product
- Engineers on Tech Support
- Open Source Vision
- Custom baseboards w/ excellent pricing and turn-around time
- Most products ship next day



We use our stuff.

visit our TS-7800 powered website at
www.embeddedARM.com
(480) 837-5200

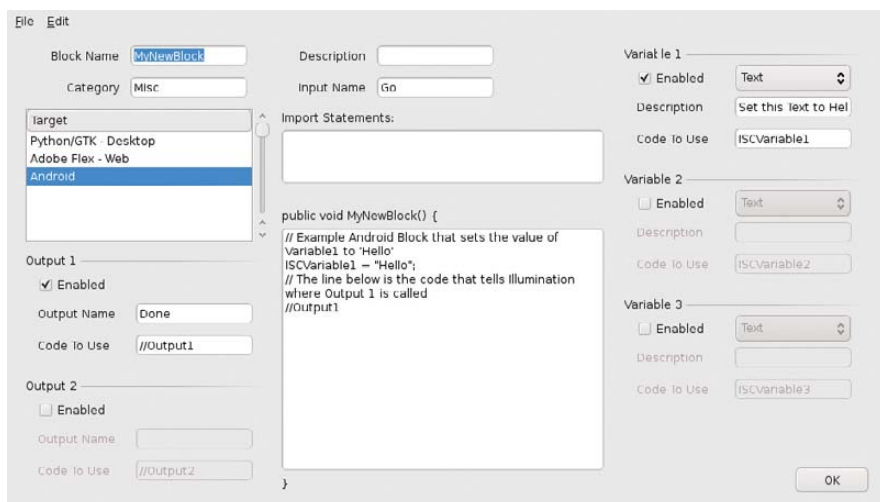


Figure 2. Custom Code Block Definition Screen

This block essentially creates the “face” of your application. You can create as many windows as you need and open and close them according to your application’s needs. You also can create message boxes and confirmation boxes. Additionally, there are blocks that allow you to get and set the value of the various window controls.

The native text manipulation blocks are fairly limited. There are concatenation, search and replace blocks, as well as blocks to convert to uppercase and lowercase. This was a surprising weakness in ISC, but one that can be worked around, as I show later.

Rather than a generalized numerical expression evaluator, ISC provides blocks that implement individual arithmetical operators. Here, you’ll find add, subtract, multiply, divide and remainder. You’ll also find a block that provides a random number in a given range. I am, however, talking about integer arithmetic, with all of the inherent limitations. That said, practical program applications were written long before the advent of floating-point processors.

There are but two logic blocks in ISC: the if/then/else block and a simple do/while block. As limiting as that may sound, these two blocks should be sufficient to implement just about any algorithm I can imagine. However, these blocks would be much more powerful if ISC had an expression evaluation capability.

ISC is what could be referred to as a “strongly typed” environment. Number variables must be converted explicitly to a text format before they can be used in

string operations and vice versa. This makes some sense given that the underlying variable implementation has to work identically in a Python, Java and Objective C environment.

The Run Shell Script code block allows an application to run a local shell script on architectures that have an underlying shell interpreter. In practical terms, this means Linux and Windows. This opens up the possibility of simply forgetting about the cross-platform capabilities in ISC and using ISC merely to put a friendly face in front of an otherwise esoteric shell script. I’m told that many of ISC’s users are system administrators who use ISC to create user-friendly system administration tools for use by other staff members.

Since many of ISC’s target architectures don’t support file I/O at all, ISC’s advertised file capabilities are fairly limited. You’re able to create file open and file save dialog boxes. However, reading and writing files is on an entire-file-at-a-time basis. Essentially, you read the file, process it and write it back out. Of course, none of this works at all on targets like Flash that don’t support accessing local files.

Earlier, I mentioned that many of ISC’s limitations could be worked around or mitigated, and that’s where the custom code block comes in. Take a look at Figure 2. Here you see the custom code block definition screen. This is how ISC users create new types of code blocks. Starting from the upper left-hand corner and working down, notice that you can name and categorize a new code block. Then, you have to choose a target for the block, and this is where it gets interesting. You actually

can implement the same code block for each target architecture you intend to support. So, if you intended to support only the Python environment, you'd need to write out only the code block in Python. However, if you wanted to support other targets, you would implement your block independently for each target. This way, the same block can be used in your application, and the appropriate implementation will be used automatically. Custom blocks even can be saved out to their own files for use in other projects.

Continuing on, notice that you can define up to two "outputs" for your block. Custom blocks accept only one input, but you can define the name of that input. This actually makes some sense because the "input" isn't actually accepting any "input". It's just a connection point. The rest of the center of the custom block definition window is where the meat is, and potentially, where the work is. This is where you write the code that actually implements the function you intend for the block. Keep in mind that you have to write code for each target architecture you intend to support, in

With ISC, even beginning programmers can produce useful applications and have them run on their mobile devices.

that target's native language. As a hint as to what your code needs to look like, you are given example code for the currently selected target.

The right-hand section of the window describes up to three side effects that the block can have. Results of any calculations done in the code body can be sent to variables for use by the rest of the application. Which variables are modified is determined when you drag the resulting block onto the canvas and configure it. There doesn't seem to be any type of extensible API for modifying other variables.

So far, I've tried to point out as many limitations as I could find in ISC, but to be fair, ISC is maturing rapidly. When I set out to review this product, I received version 2.0. I found this version to be stable and usable, but it lacked the custom block feature entirely. The custom block feature appeared in version 2.2, which apparently was released a few weeks later. I've been told that version 3.0 beta 4, which will be available by the time you read this, will

have many new features. There will be a rudimentary graphics capability suitable for writing games. There will be a clock timer block that "fires" periodically. But the feature I'm personally most interested in is the SOAP-based Web services client block. This will allow an ISC application to have real-time access to data and advanced functions over the Internet. These features will greatly expand the range of applications that can be created with ISC.

The fact that ISC is developing so rapidly is astounding. Consider that every code block, feature and bug fix has to be implemented identically in at least three different programming languages. Each target architecture imposes its own limitations and way of doing things. For ISC to be a useful tool at all, it also must expose as much functionality as it can that is common to every target platform. This is a truly ambitious project, and the fact that the project is only two years old is simply amazing.

Radical Breeze also is planning to host a repository of user-created blocks. By the time you read this article, you'll be able to

go to this repository and download custom blocks other users have created. Based on what I read in the user forums, I would expect networking blocks and SQLite blocks to emerge pretty early. There won't be any guarantees that a given block will function on all the supported targets. But because you'll be downloading the source code for the block, missing functions and targets could be added fairly easily.

I'm told that school computer departments are one of ISC's largest users. Schools are using ISC to teach programming classes. Although I happen to be a fan of structured programming, which ISC doesn't support very well, I can see where ISC would provide a means of getting young students interested in computer programming. After all, the results of any project students write with ISC would run on their cell phones. This could be a tremendous draw, though I shudder to think about what types of applications the average high-school-aged

kid might produce.

Another common usage that ISC sees is as sort of a "jump start" program to allow an organization to prototype an application quickly in a desktop environment and then port it to mobile environments without too much additional investment. Even if the company eventually moved away from ISC and onto another IDE, having had ISC automatically generate much of an application's code could shave months off a delivery date.

With just a little prior thought, a non-functioning Flash-based demonstration program also could be created and posted on a Web site as a sales tool. Potential customers could interact with the demonstration applet on the Web and feel confident that the program will function the same way on their mobile devices or workstations.

"There's an app for that" has been a tremendously successful marketing tool for Apple. However, I frequently find myself saying "I need an app for that", and I don't think I'm alone. With ISC, even beginning programmers can produce useful applications and have them run on their mobile devices. Simple games and list managers are some of the first things that come to mind. But mortgage calculators and cost estimation software also would be trivial to produce with ISC. With version 3.0 beta 4 of ISC and its Web services capabilities, real-time point-of-sale and inventory-tracking applications become easy to implement on a wide variety of platforms. Having a program that behaves the same way on both a workstation and any number of mobile devices would cut down on training expenses and much of the development of a custom application could be done in-house.

When I committed to review this product, I had formed the impression that ISC would solve all of my inter-architecture development problems. Well, it doesn't, but it sure solves a lot of them. Rather than having to invest in becoming an expert on several different architectures, I can allow ISC to write much, if not all, of my code. And with a price tag of a mere \$49.59, I can't see how you possibly could go wrong. ISC does what they say it will do, and it is very cool. ■

Mike Diehl is a contract programmer and consultant in Albuquerque, New Mexico. Mike lives with his wife and three small boys and can be reached via e-mail at mdiehl@diehlnet.com.

SCALE 9X

The Ninth Annual
Southern California Linux Expo

Mark your calendars!

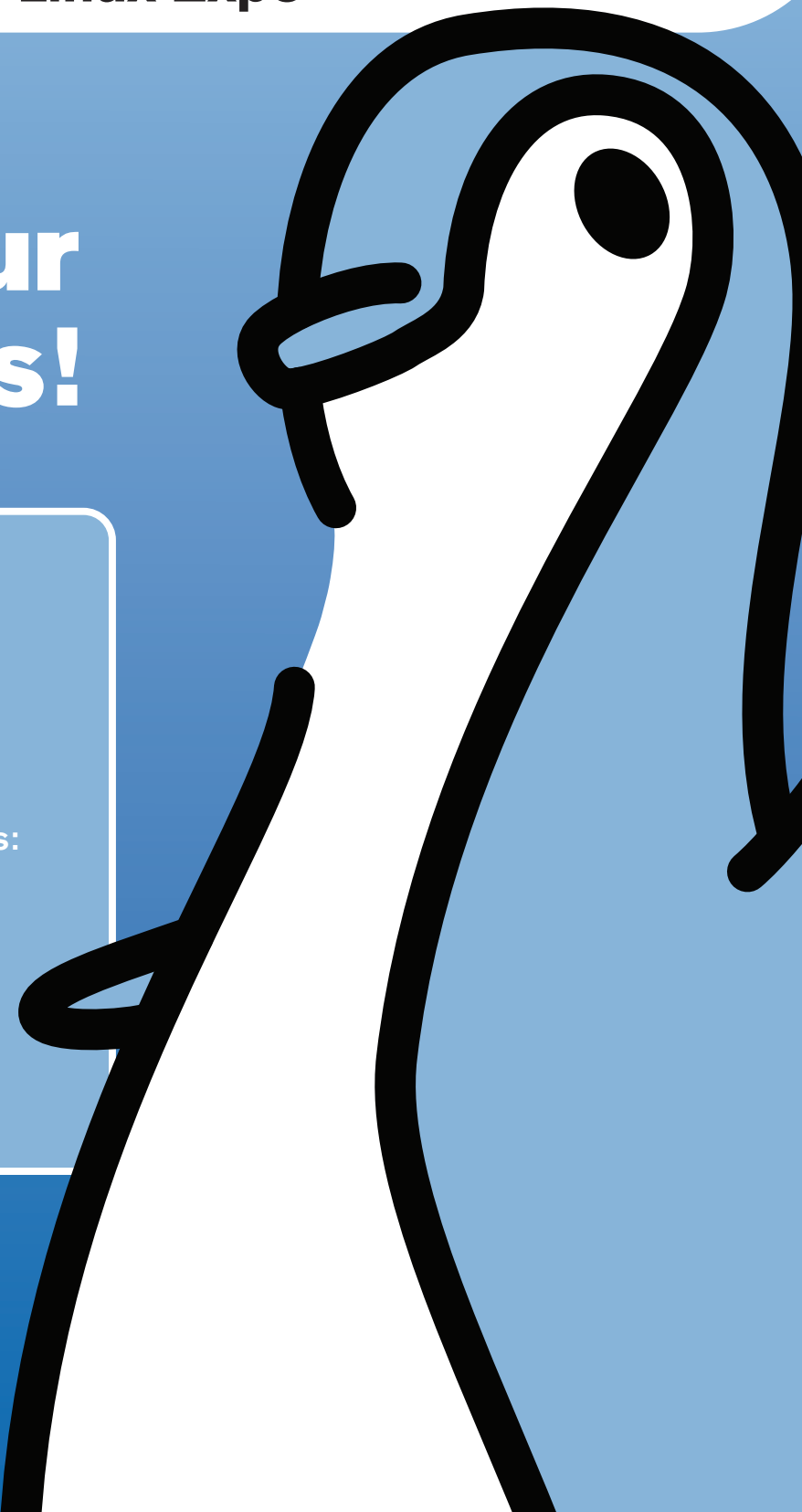
The 9th Annual
Southern California Linux Expo
is coming!

February 25-27, 2011

Expanded to five session tracks:
Beginner, Developer, SysAdmin
and two General Interest!

New, larger venue at the
Hilton Hotel @ LAX
Los Angeles, California

<http://www.socallinuxexpo.org>
Use Promo code LJAD for a 30% discount
on admission to SCALE



Python for Android

Think Java's the only game in town when it comes to programming Android apps? Think again.

PAUL BARRY



Mobile app development for smartphones is hot. This is no more prevalent than in the Android space where the activity level oftentimes is frenzied. However, when it comes to building a “real” Android app, it seems there’s only one programming choice: Java (although it is possible with a lot more work to use C/C++ with Android’s Native Development Kit). That said, Google wisely chose the popular Java programming technology upon which to base its Android SDK, which runs a customized VM.

By and large, this has been a smart strategy, as (unlike another popular smartphone) there’s no need to own specific hardware and software to get started with app development on Android. All you need is a PC (or laptop) running Linux, Windows or Mac OS X, together with a copy of Java and the free Android SDK. Google provides emulator downloads for all the Android platform

releases, and there’s even a free plugin for Eclipse to start you off and point you in the right direction.

That’s great—assuming, of course, you’re a fan of Java. If, like me, you’d rather eat glass than sit down to write some Java code in Eclipse, it would appear that you are out of luck when it comes to implementing your next project on Android. But, this is not the case. There’s a rather wonderful project called the Scripting Layer for Android (SL4A) that is bringing scripting languages to the Android platform and providing a working alternative to Java development.

In this article, I walk through the steps involved in preparing your computer for Android development with SL4A, then show how to write, test and run a simple script written in Python on your Android device.

Introducing SL4A

The Scripting Layer for Android is one of the many projects to see life as a direct result of Google's policy of allocating 20% of its employees' time to "pet projects". Damon Kohler works for Google, and he created SL4A to scratch his own itch when it came to programming Android. SL4A provides a high-level interface to Android's underlying Java technologies, exposing a subset of the API to scripting languages.

Python was one of the first languages to be supported on SL4A, but contributed interpreters also are available for Perl, JRuby, Lua, BeanShell, JavaScript and Tcl. In its default state, SL4A comes pre-installed with a working version of the shell. SL4A's API is designed to be portable across scripting languages, so if you like what you see in this article but wish I'd used Perl instead of Python, the API calls I use here should work exactly as shown with the Perl interpreter. It's true that I'm very much a fan of Perl, but I'm equally comfortable in Python, and, as Python is the SL4A "standard", I stick with Python throughout this article. Just be aware that you don't have to.

Getting Ready

To get started, you need a copy of the Android SDK running on your computer, and you need a Java VM. Although you aren't going to program your Android app with Java, you still need a Java runtime upon which to execute the Android emulator, which is part of the SDK. The inclusion of the Android emulator gives you a sandboxed testbed to play in while you create your app.

Before proceeding, you need to make sure a Java VM is installed on your Linux system. On my system (running a recent Xubuntu), I entered the following commands and was told neither program was available to me:

```
$ javac -version
$ java -version
```

Xubuntu suggested that I might like to use apt-get to install openjdk-6-jdk, which sounded reasonable to me, so that's what I did:

```
$ sudo apt-get install openjdk-6-jdk
```

If you are running a distro that's not derived from Debian, search your software repository for a similar package and install it before proceeding.

