KVM | Clutter | FPGAs | Android | Coreboot | RSpec | $RANDOM

LINUX JOURNAL™

Since 1994: The Original Magazine of the Linux Community

OCTOBER 2009 | ISSUE 186 | www.linuxjournal.com

LINUX KVM for Low-Level System Programming

Bare-Metal Hacks with **FPGA PROGRAMMING**

coreboot

Unchain Your BIOS with the Ultimate Hack

PLUS:

PUT ANDROID ANYWHERE

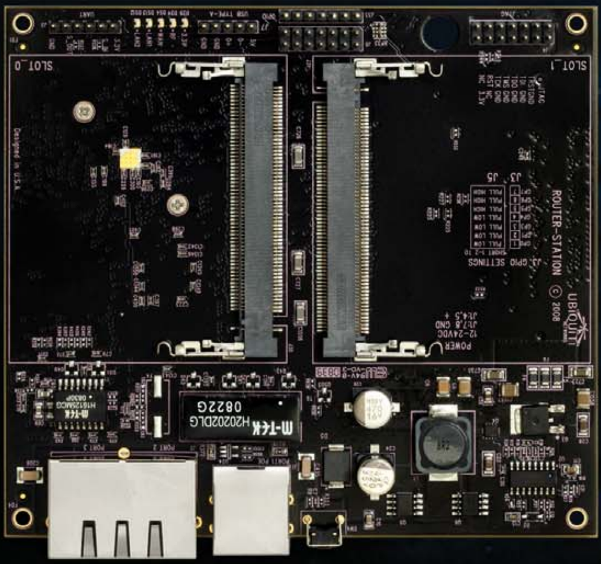
Spam-Fighting Tricks for Mutt and Postfix

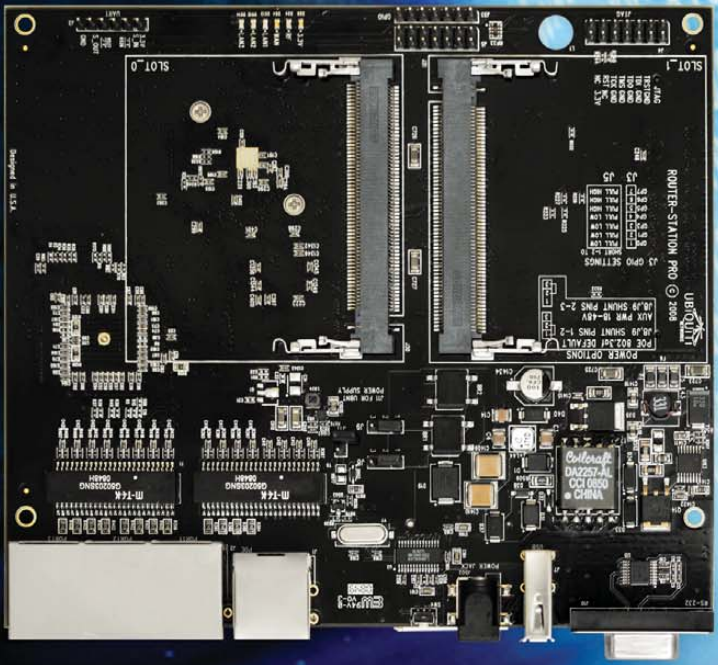
Develop OpenGL Apps with Clutter

Laptop Hardening Tips

UBiQUiTi
NETWORKS

Embedded Wireless Dream Machines.





RouterStation

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

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

MSRP $59

RouterStation Pro

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

Pro Version Enhancements:

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

MSRP $79

www.ubnt.com

CONTENTS

OCTOBER 2009
Issue 186

FEATURES

HACK THIS

ON THE COVER

Gemini²: The Fantastic Four

IXsystems is proud to introduce the latest offering in our iX-Gemini line, the **Gemini²**. Cleverly disguised as any other 2U server, the **Gemini²** secretly houses 4 highly efficient, extremely powerful RAID 5 capable servers. Each node supports the latest Intel® 5500 series processors, up to 48GB of DDR3 memory, and three 3.5" hot-swappable hard drives.

This system architecture achieves breakthrough x86 server performance-per-watt (375 GFLOPS/kW) to further satisfy the ever-increasing demands for efficiency, density and low-TCO of today's high performance computing (HPC) clusters and data centers. For more information and pricing, please visit our website at http://www.iXsystems.com/gemini2.

Features

Four hot-pluggable systems (nodes) in a 2U form factor

Each node supports the following:

- Dual 64-Bit Socket 1366 Quad-Core or Dual-Core Intel® Xeon® Processor 5500 Series
- 3 x 3.5" SAS/SATA Hot-swappable Drive Bays
- Intel® 5520 Chipset with QuickPath Interconnect (QPI)
- Up to 48GB DDR3 1333/1066/800 SDRAM ECC Registered Memory
- 1 (x16) PCI-E (Low Profile)
- Matrox G200eW 8 MB DDR2 Memory Video
- Integrated Remote Management - IPMI 2.0 + IP-KVM with dedicated LAN
- All four nodes share a Redundant 1200W High-efficiency Power Supply (Gold Level 92%+ power efficiency)

iXsystems™

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

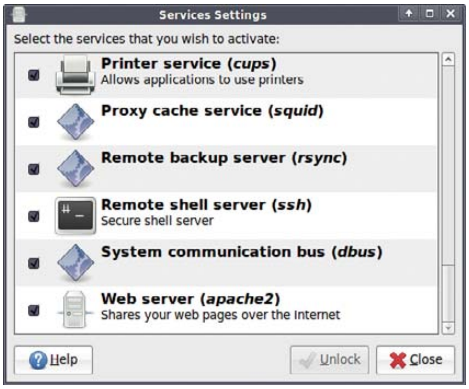
intel® Xeon® inside™

Powerful.
Intelligent.

Intel, the Intel logo, and Xeon Inside are trademarks or registered trademarks of Intel Corporation in the U.S. and other countries.

CONTENTS OCTOBER 2009 Issue 186

COLUMNS



Services Settings

Select the services that you wish to activate:

☑ **Printer service (*cups*)**
Allows applications to use printers

☑ **Proxy cache service (*squid*)**

☑ **Remote backup server (*rsync*)**

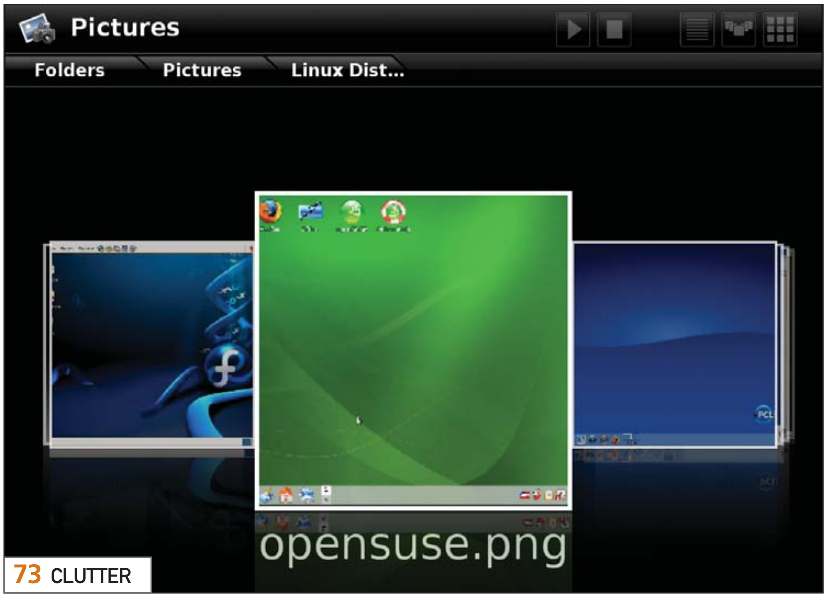
☑ **Remote shell server (*ssh*)**
Secure shell server

☑ **System communication bus (*dbus*)**

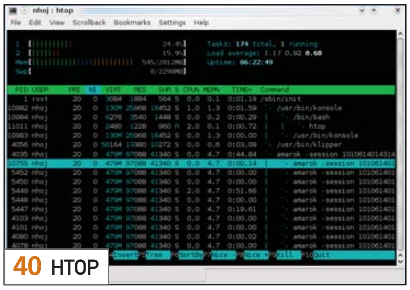
☑ **Web server (*apache2*)**
Shares your web pages over the Internet

INDEPTH

IN EVERY ISSUE

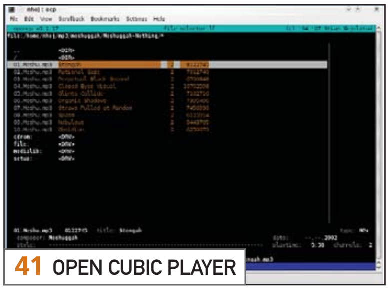


73 CLUTTER



```
  1  [|||||||||||                24.4%]     Tasks: 174 total, 1 running
  2  [||||||||                   15.9%]     Load average: 1.17 0.92 0.68
  Mem[||||||||||||||||||||||||| 545/2012MB]  Uptime: 06:22:49
  Swp[                           0/2290MB]

  PID USER     PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
    1 root      20   0  3084  1884   564 S  0.0  0.1  0:01.19 /sbin/init
10982 nhoj      20   0  130M 25968 16452 S  1.0  1.3  0:01.59  `- /usr/bin/konsole
10984 nhoj      20   0  6276  3540  1448 S  0.0  0.2  0:00.29  |   `- /bin/bash
11011 nhoj      20   0  2480  1228   960 R  2.0  0.1  0:00.72  |   |   `- htop
10983 nhoj      20   0  130M 25968 16452 S  0.0  1.3  0:00.00  |   `- /usr/bin/konsole
 4056 nhoj      20   0 56164 13380 10272 S  0.0  0.6  0:03.09  `- /usr/bin/klipper
 4035 nhoj      20   0  479M 97088 41340 S  0.0  4.7  0:44.84  `- amarok -session 1010614014314
10755 nhoj      20   0  479M 97088 41340 S  0.0  4.7  0:00.14  |   `- amarok -session 101061401
 5452 nhoj      20   0  479M 97088 41340 S  0.0  4.7  0:00.00  |   `- amarok -session 101061401
 5450 nhoj      20   0  479M 97088 41340 S  0.0  4.7  0:00.00  |   `- amarok -session 101061401
 5449 nhoj      20   0  479M 97088 41340 S  2.0  4.7  0:51.66  |   `- amarok -session 101061401
 5448 nhoj      20   0  479M 97088 41340 S  0.0  4.7  0:00.00  |   `- amarok -session 101061401
 5447 nhoj      20   0  479M 97088 41340 S  0.0  4.7  0:19.61  |   `- amarok -session 101061401
 4103 nhoj      20   0  479M 97088 41340 S  0.0  4.7  0:00.00  |   `- amarok -session 101061401
 4101 nhoj      20   0  479M 97088 41340 S  0.0  4.7  0:00.00  |   `- amarok -session 101061401
 4080 nhoj      20   0  479M 97088 41340 S  0.0  4.7  0:00.02  |   `- amarok -session 101061401
 4078 nhoj      20   0  479M 97088 41340 S  0.0  4.7  0:00.00  |   `- amarok -session 101061401
        F4Invert F5Tree  F6SortBy F7Nice - F8Nice + F9Kill  F10Quit
```

40 HTOP



```
file://home/nhoj/mp3/meshuggah/Meshuggah-Nothing/*

 ..            <DIR>
 /             <DIR>
 01.Meshu.mp3
 02.Meshu.mp3
 03.Meshu.mp3
 04.Meshu.mp3  Closed Eye Visuals
 05.Meshu.mp3  Glints Collide
 06.Meshu.mp3  Organic Shadows
 07.Meshu.mp3  Straws Pulled at Random
 08.Meshu.mp3
 09.Meshu.mp3
 10.Meshu.mp3
 cdrom:        <DRV>
 file:         <DRV>
 medialib:     <DRV>
 setup:        <DRV>

 01.Meshu.mp3                  title: Stengah
   composer: Meshuggah                                        date:   --.--.2002
```

41 OPEN CUBIC PLAYER

Next Month: INFRASTRUCTURE

Everybody loves infrastructure, even the government—if you consider that keeping good company. If not, at least you're keeping company with all the Linux users out there building up the Internet and corporate infrastructure, making everyone safe from those other operating systems.