With Java in place, it's now time to get the Android SDK. Downloads of the SDK are available for Mac OS X, Windows and Linux. Pop on over to Google's Android Developer site (see Resources) and grab the latest SDK tarball for Linux.

Installing the Android SDK

With the SDK downloaded, simply unpack the tarball within a directory of your choice (the filename you have may be different from the one I use here, but don't worry, yours is likely a later version of the SDK):

```
$ tar zxvf android-sdk_r07-linux_x86.tgz
```

This command created a new directory called android-sdk-linux_x86, which I renamed to Android. This newly created directory has a bunch of subdirectories within it. Don't be

overwhelmed by all the stuff that's in there. Only one subdirectory is of interest to you at this stage, and it's called tools.

Preparing the Android Emulator

Within the tools directory, only two programs are of interest to you. The adb utility allows you to transfer files to (and from) your Android emulator (more on this utility later). The android tool lets you prepare and run an emulator for any number of the current Android releases, and that's what you do next:

```
$ cd Android/tools
$ ./android
```

The android command starts the Android SDK and AVD Manager, which is a tool that creates Android emulators for any number of Android virtual devices. At the moment, there are no virtual devices, so you need to create one. However, before this can happen, you need to install a target Android API package. To do that, click on the Installed Packages tab, then press the Update All button. In the dialog that appears, click Accept All and then press the Install button.

The resulting download takes a while, so drag yourself away from your computer and head off into the kitchen to make a nice cup of coffee. Depending on your download speed, you may have time to make a quick sandwich too.

With the download(s) complete, let's create an Android Virtual Device (AVD). An AVD is a virtual version of an Android device that you can run on the emulator. Select the Virtual Devices tab, then click on the New button. A dialog appears that you can use to set the characteristics of your AVD. Figure 1 shows that I've given my AVD a name (LJapp), selected an API target (API Level 7, which is Android 2.1) and set my virtual SD card to 256 meg. When you have made your selections, click on the Create AVD button to initialize your AVD.

If you return to the Virtual Devices tab, your LJapp AVD now should exist. Select the AVD and press the Start button, then select the Launch button from the displayed dialog. The emulator starts to load, and after a few seconds, you see the opening screen for your emulated Android device (Figure 2). Note that the emulator runs much more slowly than the actual physical phone, so it may take a while to get to the Android startup screen. Your mileage will vary, depending on the speed

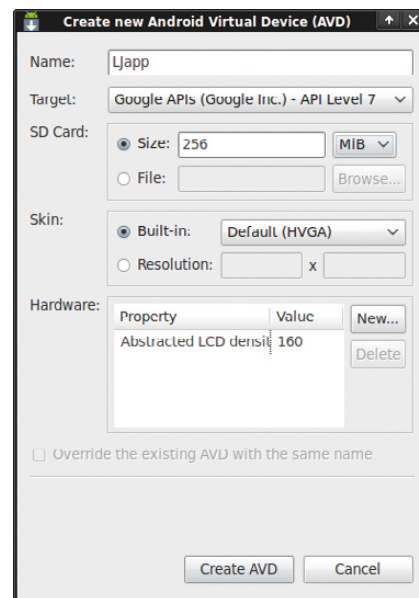


Figure 1. Setting the defaults for your emulated Android virtual device.

FEATURE Python for Android



Figure 2. Your Android emulator is ready.

of your desktop's processor.

Note: throughout the remainder of this article, I use the word “tap” to refer to using the mouse to click within the emulator. There is no mouse pointer on your physical Android device (that’s what your finger is for), but the emulator needs something, so the mouse pointer “emulates” your finger, hence the use of the word tap to describe the action required. When you see me use the word tap, think click.

Installing SL4A

Getting the SL4A on your emulator is straightforward. Start by tapping the Android browser and surfing to the SL4A Web site: code.google.com/p/android-scripting.

When the SL4A Web page appears, tap on the square graphic (the QR Code) to access the download URL for SL4A. The download should begin immediately, and once it completes, tap the downloaded package name (which is `sl4a_r3.apk` at the time of this writing) to install SL4A. Go ahead and tap on the Install button, then tap on the Open button to start the app. You’ll be asked to participate in usage tracking (it’s up to you, and it’s optional), and then SL4A informs you that you have “No matches found” under your Scripts listing. That’s okay; you’ve yet to install any scripts. SL4A now is ready for Python.

Installing Python for Android

Return to the Android browser and double-tap the screen to enlarge the Web page. Locate the Featured Downloads part of the page on the right, and tap on the download arrow beside the `python_for_android_r1.apk` link. As before, let the download complete before tapping on the downloaded file to install it. When you’re ready, tap the Install button to confirm the installation of the interpreter. When this completes, tap on the Open button, then tap on the big Install button to complete the installation. The Python for Android app downloads, extracts and installs the Python support files from the SL4A Web site and adds them into SL4A. The entire process should take only a minute or two to complete. If it takes longer than a few minutes, take a deep breath, and remember that the emulator is much slower than your physical device.

When the installation completes, you’ll be presented with an



Figure 3. Your Android emulator has SL4A installed and is ready for action.

Uninstall button. Don’t tap this button! If you do, you’ll remove the Python interpreter that you’ve just installed. Instead, tap on the emulator’s home button (the little house icon) to return to the Android main screen, tap the tab at the bottom of the screen to see the list of installed apps, and note that SL4A has been added to your emulator (Figure 3).

Go ahead and tap the SL4A icon that now displays the list of installed Python scripts. Tapping the `hello_world.py` script pops up the SL4A run menu (Figure 4). Looking from left to right, the menu icons have the following meaning: 1) run the script at the Python shell, which is useful when debugging; 2) run the script “natively”; 3) edit the script in SL4A’s built-in text editor, which is useful only for the most trivial of changes; 4) rename the script; and 5) delete the script. To test your installation quickly, tap the second menu icon, which I like to refer to as the “run wheel”. After a second or two, a message saying “Hello, Android!” appears on screen for a few seconds then disappears.



Figure 4. SL4A Menu Icons

If that worked, tap the `test.py` script, then tap the run wheel. This runs a longer-running script that showcases some of the facilities available to you as a Python programmer working on Android. Again, bear in mind that the emulator runs slowly, so be patient and wait for the various Android interface elements to appear.

Creating Scripts for Android

Your Android emulator now is ready to run your custom Python script, so let’s create one. Before you do though, note that the published SL4A API is a subset of the full Android API, so certain features either are not available, in the process of being made available or fully supported (see Resources for a link to the current list). Don’t let this put you off. What’s there is more than enough to produce usable Android apps in Python.

Linux News and Headlines Delivered To You



Linux Journal
topical RSS feeds
AVAILABLE

http://www.linuxjournal.com/rss_feeds

A Sample Script

To get a feel for Python running on SL4A, let's port an existing script to the phone. The script in question is based on some code from Chapter 2 of O'Reilly Media's *Head First Programming*, which I cowrote with David Griffiths in 2009. This simple script connects to the Web site of a fictitious company called Beans'R'Us to grab the company's home page and extract the current price of coffee beans from the page's HTML. The code is straightforward, grabbing the HTML page from the server, searching for the pricing data, extracting it from the HTML page and then displaying it on screen:

```
from urllib import urlencode
from urllib2 import urlopen

pg = urlopen("http://www.beans-r-us.biz/prices.html")

text = pg.read().decode("utf8")
where = text.find('>$')
start_of_price = where + 2
end_of_price = start_of_price + 4
price = float(text[start_of_price:end_of_price])

print "The current price of coffee is:", price
```

This is Python 2 code, which is a deliberate choice, as the Python that comes with SL4A is the 2.6.2 release. To take this program for a spin, either load it into Python's IDLE tool or execute it from the command line:

```
$ python LJapp-cmd.py
The current price of coffee is: 5.52
```

As you can see, this small script displays the currently published price of coffee beans.

Porting the Sample Script to Android

Turning this script into an Android app is just a matter of deciding on the Android UI elements you want to use, as the core functionality does not need to change. The Python on SL4A is fully functional, so the facilities you are used to with standard Python also are available on your smartphone.

To make this script more Android-like, let's display a friendly message on startup as well as one on exit. The `makeToast()` API call provides this functionality.

The `dialogCreateSpinnerProgress()` API call lets you display an Android spinner dialog, assuming you then remember to call the `dialogShow()` API call to make it visible. Let's display a spinner prior to requesting the Web page from the Beans'R'Us server, then dismiss the spinner dialog with the `dialogDismiss()` API call, once we have the data processed. And, let's vibrate the phone at this point too, just for the fun of it.

To conclude the script, use the `dialogCreateAlert()`, `dialogSetItems()` and `dialogSetPositiveButtonText()` API calls to display the price of beans within an Android dialog. To exit, simply tap the OK button.

Here's the Python code from earlier with the calls to the SL4A API added in:

```
import android

from urllib import urlencode
```

FEATURE Python for Android

```
from urllib2 import urlopen

app = android.Android()

app.makeToast("Hello from LJapp")

appTitle = "LJapp"
appMsg = "Checking the price of coffee..."

app.dialogCreateSpinnerProgress(appTitle, appMsg)
app.dialogShow()

pg = urlopen("http://www.beans-r-us.biz/prices.html")
text = pg.read().decode("utf8")
where = text.find('>$')
start_of_price = where + 2
end_of_price = start_of_price + 4
price = float(text[start_of_price:end_of_price])

app.dialogDismiss()
app.vibrate()

appMsg = "The current price of coffee beans:"

app.dialogCreateAlert(appMsg)
app.dialogSetItems([price])
app.dialogSetPositiveButtonText('OK')
app.dialogShow()
resp = app.dialogGetResponse().result

app.makeToast("Bye!")
```

Other than the addition of the Android UI code, no other changes are required to the code from earlier, other than removing the earlier script's call to print (which is no longer required).

Transferring Your Script to the Emulator

To transfer your Python script to the emulator for testing, copy your code file into your Android directory, then use the adb utility within the tools directory to push your file to the SL4A scripts directory on the emulator:

```
$ tools/adb push LJapp.py /sdcard/sl4a/scripts
6 KB/s (748 bytes in 0.116s)
```

With the file transferred, check the list of scripts within SL4A and notice the addition of LJapp.py near the top of the list.

Testing Your Script

Go on, tap your app name, then tap the run wheel. After a moment, you should see the spinner dialog appear while your script contacts the Beans'R'Us Web site and requests the current price of coffee (Figure 5). Shortly thereafter, the current price appears in another dialog (Figure 6). Congratulations! The script has been ported to your Android virtual device.

Running on a Smartphone

To run a Python script on your physical Android device, install SL4A together with Python for Android on your handset, then transfer your script.

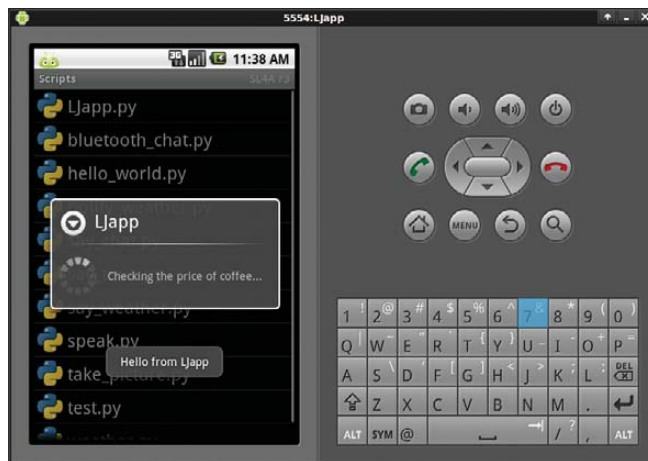


Figure 5. Your Python script is working.



Figure 6. And, there's the price of coffee beans. Is it time to place an order?

To install SL4A on your physical Android device, enable the Unknown Sources option in your device's Application settings. This setting is required to enable the installation of non-Market apps on your phone. With this done, you can follow the same steps you used when installing SL4A and Python on your emulator. To speed things up a little, install Barcode Scanner from the Android Market and use it to "read" the QR Codes from your desktop screen.

Transferring Your Script to Your Handset

There are a number of ways to get your script onto a real phone. I've found the success of using something like Bluetooth connectivity or USB cabling arrangements can very much depend on the hardware on which you're running. What works on one handset, doesn't on another, and so on. Your mileage may vary depending on your actual device. When I need to transfer a file, I've come to rely on a solution that works no matter which handset I use (as long as the handset can talk to a local Wi-Fi network). What I do is switch on the OpenSSH server on my development PC running Linux, then use the AndFTP file transfer app on the handset to scp files from the desktop to the phone. AndFTP is available from the Android Market as a free download and installs in

To run a Python script on your physical Android device, install SL4A together with Python for Android on your handset, then transfer your script.

minutes. Once I connect to my desktop with AndFTP, I can navigate to a directory of my choice, mark the files that I want, then download them to my SD card on the handset.

AndFTP works well, and I've come to depend on it for all my Android file transfers (see Resources). Just be sure to transfer your scripts to /sdcard/sl4a/scripts on the phone to ensure that your script names appear within the SL4A list of scripts.

With your script file transferred to your physical device, start SL4A as before, tap your app's name and tap the run wheel. As expected, your app runs just as it did on the emulator, only faster! I haven't included a screenshot of the app running on a real phone for two reasons. First, it looks exactly the same as it did in the emulator, and second, it's running on your device, so you can take a look at it there!

Creating an APK

There's one further kink to SL4A that might interest you. The project includes draft instructions on creating a standalone, downloadable Android APK package (see Resources). Once created, the APK file bundles your custom Python script with information that allows other Android users to install Python for Android automatically onto their handsets and then run your app from the smartphone's main menu of apps. Describing the process of creating the APK likely would take another article, so I leave it to the brave among you to try out the instructions on the SL4A Wiki. To see and learn a little about one such successfully created APK, check out Split Hitter on the Web (see Resources).

Final Words

Programming your app in Java is not the only option on Android. With SL4A, very capable scripting languages come to Android, taking the platform to a new level of hackability. If you can, get involved. More people means more eyeballs, and more eyeballs means more help and more code. Official support from Google will come if enough programmers get involved, which can only help raise the project's profile within Google's HQ. SL4A is already a good project, and I can only imagine the great project it will be with "official status" bestowed on it by Google. Let's hope this happens sooner rather than later.

You can learn lots more from the wiki documentation hosted at the SL4A Web site. Be sure to browse the Tutorials section of the wiki for links to what others are doing with SL4A. There's also an active (and very friendly) SL4A mailing list that Damon uses to announce new versions, track feature requests, report patches and spread the word about SL4A. If you build an app for Android using SL4A, join the list, and I'll see you there.

Acknowledgement

The scripts for this article were tested on an Android Milestone

handset running release 2.1 of Android OS. The handset was kindly supplied by The Institute of Technology, Carlow, in Ireland. ■

Paul Barry (paul.barry@itcarlow.ie) lectures at The Institute of Technology, Carlow, in Ireland. His latest book is *Head First Python*, which was published by O'Reilly Media in November 2010. Find out more about Paul at his Web site: paulbarry.itcarlow.ie.

Resources

Google's Android SDK: developer.android.com/sdk

SL4A Project: code.google.com/p/android-scripting

SL4A API: code.google.com/p/android-scripting/wiki/ApiReference

AndFTP Home Page:
www.lysesoft.com/products/andftp/index.html

Sharing SL4A Scripts as APKs:
code.google.com/p/android-scripting/wiki/SharingScripts

Split Hitter: www.splithitter.com

System on Module Internet Appliance Engine

SoM-9G45

- AT91SAM9G45 ARM9 400Mhz CPU
- 4 Serial Ports & 2 SPIs
- Up to 40 Digital GPIOs
- I2S Audio Interface
- 2 USB 2.0 Host/Device Ports
- 10/100 BaseT Fast Ethernet
- Up to 1 GB Flash & 256 MB RAM
- 2 SD/MMC Flash Card Interfaces
- Linux with Eclipse IDE & WinCE 6.0
- 8, 10-Bit A/Ds & 4 16-Bit Timer/Counters
- Graphic LCD Interface with 2D Acceleration
- Small, 200 pin SODIMM form factor (2.66 x 2.38")



2.6 KERNEL

The SoM-9G45 uses the same small SODIMM form-factor utilized by other EMAC SoM modules, and is the ideal processor engine for your next design. All of the ARM9 processor core is included on this tiny board including: Touchscreen Interface, Flash, Memory, Serial Ports, Ethernet, I2S Audio Interface, PWMs, Timer/Counters, A/D, Digital I/O lines, and more. Like other modules in EMAC's SoM product line, the SoM-9G45 is designed to plug into a custom or off-the-shelf Carrier board containing all the connectors and any additional I/O components that may be required. The SoM approach provides the flexibility of a fully customized product at a greatly reduced cost. Single unit pricing starts at \$190.

<http://www.emacinc.com/som/som9g45.htm>

Since 1985
OVER
25
YEARS OF
SINGLE BOARD
SOLUTIONS

EMAC, inc.
EQUIPMENT MONITOR AND CONTROL

Phone: (618) 529-4525 • Fax: (618) 457-0110 • Web: www.emacinc.com

Put the World in Your Pocket with Marble

KDE's Marble breaks free of the desktop—and it knows where it's going. **STUART JARVIS**



Have you ever wondered what it would be like to have the whole world at your fingertips? How about cradling it in a single hand, putting it in your pocket and taking it with you wherever you go? With KDE's Marble Virtual Globe on your mobile phone, you can do just that.

Yes, you now can run Marble on a smartphone that supports Nokia's Qt framework. That could be useful, the next time you are arguing with a friend about which is farther north: London,

England, or London, Canada. However, you could answer that kind of question using a mobile Web browser, or you even could use your phone to call someone and ask. At first glance, having a virtual globe on your phone seems like little more than a fun gimmick—something to play with and impress your friends. But if you look a little deeper, you will find that Marble can do much more than plug the gaps in your geography knowledge. It can help you find yourself and guide you home.



Figure 1. Marble's mobile version puts the world at your fingertips.

Navigate Freely

Marble has long been KDE's virtual globe application, providing satellite, atlas and OpenStreetMap mapping of the world. Since its 0.10 release last August, Marble has included route-finding capabilities, so you can enter two locations and have Marble calculate a route and provide step-by-step directions. However, with version 1.0, released in 2010, Marble's navigational capabilities and optimizations for mobile devices have really come of age. Combined with a suitable smartphone, this gives you a completely free navigation system—free software and free map data.

So, how can you use Marble to turn your phone into a personal navigational device? In principle, you can run the software on any device that supports Nokia's Qt toolkit. In practice, most testing is done on Nokia N900s running Maemo, and packages are available for this platform. Marble also will run fine on MeeGo when the first MeeGo-powered phones appear. In the view of lead developer Torsten Rahn, "It would be really cool to see Marble prepackaged for more platforms, such as Symbian, Android, WinCE and others."

Once you have Marble installed, you just need an excuse to play with it. Imagine that your employer calls you at home one morning and

wants you to travel to a client's office a couple cities away. Over breakfast, fire up Marble on your mobile phone, enter your current location or let Marble find it using GPS, enter your destination and wait for Marble to query a range of routing services before presenting you with a set of options, the best selected by default. If you want to avoid a road you know is particularly busy in the morning, you can choose one of the alternative routes or set an additional via point to avoid it. If you need to pick up your suit from the dry cleaning shop on the way, you can add that to the route too.

When you are happy with your journey plan, just use your home wireless network to download the maps for the entire route before you hit the road. That way, you won't have to use more expensive mobile network bandwidth on the journey. Then, you can flick Marble into guidance mode, sit back in your car and watch as Marble tracks your progress along the route and provides step-by-step directions and relevant maps. While you travel, you will see that Marble displays a progress bar to warn you of the next turn (it counts down from 1,000m). You can check the icon that shows you the direction of the next turn and know that if it is grayed out, the next turn is still more than 1,000m away. Check the step-by-step text instructions for further guidance. You



Figure 2. You easily can use a smartphone running Marble as an in-car satellite navigation device.

Get Marble on Your Nokia N900

Most testing of the mobile version of Marble is done using Maemo on Nokia's N900 smartphone. Packages are available to make installation easy.

To get the mobile version of Marble, simply point your phone's Web browser to the Marble Web site (edu.kde.org/marble), navigate to the Download section and scroll down to find the Maemo packages.

Click on the Maemo package, and when it has downloaded, the Maemo package manager should open. You just need to

make a few more clicks to confirm that you want to add the Marble software catalog, and then Marble will be ready to use. You also will find that the Marble software catalog enables you to install the marble-maps package that gives you access to additional maps, such as satellite imagery, if you want to try something different from the default OpenStreetMap maps.

If you want to be able to calculate routes without an Internet connection, you also should install monav-routing daemon, which also is part of the Marble software catalog in the Maemo package manager.

FEATURE Put the World in Your Pocket with Marble



Figure 3. Marble's customized guidance mode interface shows only the information you need: the next turn and when to expect it.

also get to see the overall distance to the destination and can toggle between guidance and map mode that enables you to pan and zoom the map manually. Remember also to keep your eyes on the road.

You might be tempted to play with Marble's routing just for the fun of it, while racking up big mobile data bills to download all the map data. So, it's fortunate that Marble allows you to download map data for planned journeys at a convenient time, such as when you are within range of your home wireless network. According to developer Dennis Niehüser, "Marble 1.0 allows you to download map data along the route, which keeps the download amount

You easily can explore and plan your journey with Marble on your home computer with a big screen and fast Internet connection and then transfer it to your mobile phone.

significantly lower if you're interested only in the route part." This can cut data downloads by 85–90%. You even can work out routes entirely off-line. Marble can download a set of off-line maps to do vehicle routing all around the world.

More than the Sum of Its Parts

Of course, other navigation solutions exist for mobile devices. Some, such as Navit, Mappero, ModRana and MoNav, even are free software. However, Marble has a few special features. Marble makes use of a range of existing algorithms and Web services, and as Dennis explains, the approach Marble takes is to "query the available ones in parallel and choose the best result". However, there is more to it than just fetching results: "many of the services can only calculate routes between two points, while Marble supports an arbitrary number of via points—therefore, we need to split up requests and combine the results, and all of that is transparent to the user". Marble also adds enhancements to the step-by-step driving instructions: "We generate them ourselves, as that allows us to translate them in all languages supported by

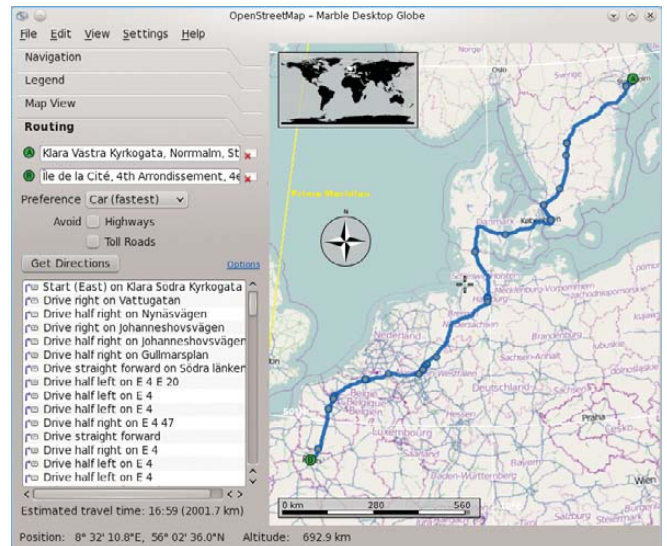


Figure 4. Marble's route finder can help you plan long journeys by car.

KDE software—no other project has that huge a number of supported languages."

Dennis explains that there are other advantages to using a selection of alternative back ends: "Since we query multiple routing back ends in parallel, Marble ranks the results, eliminates duplicates and then shows the remaining routes on the map. The best is active, but you can easily switch to one of the alternative routes."

Marble for smartphones also gains from its close relationship to its desktop cousin. You easily can explore and plan your journey with Marble on your home computer with a big screen and fast Internet connection and then transfer it to your mobile phone. Simply export a file from the computer to the phone and then open it in the mobile version of Marble. In fact, you can import route data from any application supporting the standard KML format. The mobile version of Marble does, however, have plenty of differences from the desktop version. For a start, the mobile version does not depend on the KDE platform, so none of the KDE libraries are needed for installation. This makes the application download only a few megabytes, but it does mean that the KNewStuff interface—with which you may already be familiar if you use KDE software on the desktop—is not available to install add-ons, such as new maps.

Making the software simple to use is one of the developers' main goals. Much of the technology needed has developed independently of Marble, but Marble draws it all together and presents it to the user via a consistent and easy-to-use

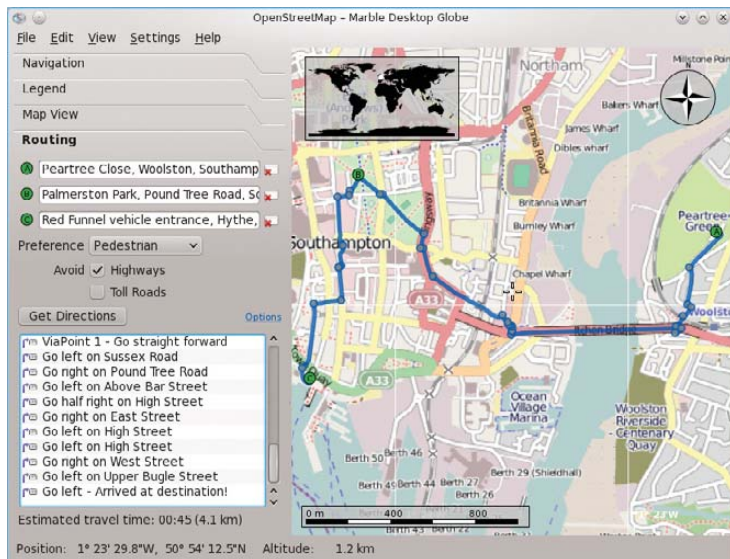


Figure 5. Large-scale OpenStreetMap data makes finding the best walking routes around your home city easy too.

interface. For Dennis, the major challenge has been “integrating the different back ends and coping with their strengths and limitations, ideally without the user knowing”. This is the key: “hiding power behind a simple user interface”.

Behind Marble's Route-Finding

In the best traditions of free software, Marble makes use of a whole load of existing routing solutions rather than re-implementing everything itself.

Addresses and current locations are located on the globe using a combination of the Nominatim Web service (for OpenStreetMap address search), GeoIP Web services to link IP addresses to locations and a local database shipped with Marble. Nominatim also is used to convert positions to addresses, as is Gosmore, which is an optional runtime dependency.

Marble works out routes by taking advantage of the OpenRouteService, YOURS Web service, Routino and the aforementioned Gosmore and Monav as optional runtime dependencies.

Of course, free software also is about contributing things back, and Dennis is excited about Marble's routing instructions implementation becoming useful for other projects. “It is used by the YOURS implementation on OpenStreetMap, which has good chances to be included on the main OpenStreetMap site at some point. That would mean that some part of Marble, and the efforts of the KDE translators, would be used by the OpenStreetMap project itself—a nice way for KDE to contribute.”

Mapping Out the Future

Plenty more is coming in the future. Being able to plan a route

Powerful: Rhino



Rhino M6500/E6510

- Dell Precision M6500 w/ Core i7 Quad (8 core)
- Dell Latitude E6510 w/ 2.53-2.8 GHz Core i5/i7
- Up to 17" WUXGA LCD w/ X@1920x1200
- NVidia Quadro FX 3800M
- 250-750 GB hard drive
- Up to 32 GB RAM (1333 MHz)
- DVD±RW or Blu-ray
- 802.11a/b/g/n
- Starts at \$1385

- High performance NVidia 3-D on a WUXGA RGB/LED
- High performance Core i7 Quad CPUs, 32 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 X201 Tablet

- ThinkPad X201 tablet by Lenovo
- 12.1" WXGA w/ X@1280x800
- 2.0-2.13 GHz Core i7
- Up to 8 GB RAM
- 250-500 GB hard drive / 160 GB SSD
- Pen/stylus input to screen
- Dynamic screen rotation
- Starts at \$1940

Rugged: Tarantula



Tarantula CF-31

- Panasonic Toughbook CF-31
- Fully rugged MIL-SPEC-810G tested: drops, dust, moisture & more
- 13.1" XGA TouchScreen
- 2.4-2.53 GHz Core i5
- Up to 8 GB RAM
- 160-750 GB hard drive / 256 GB SSD
- Call for quote

EmperorLinux

...where Linux & laptops converge

www.EmperorLinux.com

1-888-651-6686



Model specifications and availability may vary.

FEATURE Put the World in Your Pocket with Marble

and following it is useful, but what about being able to take your phone out when you run or cycle and have Marble record your route so you can upload it to your home computer and see where and how far you went? Dennis thinks it can be done: “Technically, we’re just 20 lines of code away from that. It’s mainly missing currently because we didn’t find a nice place in the user interface yet.”

Other future enhancements include integration of off-line address search, greater optimization for different transport types—for example, improving the selection of different routes for cycling and driving or choosing the fastest versus the shortest route. Voice instructions, one of the major features missing from Marble on a smartphone when compared to a dedicated satellite navigation system, also will be added using text-to-speech technology.

Torsten also is excited about the future use of OpenGL, recently demonstrated by Marble developer Bernhard Beschow. Adding OpenGL support will make it possible to move rendering of the maps from pure software to make better use of the graphics processors found in modern smartphones. Torsten believes that “OpenGL is important for the mobile use case in terms of CPU and battery usage—we hope that OpenGL support will start to appear in summer 2011.”

Implementing all these features, of course, will take a lot of hard work, and not just from developers: “We need more people to help us: with promotion, with documentation and, of course, through coding.” As Marble depends heavily on free data sources, such as those integrated in the OpenStreetMap Project, it also benefits from contributions to the stock of free geographical data by governments, individuals and companies. In particular, Torsten would like to see more aerial imagery released under free licenses. Dennis has similar views: “Any data that is commonly tagged in OpenStreetMap will be useful if it can be imported automatically in some way. Satellite maps would be awesome to have as well in more detail.”

Education or Navigation?

Marble became well known as an educational desktop globe along the lines of Google Earth. After all, it was developed as part of the KDE Education Project. However, Marble’s 1.0 release, part of KDE Software Compilation 4.6 released earlier in 2010, has brought both its navigational features and mobile interface to maturity, and the developers now prefer to say that Marble is the free software that lets you “explore the world and find your way”.

Even when work on Marble first started, Torsten’s goals were ambitious: to produce a map widget that would “become for geo-browsers what KHTML/WebKit is for Web browsers”. The aim always was very much more than just the production of an educational globe. Torsten explains: “We took the education focus as a starting point, and since then, we have explored more and more use cases and have widened our scope.”

Marble also has found a place in many other applications, from scientific mapping and weather-tracking applications to photo management, sensor data visualization and even Linux distribution installers. It goes beyond Earth too, with map data for the moon and solar system planets available for download.