To assist you in the process, next month's Infrastructure issue brings you articles on Storage Area Networks (SAN) on Linux, Host Identity Protocol for Linux (HIPL) and the Advanced Message Queuing Protocol (AMQ/AMQP).

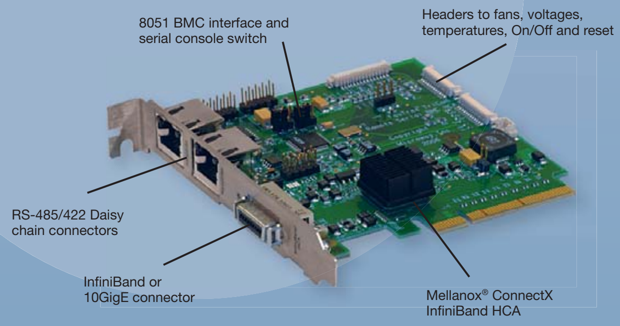
But wait, that's not all. If your Infrastructure plans go beyond the mere terrestrial, don't miss our article on IBM's InfoSphere Streams and Uppsala University's space weather project. Now that's data processing with a capital D!

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

Your Applications Will Run Faster
With Next Generation Microway Solutions!

TriCom™ X

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


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

Teraflop GPU Computing

For Workstations and HPC Clusters

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

NumberSmasher®

Large Memory Scalable SMP Server

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

FasTree™ X

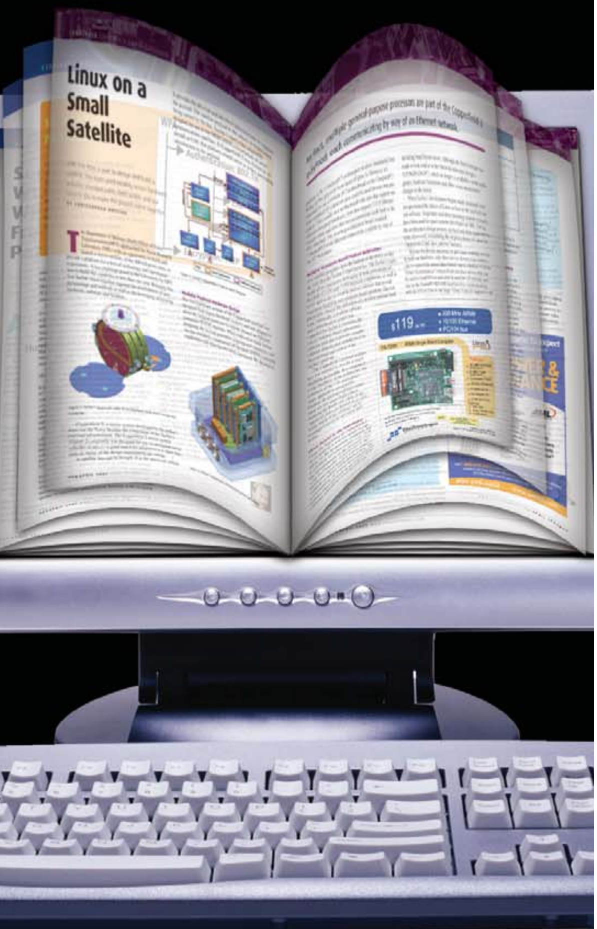
- Mellanox® InfiniScale™ IV Technology
- QDR/DDR InfiniBand Switches
- Modular Design
- 4 GB/sec Bandwidth per Port
- QSFP Interconnects
- InfiniScope™ Real Time Diagnostics

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

508-746-7341

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

Microway®
Technology you can count on℠



LINUX JOURNAL

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

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

Proofreader	Geri Gale

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

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

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

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

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

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

LINUX is a registered trademark of Linus Torvalds.

PRINTED WITH SOY INK®

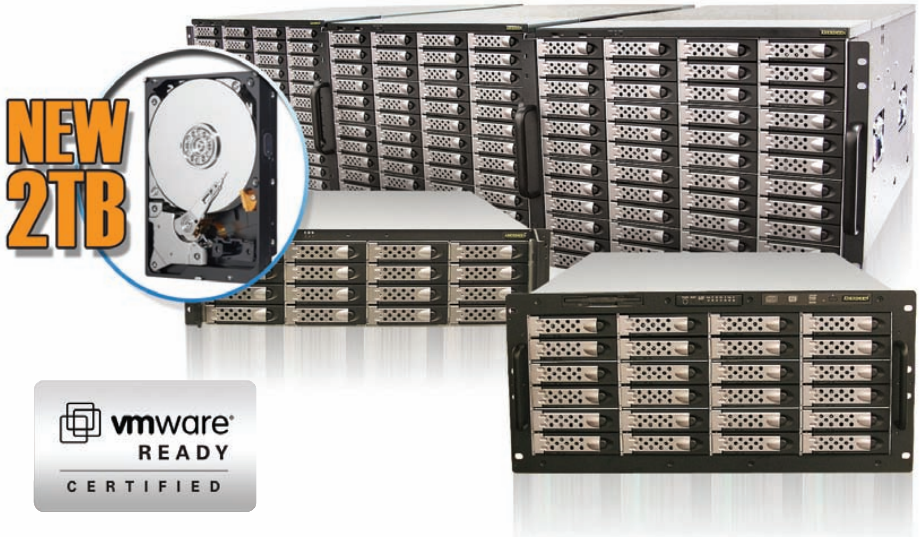
ABERDEEN
SERVERS AND STORAGE

HOW MUCH VMWARE DO YOU NEED?

VMware Certified NAS and Servers

The scalable AberNAS is the only Network Attached Storage appliance in its class to carry the VMware® Ready Certification. Aberdeen enables VMware-centric organizations of any size to affordably take advantage of AberNAS's robust features such as; superior RAID controller speed, increased scalability and optional NAS-to-NAS mirroring.

VMware Certified Aberdeen Stirling servers offer a lower cost of ownership, while eliminating concerns of compatibility complications. Stirling servers possess exceptional benefits designed to deliver improved operational efficiency within a dynamic VMware environment.



vmware READY CERTIFIED

ABERNAS

VMware Certified NAS Up To 100TB

- Quad-Core Intel® Xeon® Processor 5500 Series
- 3GB/6GB 1333MHz DDR3 memory
- Supports both SAS & SATA Storage Drives
- RAID 0, 1, 5, 6, 10, 50, 60 Capable
- SAS Expansion Ports & iSCSI Target Capable
- Dual/Quad Gigabit Ethernet Ports
- 5-Year Warranty

1U - 8TB NAS Starting at	$4,995
2U - 24TB NAS Starting at........	$10,495
3U - 32TB NAS Starting at........	$12,995
5U - 48TB NAS Starting at........	$17,995
6U - 64TB NAS Starting at........	$23,495
8U - 100TB NAS Starting at......	$33,995

intel® Xeon® inside™
Powerful. Intelligent.

STIRLING 123

1U Dual Xeon Server

- Up to 2x Quad-Core Intel® Xeon® Processor 5400 Series
- 32GB FBDIMM max memory
- Up to 4 SATA or SAS drives
- 650W redundant power
- Pre-installed VMware® ESXi on Disk-on-Module
- 5-Year Warranty

Starting at $1,845

STIRLING 132T

1U Twin Node Server

- Up to 2x Quad-Core Intel Xeon Processor 5400 Series
- 64GB FBDIMM max memory
- Up to 2 SATA drives
- 980W high efficiency power
- Pre-installed VMware ESXi on Disk-on-Module
- 5-Year Warranty

Starting at $2,675

STIRLING 209

2U Xeon Server

- Single Quad-Core Intel Xeon Processor 5500 Series
- 24GB DDR3 max memory
- Up to 6 SATA drives
- 400W high efficiency power
- Pre-installed VMware ESXi on Disk-on-Module
- 5-Year Warranty

Starting at $1,495

STIRLING 244

2U Quad Xeon Server

- Up to 4x Six-Core Intel Xeon Processor 7300 Series
- 192GB FBDIMM max memory
- Up to 6 SATA or SAS drives
- 1200W redundant power
- Pre-installed VMware ESXi on Disk-on-Module
- 5-Year Warranty

Starting at $5,275

STIRLING 269

2U Dual Xeon Server

- Up to 2x Quad-Core Intel Xeon Processor 5500 Series
- 144GB DDR3 max memory
- Up to 8 SATA or SAS drives
- 720W redundant power
- Pre-installed VMware ESXi on Disk-on-Module
- 5-Year Warranty

Starting at $2,095

STIRLING 444

4U Quad Xeon Server

- Up to 4x Six-Core Intel Xeon Processor 7300 Series
- 192GB FBDIMM max memory
- Up to 5 SATA or SAS drives
- 1200W redundant power
- Pre-installed VMware ESXi on Disk-on-Module
- 5-Year Warranty

Starting at $5,345

Intel, Intel Logo, Intel Inside, Intel Inside Logo, Pentium, Xeon, and Xeon Inside are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries. VMware is a registered trademark or trademark of VMware, Inc. in the United States and/or other jurisdictions. For terms and conditions, please see www.aberdeeninc.com/abpoly/abterms.htm. lj031

888-297-7409
www.aberdeeninc.com/lj031

SHAWN POWERS

The Kyle Rankin Issue (He Wishes!)

Back when we were creating the editorial calendar for 2009 and decided on "Hack This" as the topic for October, I think our resident Hack & / columnist Kyle Rankin thought that issue was going to be dedicated completely to him. (Either that, or he was a bit worried we'd make him write every single article in the issue.) As it turns out, neither was true. I considered telling him we needed enough content from him this month to fill the magazine, but to be honest, Kyle scares me a bit. So, I saved my joking for this piece.

Just because our Hack Editor didn't write every jot and title this month doesn't mean we won't leave as better hackers. Duilio J. Protti gets us going with low-level system programming, but we don't use an actual system. Thanks to KVM, we learn low-level system hacking on a virtual machine. Virtual machines are so robust nowadays, the concepts and practices will work on a real system too. If software hacking isn't enough for you, fear not. Marco Fioretti shows us some actual hardware hacking with Field Programmable Gate Arrays. You won't need to break out a soldering iron, but it'll just about feel like it!

If all this system hacking sounds fun, but you're worried your proprietary computer BIOS will hinder your skills, Anton Borisov might be just the man you want to listen to. He shows us the ins and outs of coreboot (you may know it as LinuxBIOS). Although flashing the firmware on off-the-shelf routers might be second nature, not too many of us have ventured into BIOS flashing. It seems a perfect fit for this issue.

If you've hacked your computer so that it bends to your every whim, you definitely want it to look cool as well. Sure, Compiz on your desktop will draw oohs and ahhs from the occasional Windows user, but even Linux users will be impressed if you implement Clutter in your programming. Clutter allows you to add rich, GUI interfaces to your applications quickly and easily. Alex Crits-Christoph demonstrates utilizing Clutter's advanced toolkit. In fact, we've got a bunch of programming hacks this month. Reuven M. Lerner talks more about testing with RSpec for Rails, and Dave Taylor shows us some cool (and $RANDOM) Web server tricks.

And, of course, the "Kyle Rankin" issue wouldn't be complete without his monthly Hack & / column. This time, Kyle shows us his secret to fighting spam (and its only slightly more palatable cousin, e-mail ham) preemptively. We all have a handful of "throw-away" e-mail addresses we use when signing up for something online, but Kyle uses a different address every single time. So while Kyle shows us how to keep unwanted things out of our inboxes, his buddy Bill Childers demonstrates how to put things we might want (namely Android) in some bizarre places. A few months back when the editorial team was talking about content for this issue, I challenged him to install Android on his Netbook. He rose to the challenge, and not only got Android running on his Netbook, but also on a Windows Mobile phone!

We have a full lineup of articles on other topics as well. Whether you are looking for desktop security tips from Mick Bauer, new product information from James Gray or a rally cry to take over the Internet from Doc Searls, you're gonna love his issue. Heck, even I get into the hacking act with Google Voice. You might be able to get unlimited minutes on your cell phone thanks to a handy little trick with your free GV phone number. So for this month, we all got to be hacks. You can too. And if you become a hacker, don't worry, we won't tell Kyle. We wouldn't want him to get jealous.■

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

Servers ▶ DIRECT®

MORE PRODUCTS, BETTER SERVICE, GUARANTEED.