For Dennis, the development of Marble into a mobile navigation system has fulfilled a long-felt need: “My wish to use a free navigation assistance goes back quite some years. I started

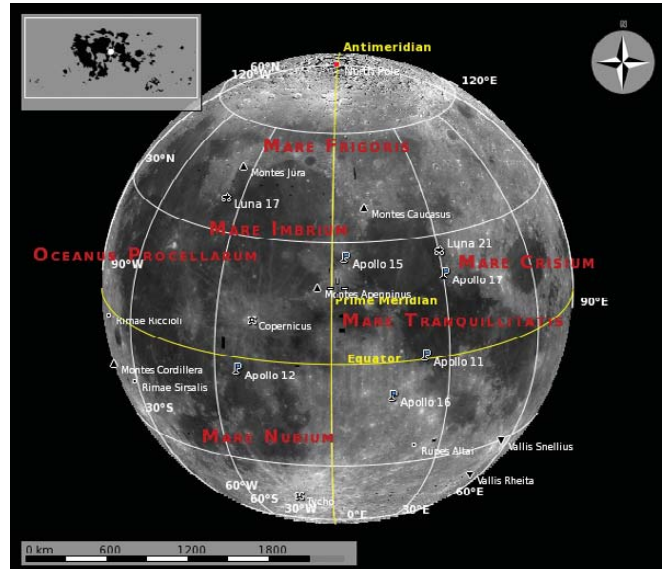


Figure 6. If you get bored with planet Earth, there are plenty of other worlds to explore.

to look into open-source navigation solutions, but the software available at that time wasn’t really capable of what I wanted.” Nonetheless, he began contributing to Marble. “Once I had an N900 and heard that there was a Marble port for the N900 already, the idea pretty much was there.”

So, what is Marble now, an educational desktop globe or a personal navigational tool? For Dennis, the answer is easy: “To me, they are not contradictory—it is both, and even more.” ■

Stuart Jarvis is a scientist and member of KDE’s Marketing Working Group. Lacking a Marble-capable smartphone, he spends a lot of time getting lost in unfamiliar places.

Resources

Marble: edu.kde.org/marble

Navit: www.navit-project.org

Mappero: maemo.org/downloads/product/Maemo5/maemo-mapper

ModRana: maemo.org/packages/view/modrana

MoNav: wiki.openstreetmap.org/wiki/MoNav

OpenStreetMap: nominatim.openstreetmap.org

Gosmore: wiki.openstreetmap.org/wiki/Gosmore

OpenRouteService: openrouteservice.org

YOURS: wiki.openstreetmap.org/wiki/YOURS

Routino: www.routino.org



ACCELERATE YOUR CLOUD STRATEGY

March 7-10, 2011 | Santa Clara Convention Center

Don't miss Cloud Connect, the leading event bringing together cloud customers and providers to drive growth and innovation.

Learn about the latest cloud technologies and platforms—including private clouds, industry standards, data storage and CloudSec—that will impact your organization.

CONFERENCE – Explore the latest cloud technologies and platforms and identify opportunities in the cloud. Topics include:

- » Private Clouds
- » Cloud Economics
- » CloudSec
- » Cloud Industry Summit
- » Design Patterns
- » Culture, Politics and Governance
- » Data and Storage
- » DevOps and Automation
- » The Future of Utility Computing
- » Performance and Monitoring



Save 25% on Conference Passes* or Get a Free Expo Pass with Priority Code: CNXCCC07

SPONSORS:

Diamond				
				
Platinum		Gold	Silver	
				
				

* 25% off discount applies to Flex, Conference and Cloud Industry Summit Passes. Discount calculated based on the on-site price and not combinable with other offers. Proof of current industry involvement required. Offer good on new registrations only.

Prices after discount applied: Flex: \$1,721.25 | Conference: \$1,496.25 | Cloud Industry Summit: \$971.25

Augmented Reality with

HTML5

*How far can HTML5 go when
writing mobile applications?*

RICK ROGERS

In a previous *Linux Journal* article (“Developing Portable Mobile Web Applications”, September 2010, www.linuxjournal.com/article/10789), I looked at HTML5 and how it could be used to write applications for mobile phones. The techniques presented in that article work well for applications that use text, buttons, images, audio and even video, but what about cutting-edge applications that stretch the envelope of what mobile phones can do? In an effort to find out, I decided to implement a rather simple mobile augmented reality application, doing as much as I could in HTML5. This article explores the techniques for extending JavaScript capabilities to write applications that do more than is possible with standard HTML5.

Augmented Reality

Augmented reality (AR) is the name given to a class of applications that combines the unique capabilities of mobile phones to extend users' perceptions of their environments. Layar (www.layar.com) was one of the first AR applications, and it's still one of the more creative. Augmented reality overlays the current camera preview screen with additional information—you can see examples in this YouTube video: (www.youtube.com/watch?v=A6Le50-QN3o&feature=player_embedded). Figure 1 shows what Layar looks like when the "Starbucks" layer is loaded and the camera is pointed at a mall where there is a Starbucks coffee shop.

This application makes use of a number of mobile phone features:

- Camera preview.
- Compass (direction the camera is pointed).
- Location.
- 2-D graphics (for the overlay).
- Database capabilities.

Layar is a very advanced application, with many options to make it easy to use. Again, the essential nature of AR is that the user sees additional information superimposed on a camera preview.

HTML5 Extensions

How would you implement this kind of application using HTML5? For the sake of creating an example application, let's reduce AR to a simple case: show the current camera preview on the user's screen and superimpose the current compass direction on top of



Figure 1. Layar with the "Starbucks" Layer

public.wholesaleappcommunity.com), are defining APIs for camera preview, but there are no WAC mobile phones yet.

3. In order to add your own HTML5 extensions to a mobile platform, you have to do some platform-specific code. That means you have to give up some portability, but let's accept the trade-off and focus on one platform, Android. Let's create the needed Dalvik/Java code to implement this simple AR application and take a look at how JavaScript can call Dalvik methods and vice versa.

The ARCompass Application

The application will be a hybrid Dalvik/HTML5 application. The HTML5 part will run in a browser. Android applications create an Internet browser view in one of two ways:

1. Issue an Intent with the URL to open, and Android will resolve that Intent by opening the browser application and passing it

Augmented reality (AR) is the name given to a class of applications that combines the unique capabilities of mobile phones to extend users' perceptions of their environments.

the preview. Let's also animate the compass card so it moves as the phone's camera pans around. In principle, the overlay could be anything, but a compass card is a start.

HTML5 has greatly extended the capabilities of HTML applications, but some things still are missing for this application:

1. HTML5 doesn't include a compass API. You need a way to access the mobile phone's current compass direction and receive periodic updates as the direction changes. You could use the API in one of the Web app toolkits (such as PhoneGap or Titanium) for this, but let's create our own interface and demonstrate how you can access just about any Object from JavaScript.
2. You need a live camera preview on the screen, and there isn't a camera API in HTML5. Extensions to HTML5, such as WAC (Wholesale Applications Community,

the URL. When you exit the browser, control is returned to the calling application. This approach works fine for regular HTML5 applications, but it doesn't provide a way to add new interfaces to JavaScript.

2. Inflate a WebView and pass it the URL. There is a lot more flexibility in the WebView compared to the browser application, including a public method, `addJavascriptInterface` (Object obj, String InterfaceName). This method lets you create your own JavaScript APIs for the scripts run by a WebView. Note that there is a bit of a security hole here—anything you make visible to JavaScript can be accessed by *any* JavaScript script run by this WebView, whether or not you wrote the script. You want to be sure the user can't navigate to random Web sites that might misuse your interface. In this case, let's include the HTML and JavaScript files in the application and not provide the user any chance to navigate away.

Listing 1. arcompass.html

```
<!DOCTYPE HTML PUBLIC>
<head>
  <title>AR Compass</title>
  <script type="text/javascript"
    src="arcompass.js">
  </script>
</head>
<body>
  <div id="extra">
    <button type="button"
      onclick="window.direction.turnOnCompass()">
      Start</button>
  </div>
  <div id="overlay" style="position: absolute;
    left:280; top:60; z-index:500;
    background-color:#0000" >
    <canvas id="e" width="200" height="200">
    </canvas>
    <script>
      drawCompass();
    </script>
  </div>
</body>
</html>
```

Listing 2. arcompass.js

```
var currDir;
var canvas;
var context;
var card;
function drawCompass() {
  currDir = 0;
  canvas = document.getElementById("e");
  context = canvas.getContext("2d");
  card = new Image();
  card.src = "CompassCard.png";
  card.onload = function() {
    context.translate(100,100);
    context.globalAlpha = 0.5;
    context.drawImage(card, -100, -100, 200, 200);
  }
}

function updateView(dir) {
  context.rotate(currDir*2*Math.PI/360);
  context.rotate(-dir*2*Math.PI/360);
  context.drawImage(card, -100,-100,200,200);
  currDir = dir;
}
```

Let's write a Dalvik application that shows the camera preview screen and overlays that with a WebView that will draw and animate the compass card. Of course, you'll also need the compass information passed from Android back to the HTML5 code, so it can animate the card properly.

Assuming you've already loaded the Android SDK (from developer.android.com), you can follow along by downloading the ARCompass.prj project file and the HTML and JavaScript files from ftp.linuxjournal.com/pub/lj/listings/issue203/10920.tgz.

The HTML5 Part

Before diving into the Dalvik part of the application, let's take a look at the HTML5 part, which draws a compass card and rotates the card to show the current direction the phone is pointed. The .html, .js and .png files used here are stored in the Dalvik application's assets folder, which is created automatically when Eclipse creates an Android project.

The header of the HTML file declares a title and references the JavaScript file. The body consists of two <div>s: one with a button and one with a <canvas>. You don't really need the button for the application, but I wanted to show how you call Dalvik routines from JavaScript/HTML. Notice that the onclick attribute for the button is set to `window.direction.turnOnCompass()`. You'll see later how that API is declared in Dalvik and how it is wired to start the compass sensor sending direction updates.

The second <div> is the canvas where you draw the compass card. Let's assume a landscape orientation for the application and position the canvas on the right side of the screen. In a real application, you'd take account of the specific screen geometry of the device you're running on. For simplicity here, I've hard-coded some pixel values. A short embedded script then asks the `drawCompass()` function to draw the initial compass card image.

The JavaScript file declares some variables and defines two functions:

1. `drawCompass()` draws the initial compass card, with north pointing up.
2. `updateView(dir)` will be called whenever you get an updated compass direction from the compass sensor (I explain how later). It rotates the drawing context appropriately and redraws the compass card.

The Android Part

Let's turn our attention to the Dalvik part of the application. You need manifest and layout files (Listings 3 and 4).

The manifest says the application consists of only one screen (the ARCompass activity) and that it needs the user's permission to access the camera and Internet. It also asked for `SET_DEBUG_AP` permission, which allows you to run the app on a real device while using the Eclipse debugger.

The layout file says the activity contains two views, a WebView cleverly named `webView0` and a SurfaceView named `preview`. I'm using a Relative Layout so you can position the views on top of each other using the `layout_align_top` and `layout_align_bottom` attributes for `webView0`. I'll handle any other needed layout in the HTML that I'll ask WebView to render.

Listing 3. AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android=
"http://schemas.android.com/apk/res/android"
package="com.lj.ARCompass"
android:versionCode="1"
android:versionName="1.0">
<application android:icon="@drawable/icon"
android:label="@string/app_name"
android:debuggable="true">
<activity android:name=".ARCompass"
android:label="@string/app_name">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
</application>
<uses-permission
android:name="android.permission.CAMERA">
</uses-permission>
<uses-permission
android:name="android.permission.INTERNET">
</uses-permission>
<uses-permission
android:name="android.permission.SET_DEBUG_APP">
</uses-permission>
</manifest>
```

The Dalvik part of the application is more complicated, but not so bad if you break it down into sections:

```
package com.lj.ARCompass;

import java.io.IOException;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.hardware.Camera;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.os.Bundle;
import android.util.Log;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.view.Window;
import android.webkit.WebChromeClient;
import android.webkit.WebSettings;
import android.webkit.WebView;
import android.widget.Toast;
```

Listing 4. main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android=
"http://schemas.android.com/apk/res/android"
android:orientation="horizontal"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
>
<SurfaceView
android:id="@+id/preview"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:layout_weight="1"
/>
<WebView
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:id="@+id/webView0"
android:layout_alignTop="@id/preview"
android:layout_alignBottom="@id/preview"
/>
</RelativeLayout>
```

```
public class ARCompass extends Activity
implements SurfaceHolder.Callback {

private WebView mWebView;
private SensorManager mSensorManager;
private float[] mValues;
private boolean compassOn = false;

private static final String TAG = "ARCompass";
final Context mContext = this;

private Camera mCamera;
private SurfaceView mSurfaceView;
private SurfaceHolder mSurfaceHolder;
private boolean mPreviewRunning;
```

These first lines import all the libraries you need, declare some needed variables, and declare the only Activity, ARCompass. Note that I've said ARCompass will implement the SurfaceHolder.Callback interface—this is needed for the camera preview.

The next block of code declares a SensorEventListener:

```
private final SensorEventListener mListener =
new SensorEventListener() {
@Override
public void onAccuracyChanged
(Sensor sensor, int accuracy) {
}
@Override
public void onSensorChanged(SensorEvent event) {
mValues = event.values;
Log.d(TAG,"Compass update: " + mValues[0]);
String url =
```

When you make a call this way from JavaScript, it can be important to know that this code is going to run in a thread separate from the one where it was invoked above.

```

    "javascript:updateView(" + mValues[0] + ");";
    mWebView.loadUrl(url);
}
};

```

Later on, I'm going to wire this listener up to the update events that I'll get from the compass sensor. For now, notice in the `onSensorChanged()` method that the ultimate result is to load a URL into the `WebView` (also created later). The URL is of the form `javascript:updateView(direction)`, because the first value passed to you in the array `event.values[]` is, in fact, the current compass direction. Loading the URL into the `WebView` has the effect of calling the `updateView()` function just defined in `arcompass.js`.

The next section of code gets into the `onCreate()` method, called when the activity is first created:

```

/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    Log.d(TAG, "onCreate");
    // Get rid of title
    requestWindowFeature(Window.FEATURE_NO_TITLE);

    setContentView(R.layout.main);

    // Initialize the surface for camera preview
    mSurfaceView =
        (SurfaceView)findViewById(R.id.preview);
    mSurfaceHolder = mSurfaceView.getHolder();
    mSurfaceHolder.addCallback(this);
    mSurfaceHolder.setType
        (SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
    Log.d(TAG, "SurfaceView initialized");

    // Initialize the WebView
    mWebView = (WebView) findViewById(R.id.webView0);
    WebSettings webSettings = mWebView.getSettings();
    webSettings.setSavePassword(false);
    webSettings.setSaveFormData(false);
    webSettings.setJavaScriptEnabled(true);
    webSettings.setSupportZoom(false);
    mWebView.setBackgroundColor(0);

    mWebView.addJavascriptInterface
        (new CompassJavaScriptInterface(), "direction");
    Log.d(TAG, "JavaScript interface added");

    /* Set WebChromeClient before calling loadUrl! */
    mWebView.setWebChromeClient

```

```

    (new WebChromeClient() {
        @Override
        public boolean onJsAlert(
            WebView view, String url, String message,
            final android.webkit.JsResult result){
            new AlertDialog.Builder(mContext)
                .setTitle("JavaScript dialog")
                .setMessage(message)
                .setPositiveButton(android.R.string.ok,
                    new AlertDialog.OnClickListener() {
                        public void onClick(
                            DialogInterface dialog, int which) {
                            result.confirm();
                        }
                    })
                .setCancelable(false)
                .create()
                .show();
            return true;
        }
    });

    mWebView.loadUrl(
        "file:///android_asset/arcompass.html");
}

```

After calling the superclass routine and setting a TAG to be used with log messages, I request the `FEATURE_NO_TITLE` for the window, because I don't want or need the usual Android title bar. Then, I connect with the `main.xml` layout file I looked at earlier.

The next block of code initializes the `SurfaceView` that you're going to use for the camera preview, and the next block of code initializes the `WebView`. I'll leave most of the details to the reader (the Android SDK help files are excellent), but note one line in particular:

```
webSettings.setJavaScriptEnabled(true);
```

By default, `WebViews` don't execute JavaScript. This setting turns on that ability.