GO STRAIGHT TO THE SOURCE!

1.877.727.7887 | www.ServersDirect.com

YOUR HIGH PERFORMANCE COMPUTING HAS ARRIVED.

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

STARTING AT **$899**

ENTRY LEVEL INTELLIGENT SERVER

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

STARTING AT **$959**

APPLICATION SERVER

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

80 PLUS

SDR-S1343-T04

STARTING AT **$1,099**

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

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

80 PLUS

SDR-S2311-T08

STARTING AT **$1,159**

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

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

SDP-IP308-T10

STARTING AT **$1,599**

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

- Intel Pedestal Chassis w/ 750W (1+1) Power Supply
- Supermicro Server Board w/Intel® 5520 Chipsets
- Support up to Dual Intel® 5500 series Xeon® Quad/Dual-Core, with QPI up to 6.4 GT/s
- Support up to 96GB DDR3 1333/ 1066/ 800MHz ECC Reg./unbuffered DIMM
- Option 10x 3.5" Hot-swap SATA Bays
- Intel® 8257EB Dual-port Gigabit Ethernet Controller

SDR-S4313-T24

STARTING AT **$1,899**

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

- Supermicro 4U Rackmount 900W (1+1) Red. Power Supply
- Supermicro Server Board w/ Dual Intel® 5520 Chipsets
- Support up to Dual Intel® 5500 series Xeon® Quad/Dual-Core, with QPI up to 6.4 GT/s
- Support up to 144GB DDR3 1333/ 1066/ 800MHz ECC Reg. DIMM
- 24x 3.5" Hot-swap SATA Drive Bay
- Intel® 82576 Dual-port Gigabit Ethernet Controller

SDR-S3305-T16

STARTING AT **$1,979**

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

- 3U Rackmount Server with 1+1 900W Red. Power Supply
- Supermicro Server Board w/ Dual Intel® 5520 Chipsets
- Support up to Dual Intel® 5500 series Xeon® Quad/Dual-Core, with QPI up to 6.4 GT/s
- Support up to 96GB DDR3 1333/ 1066/ 800MHz ECC Reg.DIMM
- 16x Hot-swap SAS/SATA Drive Bays
- Intel® Dual 82576 Dual-Port Gigabit Ethernet (4 ports)

SDR-C9303-T50

STARTING AT **$4,339**

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

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

10th year

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

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

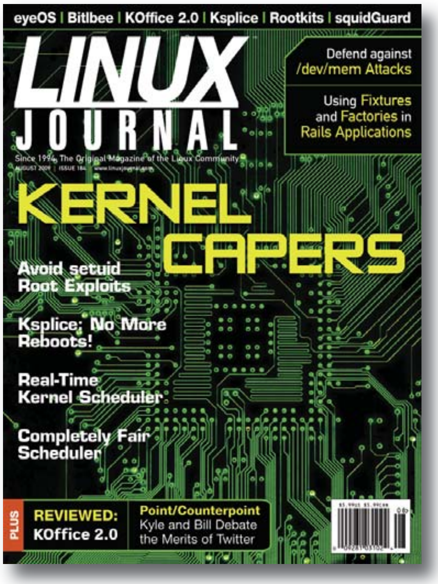
1.877.727.7886 / www.ServersDirect.com

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

intel® Xeon® inside™

Powerful. Intelligent.

letters


eyeOS | Bitlbee | KOffice 2.0 | Ksplice | Rootkits | squidGuard
LINUX JOURNAL
Defend against /dev/mem Attacks
Using Fixtures and Factories in Rails Applications
KERNEL CAPERS
Avoid setuid Root Exploits
Ksplice: No More Reboots!
Real-Time Kernel Scheduler
Completely Fair Scheduler
PLUS REVIEWED: KOffice 2.0
Point/Counterpoint Kyle and Bill Debate the Merits of Twitter

Twitter

I'm with Kyle Rankin on Twitter, and what is more, I have tried it [see Kyle Rankin and Bill Childers' Point/Counterpoint debate on Twitter in the August 2009 issue]. As far as I have found, at least 90%, probably 99%, of content is pointless and uninteresting, even from the few people that I know. Reading the drivel of strangers is even more time-wasting. What I am doing now is usually private and personal—not even for sharing with friends, let alone with, potentially, the whole world.

Several years ago, I gave up all forms of instant messaging. I am not prepared to react instantly to the needs of others. If people want to contact me, they can send me e-mail, and I will reply when I feel like it. There is no need for anything to be instant. I stopped answering the telephone six years ago and have not died from the lack of instant contact. In fact, it is quiet peaceful.

--
Rhian Geleick

I envy your ability to shut out the "on call" structure so many of us are required to live out. That said, Twitter either can be useful or it can be a time suck. The same thing can be said for tons of everyday things we use: cell phones, e-mail, instant messaging, books, food and so on. I'm not suggesting a person would die without Twitter (unlike with food), but I say, if it's useful, use it. If not, don't.—Ed.

Dropbox

I just received my August 2009 *LJ* and noticed that next month, Cross-Platform Development is the topic. I recently came across a cool app that helps share files across cross-platform OSes without a hitch. It is called Dropbox (**www.getdropbox.com**). They have installs for Linux, MS and Apple. I actually can share files in my Virtual Machine with my host, without having to make sure that the file shares are set up properly.

They offer a 2GB account for free subscribers, and you can access your files from their "cloud" from any computer that has Internet access. For larger groups, they have options for paid accounts with larger space provisions.

You guys are still the highlight to my trips to the mailbox each month! Keep it up!

--
soosurfer

You're absolutely right. Dropbox is one of those nifty apps that is indeed cross-platform and largely relies on Internet-hosted services. It's not quite a Web-based app, but leverages the Internet as a means to communicate. We demonstrated the virtues of Dropbox on our Web site as well: **www.linuxjournal.com/video/dropbox-linux**. *It's an awesome tool!—Ed.*

Using banner Instead of motd for Legal Text

In the article "Right Command, Wrong Server" [July 2009], Kyle Rankin mentioned adding a legal warning to the message of the day to be delivered to users after they log in. For legal warnings, I believe the sshd banner option would be more appropriate. The banner is presented to users before a successful login, so people are warned well before they gain access to the system. Edit the /etc/ssh/sshd_config (or distro-specific sshd_config location), and uncomment the following line:

```
Banner /etc/issue.net
```

Then create the text file (/etc/issue.net) that contains your legal warnings to be presented to users before they even attempt to log in. Remember to restart sshd to apply the change.

--
Kevin Benton

Kyle Rankin replies: *Thanks for the tip! IANAL, so I'm not even sure how effective these notices are in a court of law. I recommend checking with a lawyer to see how enforceable these notices are before you do a system-wide deployment.*

Help with MyBook Hack

I am an *LJ* subscriber, and I was so inspired by Federico Lucifredi's article "Hacking Your Portable Linux Server" in the July 2009 issue that I searched eBay until I was able to buy my own WD MyBook WE NAS drive [MDL:WD10000G032-001]. On eBay, it was advertised as WDG1NC10000N World Edition 1TB. I don't know yet if there is any subtle difference between these models. It arrived three days ago, and without trouble, my browser brought up the WD Shared Storage Manager Web interface, reporting firmware version 2.00.15. As an upgrade was available, I immediately upgraded to 2.00.18. I easily followed your article's instructions and used Martin Hinner's script to set up SSH and then log in from a terminal on my PC running Linux Mint. (Shared Storage Manager gave the "update has failed" message that you mentioned, as expected with this latest firmware.) I switched to root, made ssh permanent through inittab, and disabled MioNet (line 19 in my post_network_start.sh) and also verified that my kernel was 2.6.17.14. I logged out, rebooted, logged back in, and yes, ssh was permanent. So far so good. Martin Hinner also provided instructions to install nano, an alternative text editor to vi, so I used wget to obtain the ncurses-5.7 (required by nano) and nano-2.09 tarballs and untarred them. When I tried `./configure` for ncurses-5.7, I hit a wall! It reported “gcc no, cc no, no acceptable cc found in $PATH”.

As I understand it, the problem is (at least) that there is no gcc compiler on this MyBook. I have checked almost every directory, and it’s certainly not in /usr/bin or /usr/sbin or anywhere else that I can see. So my question is, how can I install a gcc ARM9 compiler? I will need to obtain it as a compatible binary and install it, so gcc source doesn’t help me unless I need to first cross-compile on my Linux Mint PC and then transfer from there. This is less important, but it would be interesting to know: when did WD stop including gcc? If I hadn’t upgraded from 2.00.15 to 2.00.18, would the compiler still have been there? Could it be a subtle difference between my WD10000G032-001 and the WDG1NC10000N? Could Martin Hinner’s scripts or anything else I did have deleted gcc? I am stuck and can’t make any progress until I get a working compiler.

Your article mentioned the presence of gcc3.4.2, gmake, wget and so on, and it gave output for `df -h` (same as mine) and `cat /proc/mdstat` (the same except for md4, as mine is a single disk 1TB). Everything seems to check out except the presence of the gcc compiler. Any help with my problem would be so appreciated, and a follow-up MyBook article also would be great.

--
JR

Federico Lucifredi replies: *The WDG1NC10000N is a single-drive “Blue LED” WD Mybook World Edition, and it has hardware matching what is described in the article.*

Your trouble arises from the 2.x firmware. Although the 1.x firmware versions include the full GCC toolchain, I recently upgraded one of my devices and found that GCC is missing in the more recent firmware.

*I ignore the reason for the change (indeed, Western Digital endorsed direct access to Linux in the latest “White LED” devices, where root ssh access can be configured straight off the Web interface), but you have two options for recovery: heading over to the MyBook Community Wiki (***mybookworld.wikidot.com/forum/t-50241/tutorial:how-to-recover-gcc-after-firmware-2-00-15-upgrade***) for recovery instructions would be the first. The alternative is for you to build a cross-compiling toolchain, which is more involved but certainly convenient, as this way you can build your software for beefier machines. If you choose this latter approach, O’Reilly’s* Building Embedded Linux Systems *is your best introduction.*

*I also want to note that ipkg (***mybookworld.wikidot.com/optware***) is on its way to the MyBook. Look for prebuilt packages before you rebuild everything yourself.*

There is certainly more that can be said on the subject. I will consider a follow-up article if the interest is there.

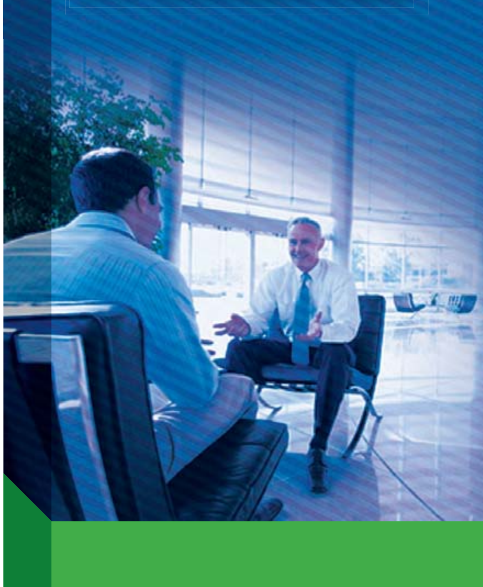
Squid Correction

I’ve been waiting for the last installment of Mick Bauer’s “Building a Secure Squid Web Proxy” series [see the May through August 2009 Paranoid Penguin columns] since it was announced back in the April issue. *Linux Journal* is one of the magazines I read from front to back, and I enjoy it.

I noticed a couple errors in the last article of the series [August 2009]. One is in the instructions to install the blacklists with the --strip argument for the tar command. The code states --strip 1 and the explanation states --strip 2. The other error is in the “Configuring Squid to Use squidGuard” section at the end of the fourth paragraph. The last word should be changed from url_rewrite_program to url_rewrite_children.

One more feature that could be added to this series is how to configure squid as a transparent proxy. I think that a lot of us use Linux or other *nix systems as a firewall to protect home or other networks. When using a transparent proxy, there is no need to configure browsers. There is also protection from other applications that might be accessing banned sites. Keep up the good work.