The line after the `WebView` settings invokes `addJavascriptInterface()` to add a new API that can be called from scripts run by the `WebView`. I define the `CompassJavaScriptInterface` class later, including the method `turnOnCompass()`, but this is where I defined the "direction" part of the function call I made back in `arcompass.html` (`window.direction.turnOnCompass()`).

The next 20 lines or so define a `WebChromeClient`, so you can issue `alert()` function calls from JavaScript, and those will be converted into Android alert boxes. This is useful for debugging, but not absolutely needed unless your JavaScript uses alerts.

The last line in this section loads the arcompass.html file into the WebView. Note the syntax of the file reference. Again, the file is in the assets folder of the application project, and the SDK includes that folder in the .apk package that is downloaded when installing the application. The next section of code connects the compass sensor to the application:

```
final class CompassJavaScriptInterface {
    /* Note this runs in a separate thread */

    CompassJavaScriptInterface() {
    }

    public void turnOnCompass() {
        Log.d(TAG, "turnOnCompass");
        mSensorManager = (SensorManager)
            getSystemService(Context.SENSOR_SERVICE);
        Sensor mSensor =
            mSensorManager.getDefaultSensor
                (Sensor.TYPE_ORIENTATION);

        if(mSensor != null){
            mSensorManager.registerListener(mListener,
                mSensor, SensorManager.SENSOR_DELAY_NORMAL);
            compassOn = true;
            Log.d(TAG, "Compass started");
        }
        else{
            Toast.makeText(mContext,
                "No ORIENTATION Sensor",
                Toast.LENGTH_LONG).show();
            compassOn = false;
            finish();
        }
    }
}

@Override
protected void onDestroy() {
    super.onDestroy();
    if(compassOn){
        mSensorManager.unregisterListener(mListener);
    }
    finish();
}
```

First, I declare the class that I referred to back in addJavaScriptInterface(). When you make a call this way from JavaScript, it can be important to know that this code is going to run in a thread separate from the one where it was invoked above. In particular, if the called routine needs to manipulate the user interface, it will not be running in the UI thread, so it needs to post a runnable for that thread to pick up. In this case, I'm just working with the Sensor interface, so running in a separate thread is not an issue.

The only method I define is turnOnCompass(), but I could define others. If I defined another method blatz(), I could call it from JavaScript as window.direction.blatz(). The turnOnCompass() method invokes the SensorManager and asks for a handle to the default orientation sensor. If there is a default orientation sensor,

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	URL	Page #
1&1 INTERNET, INC.	www.oneandone.com	1
ABERDEEN, LLC	www.aberdeeninc.com	3
CLOUD CONNECT	www.cloudconnectevent.com	59
DRUPALCON	chicago2011.drupal.org	13
EMAC, INC.	www.emacinc.com	53
EMPERORLINUX	www.emperorlinux.com	57
GENSTOR SYSTEMS, INC.	www.genstor.com	27
IPHONE/IPAD DEVCON	www.iphonedevcon.com	7
IXSYSTEMS, INC.	www.ixsystems.com	C3
LINODE, LLC	www.linode.com	71
LINUX JOURNAL STORE	www.linuxjournalstore.com	15
LOGIC SUPPLY, INC.	www.logicsupply.com	31
LULLABOT	www.lullabot.com	17
MICROWAY, INC.	www.microway.com	C2, C4
MIKRO TIK	www.routerboard.com	5
O'REILLY MYSQL CONFERENCE	conferences.oreilly.com/mysql	67
O'REILLY WHERE 2.0 CONFERENCE	conferences.oreilly.com/where	67
POSSCON	www.posscon.org	77
POLYWELL COMPUTERS, INC.	www.polywell.com	79
PYCON	us.pycon.org	39
SCALE	www.socallinuxexpo.org	47
SILICON MECHANICS	www.siliconmechanics.com	29, 41
TECHNOLOGIC SYSTEMS	www.embeddedx86.com	45

ATTENTION ADVERTISERS

June 2011 Issue #206 Deadlines

Space Close: March 21; Material Close: March 29

Theme: Embedded

BONUS DISTRIBUTIONS:

O'Reilly's Web 2.0 Expo, Linux and Open Source on Wall Street, USENIX HotOS XIII, Penguicon, php|tek

Contact Joseph Krack, +1-713-344-1956 ext. 118, joseph@linuxjournal.com

FEATURE Augmented Reality with HTML5

it registers the `SensorEventListener` I defined at the beginning, sets a housekeeping boolean and returns. If there isn't an orientation sensor, it tells the user with a `Toast`, and exits.

The final block of code in this section makes sure that you de-register the listener when the application exits. If you happen to be the only registered listener for orientation, this would give Android the opportunity to power down that service, and even that sensor.

The last section of Dalvik code deals with the camera preview:

```
// Create camera preview.
public void surfaceCreated(SurfaceHolder holder){
    mCamera = Camera.open();
    try {
        mCamera.setPreviewDisplay(holder);
    } catch (IOException exception) {
        mCamera.release();
        mCamera = null;
    }
}

// Change preview's properties
public void surfaceChanged(SurfaceHolder holder,
    int format, int w, int h){
    mCamera.startPreview();
    mPreviewRunning = true;
}

// Stop the preview.
public void surfaceDestroyed
    (SurfaceHolder holder){
    mCamera.stopPreview();
    mPreviewRunning = false;
    mCamera.release();
}
}
```

Again, the details of the camera preview implementation are best left to the Android SDK documentation. The three methods—`surfaceCreated()`, `surfaceChanged()` and `surfaceDestroyed()`—are the methods of the `Surface.Callback` interface that I said this activity would implement, and they are called for you at each of those events. When the surface is created, you connect the camera preview to the `SurfaceHolder`. When the surface is destroyed, you stop the camera preview and release the camera. The `surfaceChanged` method is called only once, after `surfaceCreated`, and you actually start the preview there.

When you build and run this program on a mobile phone, you get something like the picture shown in Figure 2. This is an HTC EVO screenshot. I haven't done a lot to account for screen geometry differences, so your phone may look a bit different.

The user sees a `Start` button and a compass card superimposed on the current camera preview. The `Start` button isn't really needed, but I included it so I could show how JavaScript/HTML can call a Dalvik method.

When you tap the `Start` button, the HTML part of the application calls `window.direction.turnOnCompass()`, which is implemented in Dalvik. The method asks the orientation sensor to start sending compass readings to `mListener`. Every

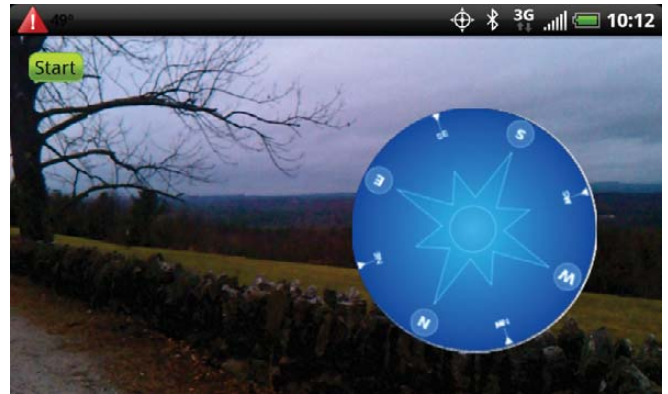


Figure 2. ARCompass Running on an HTC EVO

time `mListener` gets a new compass reading, it calls the JavaScript routine `updateView()` to repaint the compass card on the screen.

So What Does All This Mean

I've shown how to write hybrid applications with HTML5 and Dalvik. It's relatively easy to set things up so JavaScript can call Dalvik methods, and Dalvik can call JavaScript methods. I've shown that you can create rather advanced applications that composite the Dalvik and HTML5 user interfaces so they look like one to the user.

But you could just as easily have written the whole application in Dalvik, so what is the advantage of writing part in HTML5? Here are the advantages:

1. If you were writing a real application, the HTML5 part would be (relatively) portable to other platforms. You wouldn't have to rewrite it to port to, say, the iPhone. In the example, the HTML5 part is pretty small, but in principle, it could be much larger.
2. You could have kept the HTML5 part of the application on a remote HTTP server, to be updated whenever the app is run, without requiring the user to download an update.
3. If your application displayed information from the Web, it could be argued that HTML5 is a more natural place for Web interaction than Dalvik.

Hybrid applications, such as the example here, can be a valid way to create mobile applications that combine the power of HTML5 and the native platform. As long as the platform gives you a way to interact between JavaScript and the native application environment, there really doesn't seem to be any barrier to the kinds of applications you can write. ■

Rick Rogers has been a professional embedded developer for more than 30 years. Now specializing in mobile application software, when Rick isn't writing software for a living, he's writing books and magazine articles like this one. He welcomes feedback on the article at portmobileapps@gmail.com.

Compass card graphics adapted from commons.wikimedia.org/wiki/File:Compass.svg.

O'REILLY

MySQL

Conference & Expo

APRIL 11-14, 2011
Santa Clara, CA



the ecosystem and beyond

Register Now and Save 15% when you use discount code: **mys11jr**
conferences.oreilly.com/mysql

©2011 O'Reilly Media, Inc. O'Reilly logo is a registered trademark of O'Reilly Media, Inc. All other trademarks are the property of their respective owners. 11024

O'REILLY

where2.0[™] CONFERENCE

APRIL 19-21, 2011
SANTA CLARA, CA

The Business of Location

The O'Reilly Where 2.0 Conference explores the intersection of location technologies and trends in software development, business strategies, and marketing. The source for all things location-aware, Where 2.0 brings together CTOs, marketers, developers, technologists, researchers, geographers, startups, business developers, and entrepreneurs, to shed light on the issues surrounding:

- Location and Mobile
- Location and Data
- Business Strategies
- Marketing

conferences.oreilly.com/where



Register now and save 15%

Use discount code **whr11jr** at conferences.oreilly.com/where

©2011 O'Reilly Media, Inc. O'Reilly logo is a registered trademark of O'Reilly Media, Inc. All other trademarks are the property of their respective owners. 11025

Run with MeeGo

What's new in 1.1?

Ibrahim Haddad



The MeeGo Project has had multiple releases and has progressed significantly since its announcement in February 2010. This article provides an overview of the MeeGo Project for newcomers, a review of the benefits MeeGo provides to the players in the mobile ecosystem, and discusses the features in the latest MeeGo 1.1 release, announced October 28, 2010.

Introduction to MeeGo

MeeGo is an open, collaborative project between the project founders (Nokia and Intel), the Open Source community and various commercial and noncommercial partners with the goals of accelerating the adoption of Linux on a magnitude of client devices and enriching the technical Linux platform as the platform of choice for mobile computing devices.

MeeGo is a Linux-based operating system built for the next generation of computing devices across multiple hardware architectures. Different from other mobile operating systems, MeeGo is an open-source platform governed by best practices of open-source development. It includes the following:

1. Core operating system.
2. User Interface (UI) libraries and tools.
3. References user experiences for multiple devices.
4. Standard set of application programming interfaces (APIs) across all target device types.
5. A software development kit (SDK) that enables application developers to develop, install, debug and run applications either on reference devices or in an emulated environment.

MeeGo supports a magnitude of mobile client devices (handsets, connected TVs, in-vehicle infotainment, Netbooks and tablets). It provides choice and flexibility to create and deliver a uniquely differentiated service offering. It's an unusual project in that it is aligned closely with upstream projects, as MeeGo requires that submitted patches also are submitted to the appropriate upstream project and are on a path for acceptance. This development model has the great effect of improving all upstream open-source projects used in MeeGo, and it guarantees

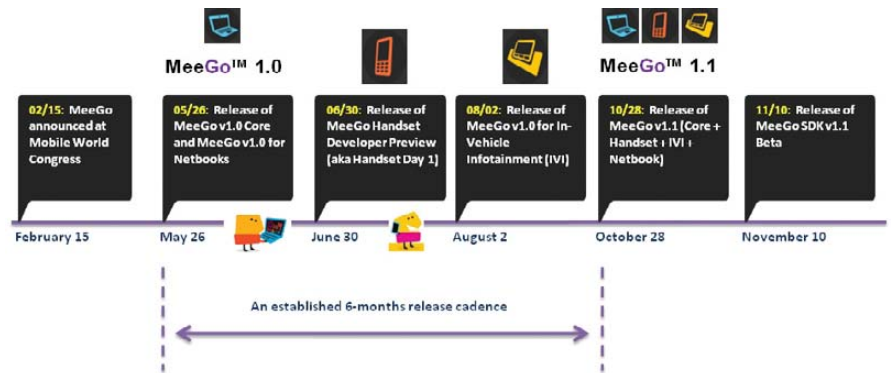


Figure 1. MeeGo Releases Since the Project's Announcement

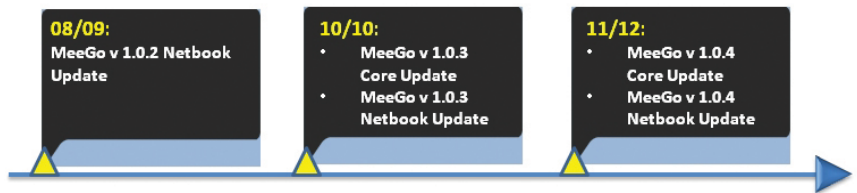


Figure 2. MeeGo Release Updates

a unified technical approach led by the upstream projects.

How Does MeeGo Benefit the Mobile Ecosystem?

If you are an open-source developer, you will enjoy working on an open-source mobile platform project that follows open-source development practices. You will have full access to everything MeeGo, and you can rest assured that any code contributed by MeeGo will be submitted to the appropriate upstream open-source projects. From this perspective, every other Linux mobile and desktop effort will benefit from MeeGo's work and contributions.

If you are an application developer, you will enjoy working with a single set of APIs across a number of client devices (handsets, tablets, Netbooks, in-vehicle infotainment systems and connected TVs). You have access to polished and easy-to-get-and-use developer tools and infrastructure. In addition, there are open forums where you can engage in discussions

directly with the platform and tool creators, exchange ideas and best practices and even participate in the evolution of the platform. Plus, you will enjoy the flexibility of hosting your applications in more than one application store.

If you are a device manufacturer or a wireless operator looking to build and/or deploy devices with MeeGo, the project offers tremendous opportunities. MeeGo is a democratic project with open access to all, at all times. It is the only platform of its kind built with unparalleled openness in the industry. It will accelerate your time to market, lower the complexities involved in targeting multiple device types, allow you to optimize the software stack and, most important, grant you an equal right to participate in the evolution of the software platform.

MeeGo 2010 Milestones

Since the project announcement in February 2010, MeeGo has delivered the core software platform in addition to three user experience implementations (Netbook, handset and in-vehicle infotainment), with several updates in between. Figure 1 provides the roadmap of releases since the project's inception and Figure 2 offers the roadmap of the release updates and what they included.

Between major releases, MeeGo offers updates that usually include general operating system fixes to enhance the stability,

MeeGo and Connected Devices

We all use mobile devices every day (such as Netbooks, connected TVs, tablets, in-vehicle infotainment and handsets). The power of these devices has reached astounding levels with unheard of performance and capabilities. The goal of the MeeGo Project is to develop the best software platform to go with these devices.

Table 1. MeeGo v1.1 Core Software Platform Key Feature List

KEY FEATURE	DESCRIPTION	RELATED UPSTREAM PROJECT
Complete MeeGo-compliant packages	Ensures compatibility.	N/A
GCC 4.5.0 toolchain	Includes support for the Intel Atom microarchitecture and runtime library functions optimized for the Intel SSSE3 instruction set.	gcc.gnu.org/gcc-4.5
Linux kernel 2.6.35	Includes support for the Intel Atom processor Z6xx series family.	kernel.org
X.org Server 1.9.0 and Mesa 7.9	Improves 2-D and 3-D graphics.	www.x.org/wiki and www.mesa3d.org
Qt 4.7 and Qt-mobility 1.0.2	Provides a rich set of APIs for creating compelling applications that include location, sensors, contacts and messaging.	qt.nokia.com
QtWebKit 2.1	Qt port of WebKit.	developer.qt.nokia.com/wiki/QtWebKit
BTRFS	Next-generation filesystem aimed at implementing advanced features while focusing on fault tolerance, repair and easy administration.	https://btrfs.wiki.kernel.org
ConnMan connection manager	Provides support for static IPv6, dhcp-lib and VPN.	connman.net
New oFono telephony stack	Provides support for the telephony functionality.	ofono.org
PulseAudio	Provides support for the audio functionality.	www.pulseaudio.org
GStreamer 0.10.30 with liborc support	General performance improvements.	www.gstreamer.net
Zypper/libzypp package management.	Provides full package management functionalities, such as repository access, dependency solving, package installation and so on.	en.opensuse.org/Portal:Zypper
Udisks and upower	Replaces the deprecated devicekit-disks and devicekit-power.	freedesktop.org/wiki/Software/udisks and upower.freedesktop.org
Buteo synchronization framework and Personal Information Management	Based on Tracker.	projects.gnome.org/tracker
DeviceKit and udev	Used for interacting with hardware devices.	fedoraproject/wiki/Features/DeviceKit and git.kernel.org/?p=linux/hotplug/udev.git
Sensor Framework	Allows developers to take advantage of platform sensors, such as accelerometers, compasses and gyroscopes.	Part of Qt
Universal Plug and Play (gUPnP)	Support for gUPnP providing an easy-to-use, efficient and flexible framework for creating devices and control points.	gupnp.org

compatibility, security and visual quality of the devices running MeeGo. Between MeeGo 1.0 (05/2010) and MeeGo 1.1 (10/2010), the MeeGo Project provided three update releases that featured improvements to the MeeGo core stack and the Netbook release.

The releases follow the six-month cycle promised by the project and are being delivered on time. The MeeGo source code repository is open for

people to pull the source code anytime they like, if they don't want to be restricted to the six-month release cycle. The release updates are available as necessary, depending on the security/stability/compatibility updates. However, you don't need to wait for the official update to become available, because you have access to the code repository, and you can create an updated image from scratch for your target device.

What's New in MeeGo 1.1?

Any device that will run MeeGo needs two things: the MeeGo core software stack and the MeeGo User eXperience (UX) for that specific device, although you are not limited to using the MeeGo UX, and you can create and deploy your own branded UX. Currently, MeeGo is available for Netbooks, IIVI and handsets.

MeeGo v1.1 Core Software Platform

The MeeGo Core 1.1 release provides a common base operating system for the user experiences of all supported device categories. It provides a complete set of enabling technologies for mobile computing. The MeeGo stack contains Linux kernel 2.6.35, X.org server 1.9.0, Web Runtime, Qt 4.7 and Qt Mobility 1.0.2, supporting the contacts, location, messaging, multimedia, and sensor and service frameworks. It also includes a number of leading-edge components, such as the oFono telephony stack, the ConnMan connection manager, the Tracker data indexer, the Telepathy real-time communications framework, the Buteo sync framework and many more.

These technologies are brought to application developers through the MeeGo API, which is based on Qt and other technologies, such as the MeeGo Touch Framework. With the latest Qt version 4.7, the MeeGo developer experience is now enhanced with the introduction of QML, the easy-to-use scripting technology for animated touch-enabled GUI apps.

MeeGo v1.1 Netbook User Experience

The MeeGo v1.1 Netbook UX provides a complete set of core applications and offers a visually rich Netbook user experience that is optimized for power and performance, all built on the latest open-source technologies. Some of the key features include:

- Integrated touch support.
- Easy-to-use applications.
- Instant access to the core applications from the MeeGo home screen (aka Myzone).
- Aggregation of social-networking content, which allows you to view your social-networking activities on one screen as they occur, easily interact

with friends and update your status and site information.

- For a fast and rich Internet experience,

MeeGo on Netbooks

Experiencing MeeGo on Netbooks is very accessible given their popularity and availability. All you need is a Netbook with an Intel Atom or Intel Core 2 CPU, a USB drive (stick) for saving the MeeGo image and booting with it, and the MeeGo image. Step-by-step instructions are available from meego.com/devices/netbook. Instructions are available for Linux, Windows and Mac OS X users, so you have no excuse not to take it for a test-drive.

the MeeGo Netbook UX integrates Google Chrome or, if you prefer, a fully open-source browser solution. Google Chromium also is provided.

MeeGo v1.1 IVI UX

In-Vehicle Infotainment (IVI) systems are devices that deliver navigation, entertainment and networked computing services

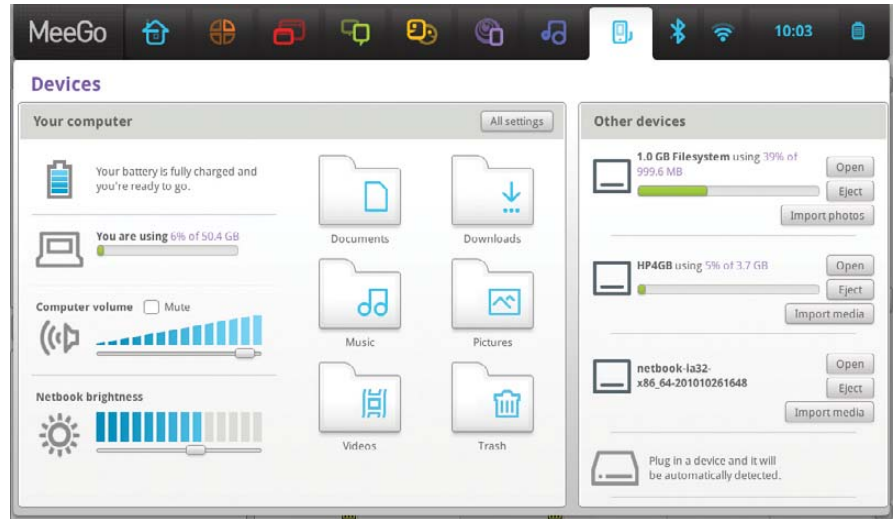
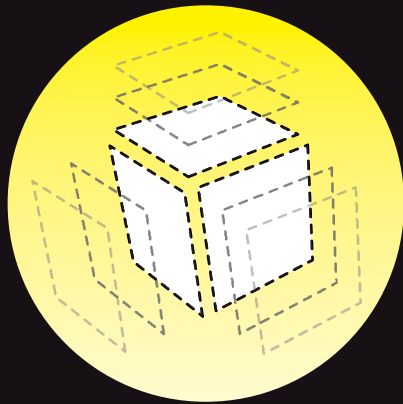


Figure 3. The Devices screen with three connected USB devices, including an Olympus camera showing as a 1GB filesystem, an HP USB device used to copy the screenshots and the USB device that holds the MeeGo Netbook image used to install MeeGo 1.1.



Develop.



Deploy.



Scale.

Full root access on your own virtual server for as little as \$19.95/mo

Multiple Linux distributions to choose from • Web-based deployment • Five geographically diverse data centers • Dedicated IP address • Premium bandwidth providers • 4 core SMP Xen instances • Out of band console access • Private back-end network for clustering • IP fail-over support for high availability • Easily upgrade or add additional Linodes • Free managed DNS

For more information visit www.linode.com or call us at 609-593-7103



linode.com

FEATURE Run with MeeGo

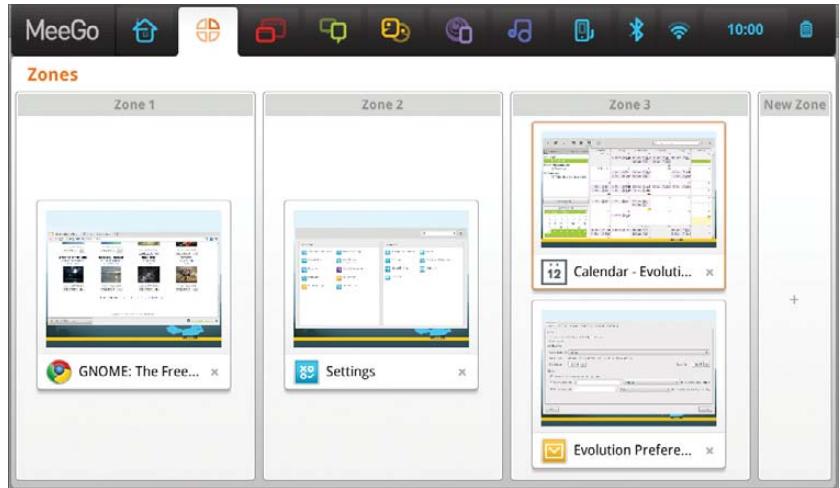


Figure 4. The Zones screen showing three different active zones or work areas.

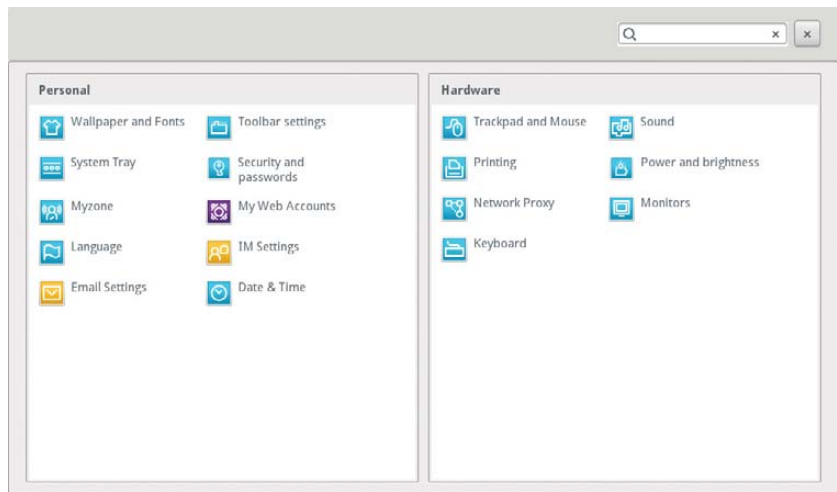
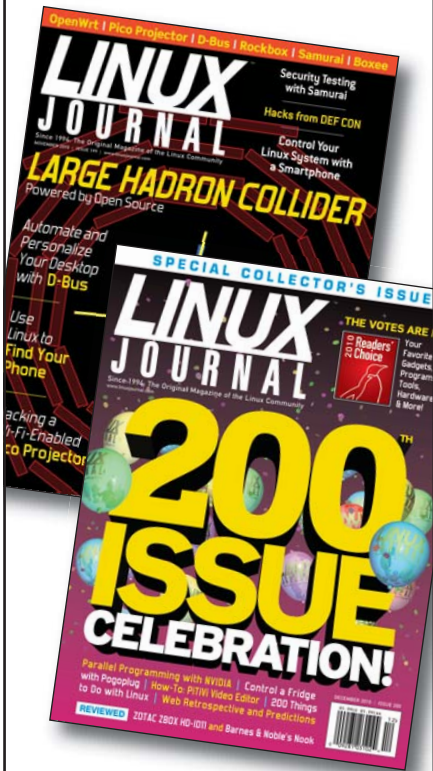


Figure 5. The Settings screen where users can configure various aspects of their MeeGo Netbook setup.



Figure 6. A user-modified home screen of the WeTab with some direct shortcuts to the author's most-used applications. The WeTab used the MeeGo Netbook stack as a base for development purposes.



LINUX JOURNAL

Since 1994: The Original Magazine of the Linux Community

SUBSCRIBE TODAY!

WWW.LINUXJOURNAL.COM/SUBSCRIBE

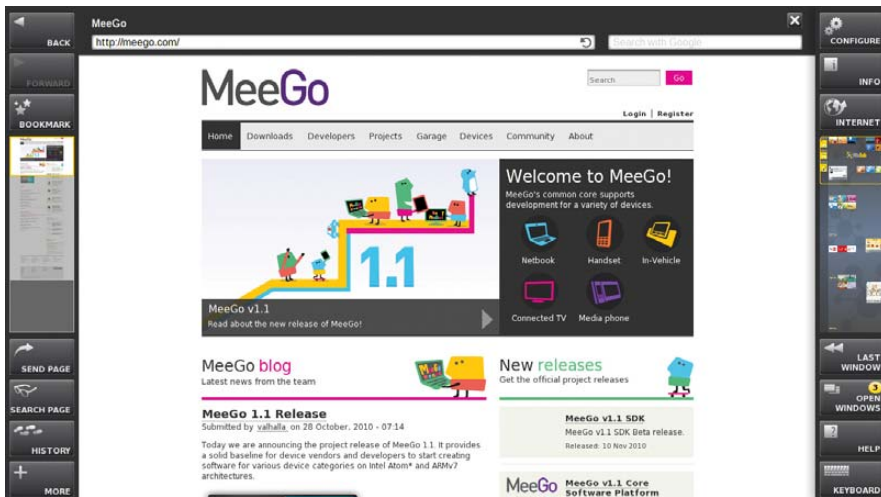


Figure 7. The Web browser on the WeTab with the MeeGo home page and the virtual touchscreen keyboard. Four different virtual work areas are on the right, and basic browser menu items are on the left.

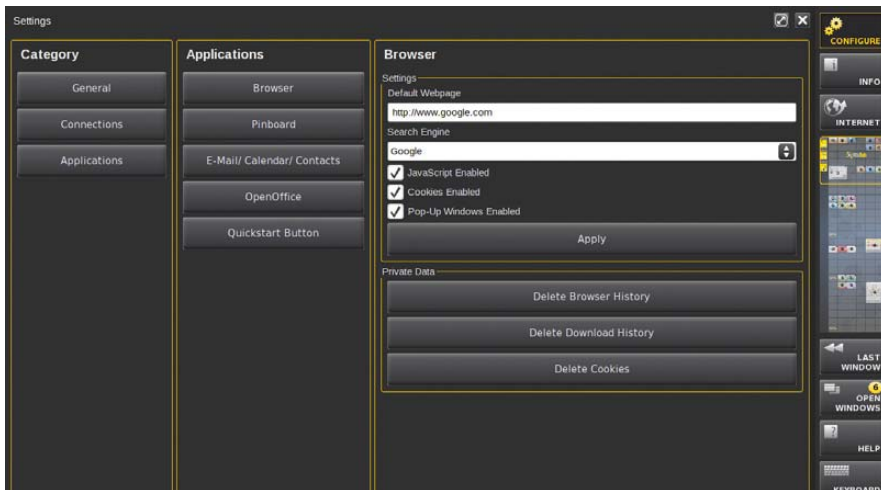


Figure 8. The WeTab Setting showcasing the browser settings.

Table 2. MeeGo IVI v1.1 Key Feature List

KEY FEATURE	DESCRIPTION
Sample IVI home screen and taskbar	The taskbar is designed with Automotive Center Console HMI requirements in mind.
Text-to-speech (TTS)	TTS is supported using Festival Speech Synthesis and is enabled by default in the ivihome menu navigation.
Speech recognition	Initial speech recognition has been added to ivihome using the integrated PocketSphinx 0.6.1 package. It's a lightweight, cross-platform engine that's built using the latest Sphinx speech recognition toolkit. PocketSphinx provides a GStreamer plugin, allowing the application to create a pipeline to parse the human voice, based on words defined in the dictionary. Voice commands for ivihome have been predefined for navigating the scroll menus.
MeeGo Touch Framework (MTF)	The MTF integration features sample applications, which include, but are not limited to, the following: video player, song player, photo viewer, hands-free dialer and settings management.
Open-source automotive projects	Several packages from open-source automotive projects are available from the repository for audio management, resource management, persistent storage management, CE device management and system health.