--
jschiavon



Dracula Correction

Your pipeline of tr and grep has the unintended consequence of eliminating any words that contain, are followed, or are preceded by punctuation [see Dave Taylor's "Looking More Closely at Letter and Word Usage" in the August 2009 issue]. This is not an insignificant number, as it eliminates the last word of every sentence and comma-separated clauses, plus the first and last word of quotations, contractions and hyphenated words.

Using the alternative pipeline, I mentioned a few issues back [see the July 2009 Letters section]:

```
tr '[:upper:]' '[:lower:]' | tr -cs '[:lower:]' '\n'
```

the word count of the *Dracula* text increased by more than 20%.

However, this pipeline results in a different error. Contractions, such as o'clock and didn't, get split into multiple words. The *Dracula* text contains many unusual contractions such as y'are, Ye'll, a'hidin' and gard'ners. Some strange words result when they are split.

--
Jon

Dave Taylor replies: *You're talking about this code snippet:*

```
$ cat dracula.txt | tr ' ' '\
' | grep -v '[^[:alpha:]]' | grep -v "^$"
```

And you're exactly right. I can't believe I didn't notice that! Thanks for the correction, Jon!

PHOTO OF THE MONTH

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

Bringing Tux Along for the Ride—Submitted by Bill Parducci

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**.

diff -u

WHAT'S NEW IN KERNEL DEVELOPMENT

The **state government of New York** is using **git**! The **github.com/nysenatecio** site hosts GPLed tools launched by the New York State Senate, including **open.nysenate.gov/openleg**, which lets users search through ongoing legislative activities. The project seems very kernel-centric, even using a MAINTAINERS file similar to that of the Linux kernel. The data at the back end of these tools also is freely available, so people can write their own front-end sites or contribute to the state's effort. Neat!

The **Xen developers** are tasting the bitter pill of rejection. Many Xen users see a real need for Xen to be accepted into the kernel in the immediate future, but the kernel folks are holding out due to various problems. First, the Xen code is very invasive and apparently is not very respectful of the areas of the kernel it touches. As **Linus Torvalds** puts it, "Xen craps all over other people's code", and elsewhere, he said, "Xen pollutes the architecture code in ways that no other subsystem does." There also are certain features of Xen that people such as **Alan Cox** feel are badly implemented, to the point that they would create long-term maintenance and debugging problems for the kernel, which could not simply be backed out when better solutions arose.

The issues standing in the way of Xen inclusion in the Linux kernel, therefore, are all messy, difficult issues that would require the Xen Project in some ways to come to a screeching halt and move off in a different direction. The Xen developers currently are of a single voice in resisting this, but there's really no alternative. Their best bet probably is to submit small, sensible patches that make the kernel more and more naturally able to support Xen features. Alan Cox, **Theodore Y. Ts'o**, **Ingo Molnar** and others have expressed an interest in helping the Xen developers identify how best to direct their energies toward getting code into the kernel. But, just hoping that one day Linus decides Xen is too important to keep out of the kernel is traditionally a losing strategy.

Once a crucial driver and the topic of endless horrifying flame wars, the **IDE code** is changing hands yet again, from **Bartlomiej Zolnierkiewicz** to **David S. Miller**, who says he will consider IDE to be "legacy code". IDE is, in fact, gradually being obsoleted by the **PATA drivers**, but there still are plenty of users tied to the IDE code, and so bug fixes and other enhancements will continue to be important for some time to come. The **linux-next tree** already has switched from using Bart's tree to David's, maintained at **master.kernel.org:/pub/scm/linux/kernel/git/davem/ide-2.6.git**.

Guo Hongruan announced that he'd successfully gotten Linux compiled and running on the **TriMedia microprocessor**, used typically for audio and video processing. TriMedia has no GCC target, making the port a difficult one. Guo had to run the kernel through a preprocessor to translate all the special GCC extensions into standard C99 source code, after which it compiled under the tmcc compiler. His work is available at **tmlinux.googlecode.com**.

Jon Masters has started a podcast of the **linux-kernel mailing list**, providing semi-daily summaries of events. He was inspired by the desire to force himself to keep up with the mailing list. Audio is available at **kernelpodcast.org**, and there's also an RSS feed available.

—ZACK BROWN

They Said It

I want all my code to be open source, but I will use the best tool for the job, and BitKeeper was the best tool, and at the time the alternatives sucked so bad. When the alternatives are so bad, I will take proprietary code. Proprietary was a downside, but what choice did I have? Hey, I usually do my presentation slides in PowerPoint.

—Linus Torvalds

Looking at the proliferation of personal Web pages on the Net, it looks like very soon everyone on Earth will have 15 megabytes of fame.

—M.G. Sriram

Information on the Internet is subject to the same rules and regulations as conversation at a bar.

—George Lundberg

Any sufficiently advanced bug is indistinguishable from a feature.

—Rich Kulawiec

A people that values its privileges above its principles soon loses both.

—Dwight D. Eisenhower

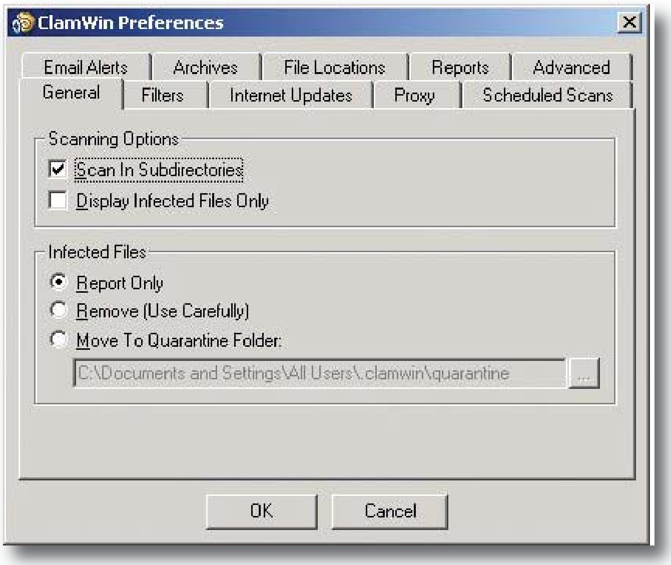
It's always the good men who do the most harm in the world.

—Henry Adams

NON-LINUX FOSS

ClamWin is an open-source antivirus program for Windows. ClamWin provides scheduled virus scans and automatic downloads of its regularly updated virus database. It also integrates with Explorer to provide right-click menu integration for scanning of individual files or directories. And if you're an Outlook user, it provides an add-in for removing infected attachments.

ClamWin is based on the open-source ClamAV engine, an antivirus toolkit for UNIX and UNIX-like systems (as well as Windows systems). ClamAV has built-in support for numerous types of archive files: tar, gzip, bzip2, zip, RAR, Cabinet, CHM and others. It also has built-in support for ELF and PE executables as well as compressed executables using numerous compression schemes. Many popular document formats also are supported: MS Office, Mac Office, HTML, RTF and PDF, among others.


ClamWin Preferences

Scanning Options
☑ Scan In Subdirectories
□ Display Infected Files Only

Infected Files
● Report Only
○ Remove (Use Carefully)
○ Move To Quarantine Folder:
C:\Documents and Settings\All Users\.clamwin\quarantine

ClamWin Preference Pages (from www.clamwin.com)

—MITCH FRAZIER

LinuxJournal.com

A little hack here and a little tweak there, and you have a new LinuxJournal.com.

While working on the new version of the site you all know and love, I have thought about the word "hack" a lot. I think we in the Open Source community in particular have found "hack" to be a useful term with a broad definition. When working with software and platforms that are community supported, there is not always a "right" way to do things or a well-documented process. Thus, we find ourselves making our own recipes and hacks to get the job done whichever way we can. To me, a new way of laying out a Web page may require a workaround that I think of as a theming hack, or I may need to write or modify a Drupal module in order to add functionality to *avoid* hacking, as I always follow the first rule of Drupal, "don't hack core!" Are these code modifications worthy of the word "hack"? I can't break in to a secure network, but I can (usually) find a solution to a Web development conundrum.

Maybe we're all hackers then. As you read through this month's articles focused on hacks, I hope you discover a new way of solving your problems, a workaround that saves you some time, or even just a bit of inspiration that leads you to something new and different. Then, visit LinuxJournal.com to share your hacks with the rest of us!

—KATHERINE DRUCKMAN

LJ Index October 2009

1. Number of Google groups with more than 100,000 members: **33**
2. Number of Google groups that receive more than 10,000 messages per month: **28**
3. Number of Google groups that receive 0 messages per month: **4,331,894**
4. Number of Google groups related to computers: **92,681**
5. Number of Google groups related to computers that receive 0 messages per month: **74,307**
6. Percent of global Internet users that visit google.com (Alexa site rank #1): **32**
7. Percent of global Internet users that visit yahoo.com (Alexa site rank #2): **26**
8. Percent of global Internet users that visit youtube.com (Alexa site rank #3): **19**
9. Percent of global Internet users that visit facebook.com (Alexa site rank #4): **19**
10. Percent of global Internet users that visit live.com (Alexa site rank #5): **16**
11. Number of articles in *Linux Journal*'s article database: **9,116**
12. Number of times the word Linux appears in *LJ*'s article database: **37,671**
13. Number of times the word Windows appears in *LJ*'s article database: **4,956**
14. Number of times the word kernel appears in *LJ*'s article database: **7,796**
15. Number of times the word KDE appears in *LJ*'s article database: **1,367**
16. Number of times the word GNOME appears in *LJ*'s article database: **1,169**
17. Number of hits per month on linuxjournal.com from the Yahoo Web crawler: **335,310**
18. Number of hits per month on linuxjournal.com from the Google Web crawler: **310,270**
19. US National Debt as of 07/05/09, 15:30:12pm MST: **$11,501,541,973,785.60**
20. Change in the debt since last month's column: **$97,726,931,237.70**

Sources: *1–5: Google / 6–10: www.alexa.com*
11: MySQL / 12–16: `fmt -1 | sort | uniq`
17, 18: Drupal / 19: www.brillig.com/debt_clock / 20: Math

Is Telnet Really the Root of All Evil?

FOR QUITE SOME TIME, security experts have been claiming that Telnet is one of the worst protocols out there, and that it should be eradicated from the earth. According to these so-called experts, no one should use anything but SSH. Well, I'm here to show you that they are completely wrong. Telnet is one of the greatest gifts to humans.

The above paragraph should ensure some rabid e-mail messages. For those of you still reading, I actually am serious about Telnet. (Note: I am not talking about telnetd.) The Telnet protocol is made up of two parts: the server portion, telnetd, and the client portion, Telnet. Now, I won't pretend to disagree with the general attitudes toward the server portion, telnetd, and its inherent insecurities. But, I also think we shouldn't throw out the baby with the bathwater.

The client portion, Telnet, is a very useful tool in its own right. Say you're trying to configure your mail client and are running into problems. What options do you have? First, check to see whether the mail POP server is up. How? Why ping it, of course. Oh, but the IT staff drank the ICMP Kool-Aid and decided to block all ICMP traffic. So, now ping won't work. Are you stuck? Of course not—Telnet to the rescue.

How can you use Telnet in this situation? Telnet isn't limited to operating on the standard port. When you run Telnet, you can specify both a server and a port number. So, you can use it to try to connect to your mail POP server and check whether it's up by using:

```
telnet popserver.com 110
```

In this example, you're trying to connect to the POP server in order to read your mail. As Telnet operates over TCP, blocking ICMP packets shouldn't be troubling. Running the above, you should see a response like this (assuming that the mail server is up and running):

```
Trying X.X.X.X...
Connected to popserver.com.
Escape character is '^]'.
+OK InterMail POP3 server ready.
```

Now you can see that the mail server is up, so there must be another issue. This is where Telnet becomes really useful. POP is a text-based protocol, like many other Internet protocols. Because it is just text going back and forth, you can use the Telnet client to talk to the mail server.