Figure 9. The MeeGo IVI home screen with the taskbar as it appears on the left side of the screen. The taskbar, with some easy customizations, can be moved to the right side of the screen to optimize access for the driver or passenger, as desired. It can be controlled by a Contour ShuttleXpress scroll wheel, touchscreen or mouse, and it's designed to reflect the scroll-wheel usage, with the ability to spin through the menu options and make selections or go back, by pressing two buttons or tapping the touchscreen (photo credit: meego.com).

vehicles, such as cars, trucks, planes, boats and buses.

Automotive manufacturers in particular are increasingly viewing IVI systems as key differentiators in their products. Drivers and passengers are coming to expect the same type of innovations they see in other devices, such as mobile computers and handsets, in their vehicles. As vehicles become connected to the Internet, the demand for Internet-based entertainment applications and services increases, and MeeGo strives to accelerate the pace of innovation in IVI. The MeeGo IVI software platform is designed to enable rich Internet and multimedia consumer experiences for vehicles. Table 2 provides a quick overview of the key features available in the MeeGo 1.1 IVI release.

MeeGo v1.1 Handsets UX

Today's users are demanding more powerful and feature-rich devices to take with them on the go. Next-generation smartphones allow users to enjoy a rich and dynamic Internet experience, watch HD movies and multitask like never before on a small-form-factor device. The MeeGo platform is specifically designed to enable the application and services ecosystem for these mobile, rich Internet and media-centric devices.

The MeeGo v1.1 Handsets UX (Figure 10) provides a technology snapshot

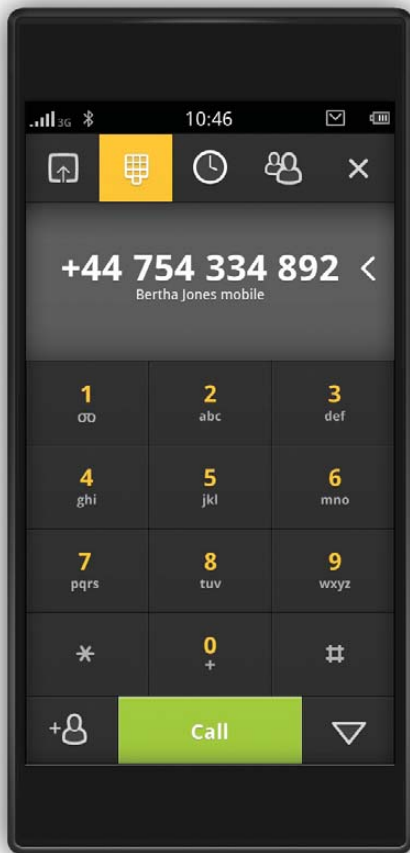


Figure 10. The MeeGo Handset Dialer (photo credit: meego.com)

that offers key handset technologies, such as cellular, connectivity, sensors and mobile browsing, as well as a basic development UX for voice calling, SMS messaging, Web browsing, music and video playback, photo viewing and connection management.

With this project release, developers will be able to work on future device and user experience software development, while simultaneously participating in the MeeGo Project to complete the Handset UX in the upcoming 1.2 release.

MeeGo SDK 1.1 Beta

The MeeGo SDK 1.1 beta was released on November 10, 2010, and it's available for download along instructions from meego.com/developers. It enables application developers to develop, install and debug applications, as well as run applications on the Nokia N900, Netbook and Aava devices with MeeGo. For developers without reference hardware, QEMU (qemu.org)

MeeGo on Handsets

If you have access to a Nokia N900 handset or an Aava Handset, you can find step-by-step instructions on how to install and run MeeGo on those devices from meego.com/devices/handset. The instructions will guide you through installing the root filesystem on an external micro-SD card. Give it a try, and have fun experimenting with it.

provides an emulated environment for debugging and testing applications.

The MeeGo Project encourages developers to use the MeeGo API, which currently consists of Qt 4.7 and Qt Mobility 1.0. The MeeGo API comes with a forward-compatibility promise and will be extended in future releases. The final nonbeta version of the SDK will be aligned with the MeeGo compliance specification (being finalized at the time of this writing). More information on MeeGo compliance is available at wiki.meego.com/Quality/Compliance.

Conclusion

MeeGo is an open-source project developed in public under the auspices of the Linux Foundation. Since it was announced in February 2010, the world has been able to watch and participate

as the project builds up and delivers the core software stack in addition to three reference user experiences for handsets,IVI systems and Netbooks, with more to come as MeeGo also targets connected TVs and tablets.

The development continues following a six-month release schedule. MeeGo 1.2 is scheduled for April 2011. Currently, there are hundreds of features targeting MeeGo 1.2 that already have been filed in the MeeGo Featurezilla (a tool that tracks feature development). The development tree of MeeGo 1.2 is open, and development is ongoing. ■

Ibrahim Haddad manages the Linux Foundation's Mobile Linux initiatives and works with the community to facilitate a vendor-neutral environment for advancing the Linux platform for next-generation mobile computing devices. Ibrahim is a Contributing Editor at *Linux Journal*.

Resources

MeeGo Project: www.meego.com

Developer Resources: meego.com/developers

Mailing Lists: meego.com/community/mailing-lists

IRC Discussions: meego.com/community/irc-channel

MeeGo Source Code: meego.gitorious.org

MeeGo Bugzilla: bugs.meego.com

MeeGo Forums: meego.com/community/forum

Precompiled Images: meego.com/downloads

MeeGo Releases: meego.com/downloads/releases

MeeGo SDK: meego.com/developers/getting-started

MeeGo OBS: wiki.meego.com/Build_Infrastructure

Is Your Personal Area Network Giving You the BlueZ?

Bluetooth is a complex beast, and recent changes to the standard Linux implementation have bamboozled many users. Untangle your personal area network with this revealing article on setting up the Bluetooth PAN Profile in BlueZ v.4. **CHUCK ELLIOT**

Bluetooth has been around for a while now (it originally was conceived in 1994 by Ericsson as a replacement for RS-232), and many of us have been enjoying the benefits of wirelessly connecting mice, keyboards, headphones, mobile phones, printers, PDAs and other devices over short distances.

Under Linux, these benefits are provided by one of two Bluetooth implementations: BlueZ and Affix. The former has become the accepted standard provided by most of the popular distributions and is the one I discuss here.

In BlueZ releases prior to version 4, the various Bluetooth services (called profiles) are implemented in separate daemons, each with its own configuration. The PAN profile, providing personal area networking services, is implemented in the `pand` daemon, dial-up networking in the `dund` daemon, service discovery in the `sdpd` daemon and so on. In BlueZ v.4, all of these are incorporated into one server process called `bluetoothd`. This change has led to a certain amount of confusion among users, so the aim of this article is to try to clarify the situation somewhat by looking at the way the PAN profile works in BlueZ v.4.

PAN Overview

The PAN profile provides three ways of connecting Bluetooth devices in a network capable of carrying general-purpose protocols and services over TCP/IP. Such networks are known as piconets, because they are limited to a maximum of seven nodes and the short distances covered by Bluetooth transmissions (up to ten meters). These are PANU, Group Ad Hoc Network (GN) and Network Access Point (NAP) configurations. The PANU<--->PANU network is the wireless analogue of an Ethernet crossover cable between two computers. The PANU<--->GN network operates just like a Wi-Fi ad hoc setup where one node acts as the master node connecting up to six client nodes in an isolated private network and the PANU<--->NAP provides network infrastructure connectivity in the same way that a 802.11 Wi-Fi access point does for Wi-Fi clients. The NAP can serve six client nodes in the same manner as the GN master but also connects them to an existing network infrastructure. The NAP service is the most useful of the three connection patterns and subsumes the other two. So, if you can get NAP working, it is relatively straightforward to achieve the other configurations by omitting certain steps in the NAP setup.

Time for a NAP

The Bluetooth daemon loads all the required kernel modules (the `lsmod` command should show `bluetooth`, `rfcomm`, `btusb`, `llc`, `sco`, `bridge`, `bnep`, `stp` and `l2cap` running) and creates a bridge-host interface called `pan0`. The `bnep` protocol provides Ethernet services over the Bluetooth `l2cap` connection and a separate `bnep` interface is created on demand for each remote device that connects (`bnep0`, `bnep1`...`bnepN`). `bnep` interfaces automatically are bridged to the `pan0` interface that serves as an IP proxy for them. Thus, only the `pan0` interface requires IP configuration. Once you have paired a remote Bluetooth device with the Linux box and discovered the BlueZ NAP service and connected/authenticated to it, all that remains is to provide IP configuration for the connected devices and the required infrastructure network connectivity. This is the part that has bamboozled many a keen BlueZman. Here, I consider two solutions: a bridged setup and a routed setup.

Take It to the Bridge

In many ways, the bridged solution is the simpler of the two. There are fewer steps involved, and it means that your NAP and your remote Bluetooth devices become part of your existing IP network/subnet. However, if you don't have an existing private network/subnet because your Linux box connects straight to your ISP (say via an Ethernet cable to an ADSL router) or your Linux box connects to your existing private network/subnet via an 802.11 Wi-Fi connection, the routed solution is the only way to go. You need a private IP subnet from which to allocate addresses to your remote BT devices, but you can't bridge to this via a `wlan0` (802.11 Wi-Fi) interface because layer 2 of the wireless device is not available to the kernel bridging facilities. Without a bridge, you will require a separate IP network/subnet, routing and routing information, but luckily, this is not too hard to arrange.

The Bridged Solution

Let's assume you have BlueZ 4 installed. If not, use your distribution's installation software to install it. You will need version 4 of the BlueZ and BlueZ-libs packages. The BlueZ kernel modules are incorporated into the 2.4 and 2.6 kernels. Another useful tool is provided by the `gnome-bluetooth` package in the form of a BT panel applet that allows control from the desktop, allowing you to pair devices and set service discovery. KDE provides a similar tool,

If you don't already run DHCP, now could be a good time to start! It isn't difficult to set up a simple service, and it reduces the amount of general, routine manual configuration.

kbluetooth. Start the Bluetooth daemon in the normal way, and execute the bluetooth-applet from a shell. Enable Service Discovery in the bluetooth-applet, and introduce a remote BT device into range. You now should be able to discover the BlueZ-NAP service from the remote device. If you are not using the GNOME or KDE desktop, device pairing and service discovery are available through the hciconfig and hcitool utilities that are part of the BlueZ package.

If you are following this solution, I will assume you have a single ISP-assigned, routable IP address that provides your Internet connectivity (Figure 1). Your local private network (say 192.168.0.0/24) uses Network Address Translation (NAT) provided by your router to access the Internet through your ISP connection, and the Linux box on which you want the BT connectivity connects to the private network via an eth0 interface to an Ethernet switch. All that being true, you are only three steps away from your bridged NAP solution:

1. Make your eth0 interface part of the pan0 bridge.
2. Configure the pan0 interface for IP.
3. Provide IP configuration for remote BT devices.

You can achieve step one by use of the brctl command:

```
brctl show (display any extant bridges - should show pan0)
brctl add pan0 eth0 (add the eth0 interface to the pan0 bridge)
brctl showmacs pan0 (show which interfaces are bridged)
```

You can use ifconfig to achieve step 1 temporarily:

```
ifconfig pan0 192.168.0.123 netmask 255.255.255.0
➔ broadcast 192.168.0.255 up
```

Use your distribution's network management tools to create a permanent configuration, and choose IP values to suit your own

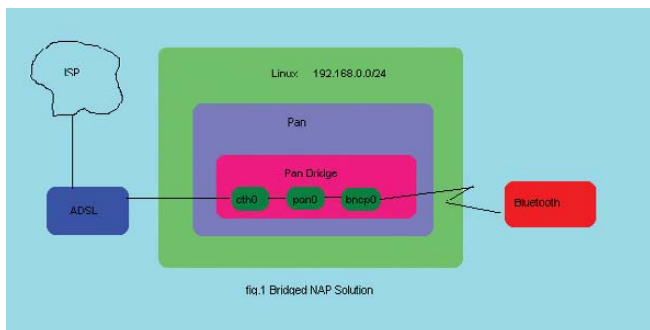


Figure 1. Bridged NAP Solution

private network topology.

The neatest solution for step three is to run a small DHCP service. If you already run DHCP for your own private network, you just have to make sure that the configuration covers any remote BT devices you might connect in addition to the clients already connected. The pan0 interface also can get its IP configuration in this way. If you don't already run DHCP, now could be a good time to start! It isn't difficult to set up a simple service, and it reduces the amount of general, routine manual configuration.

Below is a simple dhcpd.conf file sufficient to serve our PAN:

```
# small DHCP config for bluetooth PAN
ddns-update-style none;
authoritative;

subnet 192.168.0.0 netmask 255.255.255.0 {
  range 192.168.0.10 192.168.0.100;
  option domain-name-servers 192.168.0.1;
  option domain-name "bluetooth.net";
  option routers 192.168.0.1;
  option broadcast-address 192.168.0.255;
  default-lease-time 600;
  max-lease-time 7200;

  host btmobile {
    hardware ethernet 74:e7:71:ac:d0:34;
    fixed-address 192.168.0.9;
  }
}
```

The subnet declaration should correspond to your local private network topology. The range declaration specifies a pool of IP addresses that can be assigned dynamically. The options specify DNS servers (this could be your ISP's nameservers), default gateway (the NAT router) and other common IP parameters. The host declaration allows you to assign fixed IP addresses to known (by MAC address) devices. Place this file where your DHCP server expects to find it, and start the server by the method appropriate to your distribution.

Needless to say, your remote BT devices and the pan0 interface should be configured to get their IP configuration via DHCP. Once all this is in place, your NAP should be active a few seconds after you connect/authenticate your BT device to the Linux box.

The Route of the Problem

For the routed solution, you need to create a separate network/subnet for your PAN and route packets between this and the existing private network/ISP-connected machine. In this configuration pan0 will be the default gateway device and also provide the DHCP service for the PAN. Thus, pan0 will require a static IP

configuration and its DHCP service will be separate from any other DHCP and dedicated to the PAN. For clarity, let's use the 10.0.0.0/24 private IP network range for this purpose. This time, there are four steps to get your routed PAN up and running:

1. Configure pan0 with static IP information.
2. Configure and run DHCP for the PAN.
3. Turn on IP forwarding to route packets between pan0 and wlan0/eth0.
4. Enable NAT on the wlan0/eth0 interface.

Then:

```
ifconfig pan0 10.0.0.1 netmask 255.255.255.0  
➔broadcast 10.0.0.255 up
```

Use your system tools to make permanent settings.
Below is a small DHCP configuration for the routed PAN:

```
# small dhcp config for bluetooth PAN
```

```
ddns-update-style none;  
authoritative;
```

```
subnet 10.0.0.0 netmask 255.255.255.0 {  
  range 10.0.0.10 10.0.0.100;  
  option domain-name-servers 192.168.0.1;  
  option domain-name "bluetooth.net";  
  option routers 10.0.0.1;  
  option broadcast-address 10.0.0.255;  
  default-lease-time 600;  
  max-lease-time 7200;
```

```
host btmobile {  
  hardware ethernet 78:e7:d1:ab:d5:6f;  
  fixed-address 10.0.0.5;  
}  
}
```

Alter the DNS information as necessary.

pan0 should be configured and up and running before you start the DHCP service. The next step involves enabling IP forwarding so that packets are routed between pan0 and your existing interface (wlan0/eth0).

In older systems, IP forwarding is enabled by echoing a value

4TH ANNUAL **COLUMBIA, SC**
PALMETTO OPEN SOURCE SOFTWARE CONFERENCE

PEOPLE FROM 14 STATES ATTENDED IN 2010.
2011 WILL BE BIGGER AND BETTER!
EARLY BIRD REGISTRATION AVAILABLE!