For the POP protocol, you simply can log in with your user name and password by using the following commands:

```
USER username
PASS password
-ERR invalid user name or password
```

With the above, you can check that you got the credentials right. As you can see in the example, the password is wrong. Trying again with the correct password gets you into the mail POP server. You can check whether you have mail with this command:

```
LIST
+OK 0 messages
```

You even can use Telnet to send e-mail. If you have access to a mail server, you can connect to port 25 and send e-mail using the following:

```
telnet mailserver.com 25
Trying X.X.X.X...
Connected to mailserver.com
Escape character is '^]'.
220 mailserver.com ESMTP Postfix
MAIL FROM:<user1@host1.com>
250 2.1.0 Ok
RCPT TO:<user2@host2.com>
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
From: user1@host1.com
To: user2@host2.com
Subject: Test

Hello. This is a test of sending mail by telnet.

Joey
.
250 2.0.0 Ok: queued as 1DE8B2830E38
QUIT
221 2.0.0 Bye
Connection closed by foreign host.
```

POP and SMTP are not the only text-based protocols on the Internet. You can use Telnet to check Web sites too. Most Web servers operate on port 80. If you want to check connection issues with a Web site, you can use this:

```
telnet google.com 80
```

And, you'll get a result like this:

```
Trying 74.125.45.100...
Connected to google.com.
Escape character is '^]'.
```

The HTTP protocol also is text-based, so if you want a particular page, use the following command:

```
GET index.html
HTTP/0.9 200 OK
Cache-Control: private, max-age=0
Date: Mon, 06 Jul 2009 18:29:40 GMT
Expires: -1
Content-Type: text/html; charset=ISO-8859-1
Server: gws

<!doctype html><html><head>.........
```

And, you will get the page back from the Web server. Telnet is the smallest and simplest mail client and Web browser, all wrapped into one. Of course, you need to have several Internet protocols memorized, but you really shouldn't be so lazy.

Using this ability, you also can do some basic system probing. With nmap, another popular tool, you can get a list of the open ports on a host machine. Starting there, you can try connecting to each of the ports and seeing what kind of output you receive. I had to do this recently with a remote site where I was having difficulty connecting to the host through SSH. Using Telnet, I was able to see the initial response from the SSH server as the following:

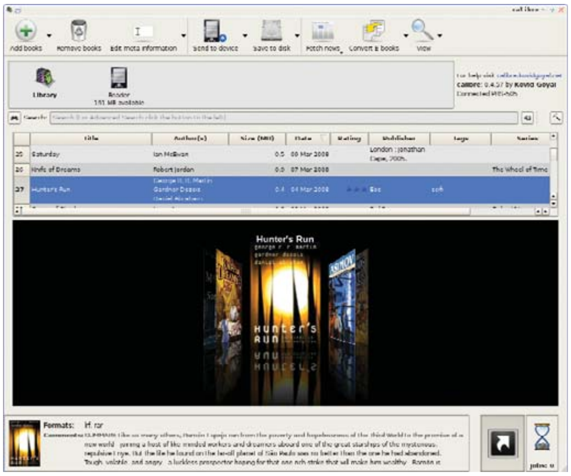
```
telnet host.com 22
Trying X.X.X.X...
Connected to host.com.
Escape character is '^]'.
SSH-1.99-OpenSSH_3.9p1
```

This told me there was an issue with the particular version being run on the server, and I was able to go to the system administrator with an intelligent argument for upgrading the SSH server program.

It's still good to keep Telnet around—at least the client portion. Although the server portion has been considered a security risk in the past, the client portion can serve as a useful tool in network diagnostics. Sometimes old tools are the best, at least for some jobs. Have fun exploring your networks.

—JOEY BERNARD

Calibre—Your eBook Librarian



Calibre in Action (screenshot from calibre.kovidgoyal.net)

As the proud owner of both a Sony PRS-505 and a Kindle DX, it's quickly come to my attention that most eBook software is designed for operating systems other than Linux. With the Kindle, Amazon is trying to remedy that with computer-less converting over e-mail and such, but many of us want more local control. We also want to manage non-DRM books obtained from other vendors. Thankfully, the people behind the open-source Calibre fully support Linux with their Java-based program. Honestly, it's the best eBook management software available for any platform, so as Linux users, we're not sacrificing anything.

Calibre manages to do several things, and do them well:

- It makes a sortable and searchable eBook library.
- It converts eBooks to formats supported by all major eBook readers.
- Books can be uploaded directly to the eBook readers.
- It automatically retrieves metadata and cover art.
- Blogs, newspapers and other on-line information can be downloaded automatically and converted to eBook formats.

With all those features, Calibre still manages to be free and open. If you own an eBook reader or are considering buying one, check out Calibre today: **calibre.kovidgoyal.net**.

—SHAWN POWERS

Move Over Video Professor

If you're a *Linux Journal* reader, chances are you're pretty familiar with the command line. Don't let that stop you from heading over to our Web site and checking out our daily tech tip videos. Heck, I make the majority of them, and I still learn stuff from time to time. Even if you know every tip we throw at you, feel free to deflect new users at us. I must warn you, however, before you know it, those "n00bs" will end up knowing more tips and tricks than you. Don't say we didn't warn you!

—SHAWN POWERS

linuxjournal.com/linux-minute

Google Voice—One Ring(tone) to Rule Them All

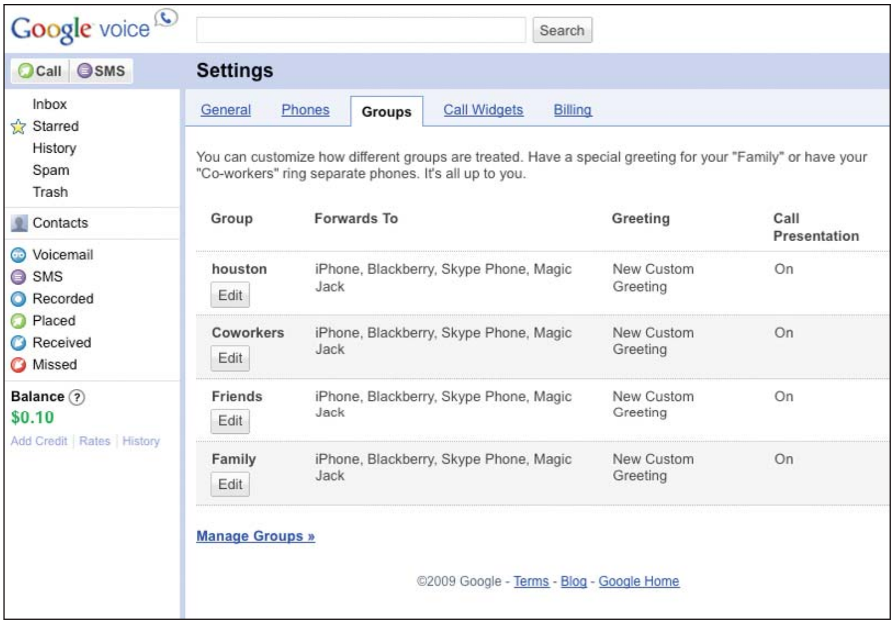
THERE IS A SCENE in the movie *Lawnmower Man* where every phone on the planet rings at the same time. Although Google may be the biggest technical force on the Internet, I don't think it's quite to that point. With the advent of Google Voice (**www.google.com/voice**), however, it's getting closer. The idea behind GV is that a single phone number can be used for all your telecommunications. Because I finally got an invite to the closed beta phase of Google Voice, I can explain a bit about it firsthand.

One Number, Choose Wisely

When (if?) you get an invite to Google Voice, it binds to your Gmail account. You get to pick a phone number from a giant pool of available numbers. The catch is you get to choose only one time. Many area codes are available, as are many exchanges within them. The coolest part about picking a number, however, is Google gives you a tool to alphanumerically choose. Think (123) Coke-Fan might be a nifty number? With the tool, you can test for availability. I like the number I got so much, I'm almost tempted to post it here—almost. Once you pick your number and verify at least one telephone, you get to set up rules. The rules are amazing.

No, You Can't Call My Home Number

Based on Caller ID, Google Voice will route calls however you desire (Figure 1). Contacts are placed into groups, and rules are set up for specific groups. It's possible for certain contacts to ring only certain phones, or you can have custom greetings for some people or even direct some people automatically to voice mail. In fact, with "call presentation", you can listen in to voice mail while it's being left.



Settings

You can customize how different groups are treated. Have a special greeting for your "Family" or have your "Co-workers" ring separate phones. It's all up to you.

Group	Forwards To	Greeting	Call Presentation
houston	iPhone, Blackberry, Skype Phone, Magic Jack	New Custom Greeting	On
Coworkers	iPhone, Blackberry, Skype Phone, Magic Jack	New Custom Greeting	On
Friends	iPhone, Blackberry, Skype Phone, Magic Jack	New Custom Greeting	On
Family	iPhone, Blackberry, Skype Phone, Magic Jack	New Custom Greeting	On

Manage Groups »

©2009 Google - Terms - Blog - Google Home

Figure 1. Contacts are sorted into groups, which can have drastically different call rules.

The Problem of Calling Others

One issue with Google Voice is that if you call people directly with your cell phone (or home phone), they see your actual number instead of your Google Voice number. Thankfully, Google has a couple ways to solve that. If you look up the contact on-line, you can click a call button that calls both you and the person you are trying to reach. You answer your phone to hear it ringing on the other end. Another option is to call your own Google Voice number from one of your phones. You get a voice prompt to make a call, and then you type in the number for the person you are trying to reach. Not only is this a way to have others see your Google Voice number as the incoming call, but if you're lucky enough to have one of those cell-phone plans that allows for a certain number of "free" numbers, you can add your Google Voice number to your "circle" and never pay for minutes again (hopefully, there aren't any cell-phone companies reading this).

Text Messaging and Voice Mailing

One more trick up the sleeve of the big Goog is what I refer to as its "Montana Magic". If you get an SMS text message to your Google Voice number, it automagically is converted to a number in the (406) area code. Thankfully, the first few letters of the message display the contact name, because a random text message from an area code in Montana might be a bit hard to explain otherwise. Once that number is created, however, you can use it to call or text directly to your contact. The number will work only if it's called from one of your phones, but if you call it, the person on the other end sees your Google Voice number instead of your actual number. It is rather complicated, but brilliant as well.

Voice transcription, on the other hand, isn't exactly what I'd call brilliant. I'm absolutely certain it will get better with time. Why am I certain? Because it couldn't possibly get worse. I realize voice mail and voice-mail transcription are both free services, so I'm not really complaining, but the quality of transcription is indescribably horrible.

If you get a chance to get a Google Voice number, or if you were a GrandCentral (**www.grandcentral.com**) user that automatically was enrolled into the Google Voice beta pool, it really is a fun service to play with. I must admit, I'm a bit leery about publishing the number as my single point of contact, because since Google Voice is a free service, it could vanish at any time. The thought of business cards and such being worthless in an instant is a bit disheartening. As a way to shield your cell-phone number and add some phone-spam protection, however, Google Voice is quite a nice tool.

—SHAWN POWERS

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

DO IT WITH DRUPAL

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

STILL GETTING YOUR HEAD AROUND DRUPAL?

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

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

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

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

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

REGISTER NOW
DoItWithDrupal.com

Lullabot
powered

Registration price includes:

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

Early registration and hotel room discounts available.

RSpec

A new way of looking at testing.

REUVEN M. LERNER

Last month, I covered Shoulda, a Ruby gem that allows you to test your code using a method called behavior-driven development. BDD, as it is known, is closely related to test-driven development (TDD), which has become increasingly popular during the past few years, particularly within the Ruby community.

In both BDD and TDD, you start to program by writing a test that the program should pass, if it's working correctly. Of course, because the program hasn't been written yet, the test will fail. You then write the smallest amount of code possible to ensure that the test passes. When that happens, you continue coding by writing another test. The fact that your code is tested completely gives you the confidence and flexibility to "refactor", moving code around and joining it together, without having to worry about introducing new, subtle bugs.

BDD differs from TDD not in its overall method, but rather in its approach and semantics. BDD concentrates on how things look from the outside, rather than from the inside of the code. In the case of a Web application, this often means looking at things from the user's perspective, or if you're a consultant, from the customer's perspective. No longer are you testing the code—instead, you are checking that it meets its specifications. Thus, working with BDD requires that you constantly think of yourself as a consumer of a particular piece of code, and that you consider what it should do at each point, if it is to work correctly. I intentionally use the word *should* here, because as you will see, that is an especially important word in the RSpec vocabulary, and it appears in nearly every test.

RSpec has become quite popular among Ruby programmers in general and Rails programmers in particular. It also is closely tied to several other high-quality testing technologies, such as Cucumber and Celerity, which I will explore in coming months. And, although RSpec is not everyone's cup of tea, it is popular enough that you should expect to encounter it if you do any Ruby development. Moreover, it is often good to try something different, and RSpec definitely is different, providing a new way of looking at testing.

Installing RSpec

The home page for RSpec is **rspec.info**, which contains instructions for installing RSpec, either on its own or as part of a Rails application. I'm looking at a simple Rails application this month as an example, so you need to install both parts.

The first requirement is installing two Ruby gems, both of which are stored on the popular repository for open-source projects, GitHub. You can install these gems with:

```
sudo gem install rspec rspec-rails -V --source
➥http://gems.github.com/
```

(If you already have installed GitHub as a source for gem installations, you don't need to specify it in this command.)

Note that if you have older RSpec-related gems installed, such as rspec_generator or spicycode_rspec_extensions, you probably should remove them from your system. Current versions of RSpec handle these functions for you, and I have encountered problems and conflicts that disappeared when I removed those old gems.

Now that you have RSpec installed, let's create a new, simple Rails project. I often like to use an address book (and appointment calendar) for my examples, so let's create one:

```
rails --database=postgresql appointments
```

Remember, Rails assumes you have three databases for your application, one each for the development, test and production environments. The database parameters are defined in config/database.yml. I assume you are able to set these configuration parameters correctly. Although you don't necessarily need a production database for the purposes of this column, you will need both development and test databases.

Now you must tell the Rails application to include RSpec. There are plugins for RSpec, but I generally prefer to use gems when possible. Modern versions of Rails allow you to include gems in config/environment.rb by adding the following two lines:

```
config.gem "rspec", :lib => false, :version => ">= 1.2.0"
config.gem "rspec-rails", :lib => false, :version => ">= 1.2.0"
```

With the gems in place, you now can put RSpec in place for your Rails application:

```
./script/generate rspec
```

This creates a spec directory (parallel to the test directory, which it effectively replaces). The spec directory contains, by default, three files:

- rcov.opts: setting options for running the Ruby coverage tool rcov when run from within RSpec.
- rspec.opts: setting options for RSpec itself.
- spec_helper.rb: a Ruby file containing global definitions and configurations for the individual specifications, much like test_helper.rb performs in Test::Unit.

With the spec directory in place, you can begin to use the special RSpec generators for models, controllers and scaffolds. For example, you normally would generate a person model with:

```
./script/generate model person first_name:text last_name:text
```

This still will work, but any automatically generated tests will use Test::Unit, installing files into the test directory. By contrast, you can use:

```
./script/generate rspec_model person first_name:text last_name:text
```

This creates the same model file, but also creates a skeleton set of RSpec tests.

Model Testing with RSpec

Let's create a slightly more sophisticated version of the person model:

```
./script/generate rspec_model person first_name:text \
    last_name:text email_address:text phone_number:text \
    sex:text
```

This creates a migration, which you can use to create the first version of your person model:

```
rake db:migrate
```

Now, it's true that you should go into the migration file and modify things, such that (for example) the person's name, e-mail address and sex are all mandatory. However, let's ignore that

Rather, you should imagine the specification that a consumer, or the manager, might want from a “person” object, and then build the object up to adhere to those standards.

step for now and assume that you want all of your validation logic to be at the application layer. In such a case, you would want to put some validations in the model file.

Well, you could do it, but that wouldn’t be very BDD of you, would it? Rather, you should imagine the specification that a consumer, or the manager, might want from a “person” object, and then build the object up to adhere to those standards.

For example, you might want to ensure the presence of the first and last names. So, the first file to modify is spec/models/person_spec.rb, rather than app/models/person.rb. (For reasons I don’t quite understand, Test::Unit calls model tests unit tests, and RSpec calls them model tests, and the controller tests are called functional tests.) If you open that file, you’ll see a new, bare-bones specification:

```
require File.expand_path(File.dirname(__FILE__) + '/../spec_helper')

describe Person do
  before(:each) do
    @valid_attributes = {
      :first_name => "value for first_name",
      :last_name => "value for last_name",
      :email_address => "value for email_address",
      :phone_number => "value for phone_number",
      :sex => "value for sex"
    }
  end

  it "should create a new instance given valid attributes" do
    Person.create!(@valid_attributes)
  end
end
```

You can run your full suite of specs at any time, by typing:

```
rake spec
```

The first line imports anything defined in spec_helper, which I mentioned earlier. Next comes a describe line; this will be familiar to those of you who have looked at Shoulda. The basic idea is that someone reading the specification reads the argument to “describe” and then reads each of the individual specifications that start with “it”. In other words, this spec file tries to say “Person should create a new instance given valid attributes.” And, sure enough, it does.

The before(:each) block tells RSpec what it should invoke before each “it” block. This ensures that the @valid_attributes instance variable will be set to a predictable value before running each spec. You then can modify @valid_attributes as necessary within each spec, as you will soon see.

The thing is, you’re checking the validity of your specification by creating a new instance of Person. You can do that, but if the spec fails, you will end up with a code backtrace mixed in with your report. For this reason, I’m going to change the existing spec definition to look like this:

```
it "should create a new instance given valid attributes" do
  p = Person.new(@valid_attributes)
  p.should be_valid
  p.save.should_not == false
end
```

Instead of Person.create, you now are invoking Person.new, assigning it to the variable p. Let’s check p in two different ways, once using should and the other using should_not. These methods are mixed in by RSpec to the Object class and contain a great deal of behind-the-scenes magic to make specifications readable, almost as if they were in plain English. For example, when you say:

```
p.should be_valid
```

RSpec’s should method looks for a method named valid? for that object and checks that the invocation of this method returns true. This works for any predicate (that is, method that returns true or false). If should or should_not is followed by be_XXX, RSpec turns that into a method call of XXX? on the object instance.

So, you can understand what it means to say:

```
p.save.should_not == false
```

which you equivalently could write in a more positive, optimistic way:

```
p.save.should == true
```

In both cases, you invoke the save method on the object and check that its returned value is true. You might argue that you don’t need to invoke both new and save on your object, but I like to make sure the object is valid in both Ruby and the database. After all, it could be that you told the database to reject null values, but that you allowed it using validations in your ActiveRecord definition.

Now let’s move a bit beyond the defaults to set some limits on attributes. Presumably, you want people in your database to have all of these fields (first name, last name, e-mail address, phone number and sex) defined. If you were developing in a non-TDD/BDD way, you first would set up validations for all of those and then add some tests. But, here you're trying to write tests first, thinking from the "outside" how your object might behave. And indeed, each person should have a first name, a last name, an e-mail address and a telephone number. (Strange as it might seem now, there was once a time when having an e-mail address was not expected.)

So you could, for example, include the following:

```
it "should not be valid without a first name" do
  @valid_attributes.delete[:first_name]
  p = Person.new(@valid_attributes)
  p.should_not be_valid
  p.save.should == false
end
```

In other words, you take @valid_attributes, remove the :first_name key from it and then create a new person with the rest of the name-value pairs from @valid_attributes. This should not work, because everyone needs a first name. But when I run the specs, I get:

```
1)
'Person should not be valid without a first name' FAILED
expected valid? to return false, got true
./spec/models/person_spec.rb:23:

Finished in 0.038731 seconds

2 examples, 1 failure
```

In other words, the specification failed. But that's okay—that's precisely what you want when you're working in BDD fashion. You wrote a test, it failed, and now you can go into the code and modify it, so as to ensure that the test passes. Ensuring that this current test passes is a simple matter of adding a validation to your ActiveRecord model. Instead of being the empty default:

```
class Person < ActiveRecord::Base
end
```

you need to make it:

These methods are mixed in by RSpec to the Object class and contain a great deal of behind-the-scenes magic to make specifications readable, almost as if they were in plain English.

```
class Person < ActiveRecord::Base
    validates_presence_of :first_name
end
```

I save this change, run `rake spec` again, and sure enough, I get:

```
Finished in 0.070752 seconds
2 examples, 0 failures
```

What's next? Now I can move on to the other fields, one by one, in order to test them. And indeed, this back and forth is precisely the way you want to work when you're programming in TDD/BDD fashion. You add a spec indicating what the object *should* do, watch the spec fail and then add the appropriate line or lines for it to work that way.

You can get a bit fancier than merely checking whether attributes exist. RSpec's should method is very powerful, allowing you to check equality (==), numeric comparisons (< and >) and regular expression matches, among other things.

When using RSpec on models, to a large degree, you can rely on the built-in validations that Rails provides. For example, you presumably want the sex field to contain either an M or an F. If someone enters a value other than that, you should not save it to the database. The first step toward such a feature is the introduction of a new spec:

```
it "should forbid characters other than M and F" do
  @valid_attributes[:sex] = 'Z'
  p = Person.new(@valid_attributes)
  p.should_not be_valid
  p.save.should == false
end
```

I run `rake spec`, and find that this test fails. Again, that's to be expected, and now I can modify my Person class such that it is more restrictive:

```
class Person < ActiveRecord::Base
  validates_presence_of :first_name
  validates_inclusion_of :sex, :in => %w(M F), :message =>
  "Sex must be either M or F"
end
```

When I run `rake spec`, I get a failure, but not from this latest spec, which passed just fine, telling me that Z is illegal. Rather, what fails is the first spec, in which @valid_attributes has set the key sex to the value for sex. Once again, that's fine; the fact that I have moved forward in small, incremental steps gives me a chance to identify such issues and fix them, before things get too out of hand. By modifying @valid_attributes such that it uses an M (or an F, if you prefer), the specs work.

Conclusion

RSpec offers a refreshingly different, but still somewhat familiar, approach to issues of testing. By thinking in terms of behavior and specifications, rather than configuration and internals, it becomes easier to create tests. The natural "describe", "it" and "should" terms used in RSpec were chosen carefully, and they help turn testing into a joint venture among all parties, not just programmers.

Although I covered only built-in RSpec matchers (that is, the test that comes after should), it is possible, and even encouraged, to create your own custom matchers for objects in your project.

Next month, I'll continue exploring RSpec by looking at the ways you can test controllers. This raises a number of questions and issues, including those having to do with model objects that are instantiated while inside a controller. As you will see, RSpec's "mock objects" will make this problem much less painful than it otherwise might be.■

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

Resources

The home page for RSpec is **rspec.info**, and it contains installation and configuration documentation, as well as pointers to other documents. The Pragmatic Programmers recently released a book called *The RSpec Book*, written by RSpec maintainer David Chelimsky and many others actively involved in the RSpec community. If you are interested in using RSpec (or its cousin, the BDD tool Cucumber), this book is an excellent starting point. An RSpec mailing list, which is helpful and friendly but fairly high volume, is at **groups.google.com/group/rspec**.

Web Server Tricks with $RANDOM

DAVE TAYLOR

Add some pseudo-randomness to your scripts and user interaction.

I just migrated onto a newer, bigger server (read that as "more expensive", of course, but because my traffic's justifying it, I'm good with the change). To make matters more interesting, I also just bought a new laptop (a MacBook Pro), and between the two migrations, I've been looking through a lot of old directories and bumping into all sorts of scripts I've written in the past few years.

The one I thought would be interesting to explore here is one I wrote for a pal who was involved in a charity and wanted a way to have a single URL bounce people 50/50 to one of two different Web pages—a sort of mini-load balancer, though his application wasn't quite the same.

The core piece of this is the $RANDOM shell variable that's actually kind of magical—each time you reference it, you'll find it's different, even though you aren't actually assigning a new value to it. For example:

```
$ echo $RANDOM
21960
$ echo $RANDOM
19045
$ echo $RANDOM
2368
$ echo $RANDOM
2425
$ echo $RANDOM
10629
```

This violates the core user design principles of the shell and even the very definition of variables (which are supposed to be predictable—if you assign the value 37 to it, it still should have that value 200 lines and 17 references later). Other variables change value based on what you're doing, without you actually assigning it a new value, like $PWD, but because that's the present working directory, if you move around in the filesystem, it's logical that its value would change too.

The RANDOM value, however, is in a category of its own and makes it super easy to add some pseudo-randomness to your scripts and user interaction (whether it's truly random is a far more complicated—mind-numbingly complex—issue. If you're interested, try Googling "determining the randomness of random numbers" to jump down that particular rabbit hole.

In the Bourne Again Shell (bash), RANDOM numbers are within the range of 0..MAXINT (32,767). To chop it down and make it useful, you can simply divide it by the max numeric value you seek.