POSSCON
PALMETTO OPEN SOURCE SOFTWARE CONFERENCE

MARCH 23-25, 2011
at the
COLUMBIA CONVENTION CENTER
Columbia, South Carolina

FEATURING:
LATEST OPEN SOURCE TECHNOLOGY
WORLD-CLASS SPEAKERS
HACKATHONS AND BUG SQUASHES
RESUMÉ DROP AND "SUITS MEET SHORTS" EVENT
WORKSHOPS

FOR MORE INFORMATION, GO TO
WWW.POSSCON.ORG
OR CALL EVENT CHAIR AT (803) 240-1213

presented by
CONSORTIUM
FOR ENTERPRISE SYSTEMS MANAGEMENT

FAMOUS FOR LOW REGISTRATION FEES AND WORLD-CLASS SPEAKERS

of "1" into the appropriate system file:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

In more recent systems, this is achieved by editing `/etc/sysctl.conf`:

```
# Controls IP packet forwarding
net.ipv4.ip_forward = 1
```

The setting also may be available in your distribution's firewall configuration GUI tool. If none of these methods work on your system, consult your distribution's documentation.

Finally, you need to tell the kernel netfilter software to "masquerade" (NAT) PAN packets through the `wlan0/eth0` interface. You may be able to do this using your distribution's firewall configuration GUI tool (the IP forwarding setting may be available here too). If not, it can be achieved using the `iptables` command:

```
iptables -A POSTROUTING -t nat -o wlan0 -j MASQUERADE
```

Substitute `eth0` for `wlan0` if you are using the routing option because your Linux box connects directly to your ISP and your `eth0` interface has an ISP-assigned, routable IP address (Figure 2).

BlueZ v.4 does not appear to provide separate configurations for GN and PANU modes of PAN operation, but this is of no consequence because, as was noted above, these are subsumed by the NAP mode.

If all has gone well up to this point, your NAP service should be active a few seconds after you connect/authenticate your remote BT device. You should, for example, be able to ping the device from your Linux box. If IP is not yet running, you can use `l2ping <MAC-ADDRESS>` to ping the remote device.

Oh, for the Simple Life

BlueZ v.4 does not appear to provide separate configurations for GN and PANU modes of PAN operation, but this is of no consequence because, as was noted above, these are subsumed by the NAP mode. If you only want to connect remote BT devices to your Linux box, and do not require access to the

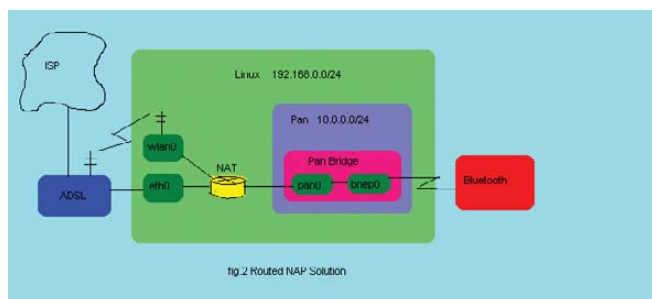


Figure 2. Routed NAP Solution

local network or Internet, you simply can omit step 1 from the bridged solution and either employ the DHCP configuration from the routed solution or manually set IP parameters for `pan0` and your BT devices.

In fact, Bluetooth is supposed to implement the draft Link-local Autoconfiguration Protocol (variously known elsewhere as Avahi, Bonjour, Rendezvous and APIPA), so you could try using this for IP configuration instead of running a DHCP service. However, I had no joy with this approach under BlueZ v.4, so I leave it as a potential solution for those of an experimental nature. I would be happy to hear that this facility is alive and well in the BlueZ package if anyone succeeds where I have failed.

Nostalgia Is Not What It Used to Be

For those who want to retain the old ways of configuring and running the Bluetooth facilities, the development team has provided a legacy implementation in the form of a package that contains the separate daemons as provided in BlueZ 3.x. This package is called `BlueZ-compat` and should satisfy the change-resistant among you. Michael Schmidt (see Resources) has produced a useful how-to document covering the legacy formats.

What's Up Doc?

I am always very reluctant to criticize developers of free and open software. It is all too easy to forget or take for granted

the enormous amount of dedicated work that goes into projects like BlueZ and to moan and gripe over minor bugs and missing features. On the whole, I have nothing but admiration for these wonderful people. But, if I could just gently and humbly raise one point, it would be that the documentation for BlueZ seems very thin on the ground. In fact, that's what motivated me to write this article in the first place (for example, there is no man page for the `/etc/bluetooth/main.conf` file). Anyway, let's hope this is a temporary glitch in an otherwise unblemished record of developmental excellence. ■

Dr Chuck Elliot is Principal Lecturer in Networking & Information System Security at Sheffield Hallam University and a Red Hat Certified Engineer.

Resources

How to Set Up Common PAN Scenarios with BlueZ's Integrated PAN Support, by Michael Schmidt:
bluez.sourceforge.net/contrib/HOWTO-PAN

Personal Area Networking Profile, Version 1.0:
www.bluetooth.com/SiteCollectionDocuments/PAN_SPEC_V11.pdf

Polywell Industrial Mini-PCs

Cost Effective Embedded PC For Appliances



NVIDIA ION™



ITX-10A



ITX-30A with PCI Riser



ITX-30G with NVIDIA® ION™ Graphics
Barebone system **\$199**



ITX-50 Series



ITX-40A with Slim Optical Bay



VESA / Wallmount option



Full Height Riser or Low-Profile Add-on Slot
up to 8 x 2.5" or 4 x 3.5" Hard Drives

Over 250 Mini-ITX Models Available:

- NVIDIA® GeForce 8200/8100 with AMD® Athlon/Phenom Processor
- NVIDIA® GeForce 9300/9100M/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

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.



Polywell Storage Servers

More Choices, Excellent Service, Great Prices!

Netdisk 9015N



Quiet Storage NAS/SAN/iSCSI

10TB \$1,799

20TB \$2,999

30TB \$4,999

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

Silent Eco Green PC

The Best Terminal PC

Intel® Low Voltage Processor
Energy efficient, Quiet Platform.
starts at **\$199**



Mini-ITX LD-001

High Performance 5048A



5U-48Bay 96TB Storage Server

4U45JB / 4U36A / 4U24A
RAID-6, NAS/iSCSI/SAN Storage
Mix SAS / SATA, 4 Giga / 10Gbit LAN



4U-24Bay 48TB \$8,999
4U-36Bay 72TB \$12,500
4U-45Bay 90TB JBOD

2008A / 2012A / 2012B
RAID-6, NAS/iSCSI/SAN Storage
Mix SAS / SATA, 4 GigaLAN



2008A 16TB \$2,699
2012A 24TB \$3,999
2012B 24TB \$4,899

1U945GCL2



Mini-1U Server \$499

Intel® Dual-Core Processor, 2 x 500G RAID
Dual GigaLAN, 4GB DDR2 RAM

- Over 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/Storage

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

Intel is trademarks of Intel Corporation. Other names are for informational purposes only and may be trademarks of their respective owners.



Freeing Up Level 7

Your own life is the last frontier. DOC SEARLS



There was a time when the decisions you made about what you did with apps started down at OSI Layer 1: with wiring. That's because everybody sold "integrated" systems that were silo'd up every layer of the OSI stack, which looked like this:

7. Application
6. Presentation
5. Session
4. Transport
3. Network
2. Data Link
1. Physical

(Yes, there are other ways of sorting these out. But these seven are the most widely used, so bear with me.)

So, several decades ago, if you had an IBM system 34, 36 or 38, you needed twinaxial wiring at the bottom and apps written just for those systems on top. Even different IBM systems used different wiring, networking and apps.

Efforts to make wiring more generic took place a level or two up. That's what Ethernet and IBM's own Token Ring did. But, there were lots of different Ethernet and Token Ring products, each with their own special wiring as well. These too were problems to be solved. Novell got that started in the mid-1980s with NetWare, and TCP/IP finished the job by taking care of layers 3 and 4 (network and transport) for everybody and everything, through the Internet. TCP/IP's liberating effects go both up the stack and down. You can

put whatever you want under it or over it, without worrying about what's on the other side.

Our lives sit atop Layer 7. This is the layer we see when we look at the collection of apps on the mobile devices that go in our pockets. Since we're talking Linux here, that mostly means Androids, which passed the 100,000 app mark in late October 2010. For the sake of the point I'm about to make, let's add the 300,000 apps that also are on iPhone, and the thousands each on Windows, Symbian and other platforms. My point is that the OSes on which they run shouldn't limit how you connect them to whatever you want, including other apps.

But, most of the apps we get are extensions of servers and services. At that level, lots of great stuff is happening, especially with APIs. Facebook's and Twitter's APIs give you a single-sign-on shortcut on many sites. You also can connect services, such as by making your Flickr photo uploads appear on Facebook.

But, what can we do all by ourselves to connect apps and data together in our pockets? For that, we're still early.

For example, let's say you're a surfer. You have a surfing app on your phone. You also have a calendar, a map app with traffic information and a fitness app with data gathered from your bathroom scale, your sleep monitoring system and your workout app. In the old world, where we still live, the first place most of us naturally look for putting these things together is within and across the different services provided by the makers of those apps and devices. But, why not program these connections for ourselves? That's the only way we can be fully autonomous at Level 7.

"Level 7 is the last frontier", Craig Burton says. At Novell in the 1980s, he made NetWare liberate dependencies

from levels 1 and 2. Now he's working with Kynetx, a startup in Utah. Rather than looking at the world as a bunch of clients and servers, Kynetx sees end points, events, rules, data and rules engines. To those, they've added some definition and a language (KRL) for writing rules. Using KRL, you can program rules to say surf's up (when it actually is, in real time), but only after also noticing that you're five pounds overweight, it's four days since you've exercised, you've had enough sleep, you have no appointments coming up, and the traffic is clear. Notice that all the control here is in the hands of the user. (The same can be true for companies, which can write their own rules too, but our focus here is personal.)

Back in the client-server world, which is still with us, we now have a proactive model, in which there are what Craig Burton calls "full-duplex" interactions across APIs, live. This is new too.

Here in Cambridge, Massachusetts, I've also been talking with Ben Rubin, founder and CTO of Zeo, another developer whose work I dig. Zeo makes the Sleep Coach, a device that monitors the health of your sleep. (Yes, it runs on Linux.) Zeo's corporate friends include RunKeeper, DalilyBurn, Digifit and Withings. I already have a Withings bathroom scale. I just got my Zeo set up today. And, I've also just started working out at the local Y. Over the next couple months, I'm going to see how these all mash up, keeping in mind how I'd like to control the ways these sites, services, apps and devices let me do that—on my own—and how new frameworks like Kynetx's make that possible. I'll let you know how it goes. ■

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.

iX-Green Neutron 10G Rackmount Servers: Energy Efficient, Next-Generation 10 Gigabit Performance

➤ INNOVATIVE TECHNOLOGY ➤ HIGH DRIVE DENSITY ➤ LOW POWER ENVELOPE

The new **iX2216-10G 2U Rackmount Server** features built-in Intel® 10 Gigabit Ethernet Adapters, resulting in increased performance in an energy-efficient design.

**Intel® 10 Gigabit Ethernet Adapters
built onto the motherboard**



iXsystems announces a new 2U Rackmount Server that is equipped with energy-efficient, next-generation 10 Gigabit performance. 10 Gigabit Ethernet provides unparalleled throughput for virtualized servers and unified networks, and has emerged into the enterprise and service provider space as a viable, low-cost networking alternative to Fibre Channel.

The iX2216-10G features the latest Intel® processors based on the 32nm and 45nm next-generation microarchitecture, which represents the next step in intelligent performance, automated energy efficiency, and flexible virtualization.

The iX2216-10G also features dual on-board Intel® 82599EB 10 Gigabit SFP+ Ports, dual on-board Intel® 82576 Gigabit Ports, dual Intel® Xeon® Processors 5600/5500 series, and 18 DIMM slots supporting up to 192GB of DDR3 ECC Registered memory. This system would be ideal for HPC, Data Center, Virtualization, Clustering, and Cloud Computing applications.

iXsystems is dedicated to bringing the energy saving advantages of 2.5" disk drives and high-efficiency power supply technology to our customers. With the iX2216-10G, you can have up to 16 x 2.5" high-rotation SAS or SATA drives. The redundant AC-DC high-efficiency 920W, 94%+ certified power supplies, will also help lower your total cost of ownership.

For more information on the **iX-2216-10G Rackmount Server**, or to request a quote, visit: <http://www.iXsystems.com/GN10G>

Key features:

- Supports Dual 64-Bit Six-Core, Quad-Core or Dual-Core, Intel® Xeon® Processor 5600/5500 Series
- 2U Form Factor with 16 Hot-Swap SAS/ SATA 2.5" Drive Bays
- Intel® 5520 chipset with QuickPath Interconnect (QPI)
- Up to 192GB DDR3 1333/1066/800 SDRAM ECC Registered Memory (18 DIMM Slots)
- 2 (x8) PCI-E 2.0 slots + 1 (x4) PCI-E 2.0 (in x8 slot -Low-Profile - 5.5" depth)
- Dual Port Intel® 82599EB 10 Gigabit SFP+ - Dual Port Intel® 82576 Gigabit Ethernet Controller
- Matrox G200eW Graphics
- Remote Management - IPMI 2.0 + IP-KVM with dedicated LAN
- Slim CD-ROM / DVD-ROM and/or 3.5" DVD-RW Drive
- 920W high-efficiency (94%+) AC-DC Redundant power supplies with PMBus and I2C



Call iXsystems toll free or visit our website today!
+1-800-820-BSDi | www.iXsystems.com



Cut Execution Time by >50% with WhisperStation-GPU

Delivered ready to run new GPU-enabled applications:

Design

3ds Max
Bunkspeed
Shot
Adobe CS5

Simulation

ANSYS Mechanical
Autodesk Moldflow
Mathematica

MATLAB
ACUSIM AcuSolve
Tech-X GPULib

BioTech

AMBER
GROMACS
NAMD, VMD
TeraChem

Integrating the latest CPUs with NVIDIA Tesla Fermi GPUs, Microway's WhisperStation-GPU delivers 2x-100x the performance of standard workstations. Providing explosive performance, yet quiet, it's custom designed for the power hungry applications you use. Take advantage of existing GPU applications or enable high performance with CUDA C/C++, PGI CUDA FORTRAN, or OpenCL compute kernels.

- ▶ Up to Four Tesla Fermi GPUs, each with: 448 cores, 6 GB GDDR5, 1 TFLOP single and 515 GFLOP double precision performance
- ▶ Up to 24 cores with the newest Intel and AMD Processors, 128 GB memory, 80 PLUS® certified power supply, and eight hard drives
- ▶ Nvidia Quadro for state of the art professional graphics and visualization
- ▶ Ultra-quiet fans, strategically placed baffles, and internal sound-proofing
- ▶ New: Microway CL-IDE™ for OpenCL programming on CPUs and GPUs



WhisperStation with 4 Tesla Fermi GPUs

Microway's Latest Servers for Dense Clustering

- ▶ 4P, 1U nodes with 48 CPU cores, 512 GB and QDR InfiniBand
- ▶ 2P, 1U nodes with 24 CPU cores, 2 Tesla GPUs and QDR InfiniBand
- ▶ 2U Twin² with 4 Hot-Swap MBs, each with 2 Processors + 256 GB
- ▶ 1U S2050 servers with 4 Tesla Fermi GPUs

Microway Puts YOU on the Cutting Edge

Design your next custom configuration with techs who speak HPC. Rely on our integration expertise for complete and thorough testing of your workstations, turnkey clusters and servers. Whether you need Linux or Windows, CUDA or OpenCL, we've been resolving the complicated issues – so you don't have to – since 1982.

Configure your next WhisperStation or Cluster today!

microway.com/quickquote or call 508-746-7341

Sign up for technical newsletters and special GPU promotions at microway.com/newsletter



OcioPuter™ 4U Server with up to 8 GPUs and 144 GB memory

1U Node with 2 Tesla Fermi GPUs

2U Twin² Node with 4 Hot-Swap Motherboards
Each with 2 CPUs and 256 GB



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

Microway
Technology you can count on™