In other words, if you want a random number between 1..10, for example, use the % "remainder" function with a call to expr:

```
$ expr $RANDOM % 10
7
$ expr $RANDOM % 10
5
$ expr $RANDOM % 10
9
$ expr $RANDOM % 10
6
$ expr $RANDOM % 10
8
```

Boiling this down further, how to choose between two options randomly now should be jumping out of the page at you, dear reader:

```
if [ "$(expr $RANDOM % 2 )" -eq "0" ] ; then
      conditional expression
fi
```

If you wanted to be a purist, you also could write this with the $(( )) math notation, of course, as you'll see a bit later in this column.

That's enough for us to write the shell script I mentioned earlier, the one that randomly switched between two possible pages when invoked:

```
#!/usr/local/bin/bash
url1="http://www.bing.com/"
url2="http://www.google.com/"
if [ "$(expr $RANDOM % 2 )" -eq "0" ] ; then
  echo "Location: $url1"; echo ""
else
  echo "Location: $url2"; echo ""
fi
exit 0
```

The core piece of this is the $RANDOM shell variable that's actually kind of magical—each time you reference it, you'll find it's different, even though you aren't actually assigning a new value to it.

Can you see what this example script does? If you guessed "randomly redirects you to either Google or Bing", you're right! If not, well, what the heck? Go back and read the code again!

Now, let's say my friend said "75% of the time, I really want to take people to URL1. Can you do it, Dave?"

Here's how that might look:

```
if [ "$(expr $RANDOM % 100 )" -lt "75" ] ; then
```

(Or, even more clearly as `% 4 -lt 3`, for that matter.)

If you have more than two choices, you can use a case statement that makes uneven allocation a bit tricky but otherwise is straightforward:

```
case $(( $RANDOM % 4 )) in
  0 ) echo $url1;                ;;
  1 ) echo $url2;                ;;
  2 ) echo $url3;                ;;
  3 ) echo $url4;                ;;
esac
```

Load Balancing with ruptime

With this in mind, we could write an n-way load-balancing script, so that when people come to the home page, they automatically would be bounced to one of the *n* possible servers.

The interesting step actually would be to round-robin them, based on the server load, of course, which could be done by stepping through the data using the ruptime command.

So, given the uptime output of:

```
$ ruptime host1
host1    16:51   up 3+53:17, 3 users, load 0.65 0.68 0.51
```

What we really want is to get a list of hostnames sorted by how busy those systems are, which can be generated by ruptime with the -rl flags, as shown here:

```
$ ruptime -r -l
host1   down   16+08:34
host4   up      10+13:26,   7 users,  load 0.07, 0.39, 1.04
host3   up      14+06:49,   3 users,  load 0.10, 0.38, 0.49
host2   up       1+17:40,   4 users,  load 0.18, 0.13, 0.09
```

As you can see, the first step is to screen out the hosts that aren't actually up at the present moment, then grab the first field (as it's sorted by how busy the system is at the current moment).

One approach to this could be to call ruptime every time a request comes in and just grab the first value. This can be done like so:

```
$ ruptime -rl | grep -v down | head -1 | cut -d\  -f1
host2
```

The trouble is that the systems report uptime information only approximately every minute, and calling ruptime dozens or hundreds of times per second can end up producing a problem—the least-busy system will be swamped. If you get a lot of traffic, that's not going to be a manageable solution.

Here's where we could have our friend $RANDOM step back into the picture. Instead of always simply picking the machine with the lowest load average, let's randomly choose one of the three least-busy systems. The core snippet would look like this:

```
getline="$(( ( $RANDOM % 3 ) + 1 ))"
targethost="$(ruptime -rl | grep -v down |

   sed -n ${getline}p | cut -d\  -f1)"
```

With a bit more code, you could bias it so that, say, 50% of the time it would pick the least-busy system, 33% of the time it would pick the second-least-busy system, and 17% of the time it would pick the third-least-busy system. As time passed and as the load moved around, these systems would keep changing, and you'd achieve a crude but effective load-balancing system.

Knowing how easily you can select one of a number of possible paths randomly in a shell script, what else can you imagine that would be helpful or just fun?■

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

MICK BAUER

Brutally Practical Linux Desktop Security

Navigate hostile networks with impunity!

As I write this month's column, I'm getting ready to attend DEFCON, my all-time favorite information security conference and hacker rave party. I'll catch up with treasured Known Associates, attend cutting-edge technical presentations and drink Sam Adams beer two-fisted at *Hacker Jeopardy* (it's a tough job, but I'm up to it).

And, at some point, I'll engage in two closely related activities: connecting my laptop to the DEFCON WLAN (wireless local-area network) to check e-mail, hoping fervently that I won't do anything dumb enough to expose my passwords or other personal information to the thousands of other mischievous punks connected to the DEFCON WLAN, and I'll have a nervous chuckle or two at the Wall of Sheep, a real-time list of WLAN users who have done something dumb enough to expose their passwords and other personal information to the thousands of mischievous punks on the DEFCON WLAN.

There isn't necessarily that much shame in ending up on the Wall of Sheep. Several years ago it happened to none other than world-renowned security expert Winn Schwartau. I should mention that Winn was a very good sport about it, too—no identity theft, no foul, as they say.

But, that doesn't mean I'm quite ready to put my own reputation on the line without a fight. You can bet that before I board the plane for Las Vegas, I'm going to lock my laptop *down*, and when I'm there, I'm going to take care of myself like I was back home in the hood, on the wrong side of the tracks, after dark, with a pork chop hung around my neck. Nobody's going to pwn Mick at DEFCON this year without busting out some *supernatural* kung fu. (I hope.)

So what, you may ask, does any of this have to do with those of you who never go to DEFCON and generally stick to your friendly local coffee shop wireless hotspots and neighborhood cable-modem LAN segment? Actually, I think that question pretty much answers itself, but I'll spell it out for you: the tips and techniques I use to navigate the DEFCON WLAN safely with my trusty Linux laptop should amply suffice to protect *you* on whatever public, semiprivate or spectacularly hostile networks to which you may find yourself having to connect.

This month's column is about ruthlessly practical Linux desktop security—what to do to harden your system proactively and, even more important, what to avoid doing in order to keep it out of harm's way.

Overview and Generalities

Here's a summary of what I'm about to impart:

1. Keep fully patched.

2. Turn off all unnecessary network listeners or uninstall them altogether.

3. Harden your Web browser.

4. Never do anything important in clear text. Actually, do *nothing* in clear text.

5. Use VPN software for optimal imperviousness.

6. Pay attention to SSL certificate errors.

7. Be careful with Webmail and surf carefully in general.

8. Make backups before you travel.

Some of those things should be extremely familiar to my regular readers, or simple common sense, or both. Patching, for example, is both critically important and blazingly obviously so. Most network attacks begin with a vulnerable piece of software. Minimizing the number of known bugs running on your system is arguably the single-most important thing you can do to secure it.

I'll leave it to you to use the auto-update tools on your Linux distribution of choice, and the same goes for making backups, an equally obvious (though important) piece of advice.

At least equally important is minimizing the number of software applications that accept network connections. If a given application either is turned off or has been uninstalled, it generally doesn't matter whether it's vulnerable or not. (Unless, of course, an attacker can enable a vulnerable application for purposes of privilege escalation, which is one reason you should not only disable but also remove unnecessary

What about Targeted Malware?

I don't want you to come away from this with the notion that malware *never* figures into Linux security or that it never will. In settings where you can't control what software people run or install on their systems or can't fully enforce automated, timely patching, good antivirus software is essential.

And, I worry quite a bit about targeted malware—that is, hostile code that has been custom created to attack a specific organization or individual. That is becoming an increasingly common tool used by organized crime in stealing large quantities of sensitive data (most typically credit-card numbers and identity data) from specific organizations. Often, the worm or virus will be "planted" in the target network by someone with inside access.

Because a given worm, virus or trojan of this type has been "handcrafted" and never has been released against the general public, there's zero likelihood that any antivirus software vendor even will know about it, let alone provide antivirus signatures that can detect it. Mainstream, signature-based antivirus software is, therefore, generally useless against targeted malware. For this and other reasons, targeted malware is very, very difficult to defend against, even with good patching practices.

But, this article isn't about protecting large networks or even about defending yourself from targeted attacks by well-funded attackers. It's about protecting yourself from attacks by more or less random strangers you may encounter on the Internet, at your local coffee shop's wireless LAN and so forth. And in those contexts, I don't worry very much about Linux malware.

applications.) I cover service disabling in depth later in this article.

So far, so obvious. But, what about antivirus software? As a matter of fact, and by the way I'm waiting for someone to convince me otherwise on this, viruses and worms are not a threat I take very seriously on Linux. In all my years using and experimenting with Linux, including in university lab settings and in my own Internet-facing DMZ networks, I never have had a single malware infection on any Linux system I ran or administered.

Is this because there are no Linux worms or viruses, or because Mick is so fabulously elite? No, on both counts. Rather, it's because I've never been lazy about keeping current with patches, and because I've always very stubbornly used plain text for all my e-mail.

Worms exploit vulnerable network applications—no vulnerability (or no app), no worm. E-mail viruses depend on users executing e-mail attachments or on their e-mail software's running scripts embedded in HTML-formatted e-mail—no attachment executing or script running, no infection.

I've also been lucky in this regard because there are few Linux worms and viruses in the wild to begin with. But, even if there were more, I would repeat, *keeping current with patching and using e-mail carefully is more important than running antivirus software*.

Listing 1. Network Listeners (Pre-Hardening)

```
Proto  Recv-Q  Send-Q  Local Address        Foreign Address     State
tcp         0       0  *:swat               *:*                 LISTEN
tcp         0       0  *:ssh                *:*                 LISTEN
tcp         0       0  localhost:ipp        *:*                 LISTEN
tcp         0       0  *:3128               *:*                 LISTEN
udp         0       0  iwazaru:netbios-ns   *:*
udp         0       0  *:netbios-ns         *:*
udp         0       0  iwazaru:netbios-dgm  *:*
udp         0       0  *:netbios-dgm        *:*
udp         0       0  *:49176              *:*
udp         0       0  *:57500              *:*
udp         0       0  *:icpv2              *:*
udp         0       0  *:bootpc             *:*
udp         0       0  *:mdns               *:*
```

Turning Off Network Listeners

So, assuming you're fully patched already—and I assure you I am—let's get busy disabling network listeners. The first step in doing this is to *find* them. If I run the command `netstat --inet -al` on my Ubuntu laptop, I see what is shown in Listing 1.

You can see I'm running the Swat front end for administering Samba services, the Secure Shell dæmon, the Internet Printing Protocol and Squid (whose default port is TCP 3128). Hmm, you'd never guess that I recently wrote articles on Samba and Squid, would you?

Well, those articles are long finished, so right now I don't have any compelling reason to keep *any* of these services running, especially when I travel. I not only need to shut them down, but also disable their startup scripts. I could simply uninstall them, but I might need them later. Still, as a general rule, if you *can* uninstall unnecessary software, you *should*. Doing so via your preferred package manager is simple enough for me to skip describing here.

At the application level, I can use Swat to shut down Samba cleanly. This clobbers the netbios nameservice (netbios-ns) and netbios datagram (netbios-dgm) udp listeners in Listing 1. But, I also need to disable the Samba startup scripts and Swat itself.

Listing 2. Network Listeners (Post-Hardening)

```
Active Internet connections (servers and established)
Proto   Recv-Q    Send-Q   Local Address     Foreign Address    State
tcp          0         0   *:swat            *:*                LISTEN
udp          0         0   *:bootpc          *:*
```

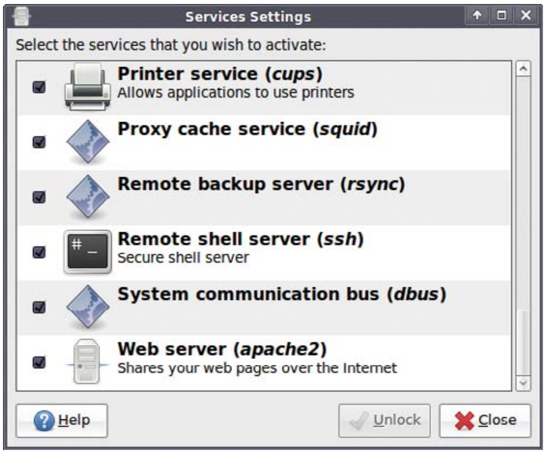
Listing 3. Using lsof to Find Swat's Process

```
bash-$ sudo lsof -i :swat
COMMAND     PID   USER    FD    TYPE   DEVICE SIZE  NODE   NAME
inetd     29534   root    4u    IPv4   521556        TCP   *:swat (LISTEN)
```

Distributions vary in how they handle startup scripts for system dæmons like these. On SUSE, you can use YaST2 or the command `insserv`. On Red Hat, Fedora and CentOS systems, use the command `chkconfig`.

Because my system runs Ubuntu, I can use either the Services Settings applet (Figure 1) in my X Window System's Applications menu or the update-rc.d command. Let's start with the Services Settings applet, which, by the way, is part of GNOME and, therefore, may very well be installed on your non-Ubuntu GNOME desktop too.

Figure 1 shows the Services Settings applet after I've already clicked the Unlock button and provided my root password. Figure 1 also shows the bottom of the list of services running on my system, but that's where some of the juicier items are. I definitely want to uncheck the boxes next to Proxy cache service (squid), Remote shell server (ssh) and Web server (apache2).



Select the services that you wish to activate:

☑ **Printer service (*cups*)**
Allows applications to use printers

☑ **Proxy cache service (*squid*)**

☑ **Remote backup server (*rsync*)**

☑ **Remote shell server (*ssh*)**
Secure shell server

☑ **System communication bus (*dbus*)**

☑ **Web server (*apache2*)**
Shares your web pages over the Internet

Figure 1. GNOME Services Applet

What about Printer service (cups)? I'll disable that too, because at DEFCON, it's highly unlikely I'll need to print anything (or even have the opportunity to). But, note that as Listing 1 shows, my system is listening only for incoming IPP connections on the loopback interface (localhost:ipp). It isn't listening for remote connections to this service.

Me being me, I'll disable it anyhow. A "local" attack vector is local only until some *other* process is hijacked by a remote attacker, at which point the hijacked process might be used to spawn some other process that can attach to the thing having the "local" vulnerability.

Along the same lines—that is, in the interests of generalized paranoia—I'll also disable the following in Services Settings (not shown in Figure 1):

- Account information resolver (winbind).
- Folder sharing service (samba).
- Multicast DNS service discovery (avahi-daemon).
- Network service (xinetd).

Those are all things I'm sure I can live without in an untrusted environment. File sharing in particular, in the form of Samba and its winbind service, is to be avoided in such settings. Now if I re-run my `netstat --inet -al` command, I see only what is shown in Listing 2.

Not bad! Listing 2 shows that by clicking two buttons in the Swat interface and unchecking some boxes in the Services Setting applet, I've clobbered 11 out of the 13 network listeners that previously had been active on my system.

Listing 4. Manually Disabling a Startup Script

```
bash-$ sudo /etc/init.d/openbsd-inetd stop
bash-$ sudo update-rc.d -f openbsd-inetd remove
 Removing any system startup links for /etc/init.d/openbsd-inetd ...
    /etc/rc0.d/K20openbsd-inetd
    /etc/rc1.d/K20openbsd-inetd
    /etc/rc2.d/S20openbsd-inetd
    /etc/rc3.d/S20openbsd-inetd
    /etc/rc4.d/S20openbsd-inetd
    /etc/rc5.d/S20openbsd-inetd
    /etc/rc6.d/K20openbsd-inetd
```

Listing 5. Manually Re-enabling a Startup Script

```
bash-$ sudo update-rc.d openbsd-inetd start 20 2 3 4 5 . stop 20 0 1 6 .
 Adding system startup for /etc/init.d/openbsd-inetd ...
    /etc/rc0.d/K20openbsd-inetd -> ../init.d/openbsd-inetd
    /etc/rc1.d/K20openbsd-inetd -> ../init.d/openbsd-inetd
    /etc/rc6.d/K20openbsd-inetd -> ../init.d/openbsd-inetd
    /etc/rc2.d/S20openbsd-inetd -> ../init.d/openbsd-inetd
    /etc/rc3.d/S20openbsd-inetd -> ../init.d/openbsd-inetd
    /etc/rc4.d/S20openbsd-inetd -> ../init.d/openbsd-inetd
    /etc/rc5.d/S20openbsd-inetd -> ../init.d/openbsd-inetd

bash-$ sudo /etc/init.d/openbsd-inetd start
 * Starting internet superserver inetd
```

But, I'm not done with listeners yet. There still are two left. I can't do much about bootpc, which is part of the dhcp client dæmon that most of us use to configure low-level TCP/IP settings automatically when we connect to a LAN. Even at DEFCON, I'll need dhcpcd (bootpc) active in order to connect to the DEFCON WLAN.

Swat, on the other hand, is fair game to shut down, especially considering I've disabled all the *rest* of Samba. But hold on a second, I've forgotten how! There's neither a Swat entry anywhere in the Services Settings applet nor any applicable script in /etc/init.d. Maybe I can figure out the name of the actual process listening on the Swat port using the lsof (list open files) command, as shown in Listing 3.

Oh, now I remember! Swat is run by inetd, which on Ubuntu systems is part of the package openbsd-inetd. You may remember my disabling xinetd in Services Settings, but openbsd-inetd's startup script has to be disabled manually, the old-school Debian way (Listing 4).

In Listing 4, you can see that I first stopped openbsd-inetd via its startup script and then forcibly removed the various runlevel-links in /etc/rc0.d, etc/rc1.d and so forth, via the update-rc.d command. I can undo all this later, as shown in Listing 5.

Obviously, I will need to make note of the sequence number (in this example, 20 for both the start and kill links) and the runlevels (2–5 for starting and 0, 1 and 6 for killing). As it happens, the settings for openbsd-inetd also are Ubuntu's defaults, so I could use the command `sudo update-rc.d openbsd-inetd defaults` when re-enabling that particular service.

I've spent the bulk of this column shutting down network services. Is that all there is to system hardening?

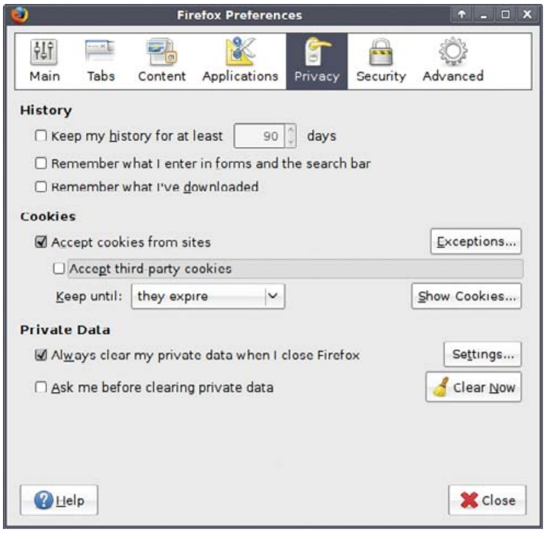
Ordinarily not. If we were talking about a server, we'd have a lot more work to do: configuring individual applications for maximum security, disabling unnecessary user accounts, tightening file permissions, configuring an integrity checker such as tripwire, maybe creating a local iptables firewall script and so forth.

But this is my personal laptop, a single-user system. Shutting down and disabling unnecessary network listeners really is 90% of what I need to do to "harden" it. Most of the rest of what I need to do concerns how I *use* this system. Before I get to that, however, I need to harden one killer application: my Web browser.

Hardening Firefox

Firefox's default security settings are surprisingly okay. Personally, however, I prefer to disable third-party cookies (which admittedly breaks some sites), and sometimes I temporarily turn third-party cookies back on. I also like to turn off my browsing history completely. I don't need to know where I've been, and neither does anybody else. Figure 2 shows these privacy settings.

Under Firefox's Security settings, I certainly want to make sure Firefox's default warnings for add-on installations, suspected forgeries and other suspected hostile sites are intact. I also turn off all password caching—the very idea of allowing my browser to store passwords is, if you ask me, the way of tears. Figure 3 shows these settings.



History

☐ Keep my history for at least 90 days

☐ Remember what I enter in forms and the search bar

☐ Remember what I've downloaded

Cookies

☑ Accept cookies from sites

☐ Accept third party cookies

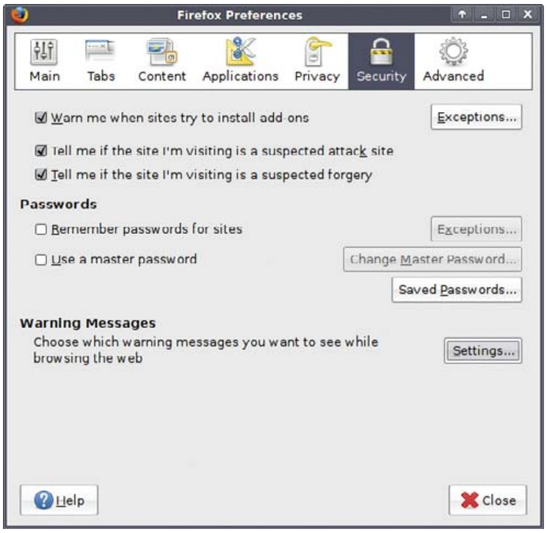
Keep until: they expire

Private Data

☑ Always clear my private data when I close Firefox

☐ Ask me before clearing private data

Figure 2. Firefox Privacy Settings



☑ Warn me when sites try to install add-ons

☑ Tell me if the site I'm visiting is a suspected attack site

☑ Tell me if the site I'm visiting is a suspected forgery

Passwords

☐ Remember passwords for sites

☐ Use a master password

Warning Messages

Choose which warning messages you want to see while browsing the web

Figure 3. Firefox Security Settings

Finally, I should mention a couple useful Firefox add-ons. I swear by Adblock Plus, which enforces a blacklist of known Web advertisement sites whose content is frequently streamed to various other sites. Blocking those sites effectively blocks in-line ads. You can get Adblock Plus by searching for "adblock plus" in Firefox's Get add-ons tool, under Tools→Add-ons.

I realize this breaks various people's Internet revenue streams, but I use Adblock Plus less for aesthetic or performance reasons (ad-blocking certainly shortens Web site load times) than for security. Blocking ads reduces the attack surfaces of the sites you visit and, therefore, reduces your chances of being exposed to spyware or other hostile content.

It may be difficult for a given Web hacker to compromise nytimes.com directly, but it's considerably easier to compromise one or more advertisers whose content is loaded in tandem with http://www.nytimes.com. Personally, I'm less worried about destroying Internet ad revenue than I am about protecting my humble browser.

(Before I forget to mention it, you should minimize the number of unfamiliar sites you visit in the first place when using an untrusted network for the very same reason.)

Finally, the Firefox add-on Ghostery allows you to see what Web bugs (trackers), ad feeds and other hidden scripts are active on each Web site you visit. For most such scripts, Ghostery can tell you from whence it originates and why you should or shouldn't worry about it. You can get Ghostery at **www.ghostery.com**.

Now that Ubuntu and Firefox are hardened for DEFCON use, here are some things I'll do when actually connected to that wicked DEFCON WLAN to minimize my chances of ending up on the Wall of Sheep.

Never Transmit Unencrypted Passwords

Always, always assume somebody can and will eavesdrop on all network traffic. Whether you personally can believe or imagine how they'll do this or not is unimportant—it's the attacker's imagination and skill that matter here, not yours. The only sensible assumption for you to make about the network's integrity is that there isn't any, and that someone can see all traffic going to and from your system. Accordingly, you must not log on to any remote system through any unencrypted protocol.

Telnet, non-anonymous FTP, IMAP, POP3 and any browser-based login involving an http:// URL rather than https://, therefore, are all off limits. In the modern era, all these applications (remote shell, file transfer, e-mail and most Web applications) can and should be used in encrypted implementations, such as SSH, FTPS or SFTP, IMAPS, POP3S and https, at least for logons and other sensitive transactions.

Use VPN

If your home or corporate network supports it, use a strong VPN protocol such as IPsec or SSL-VPN to connect back, and do all your Web surfing and other Internet business via the home network. Yes, this will degrade the performance and speed of your Web-surfing experience; however, it will all but obliterate risks associated with eavesdropping, DNS spoofing, evil twinning and similar attacks (although, of course, if your home or corporate network is targeted further downstream from the hostile LAN you're connected to locally, those downstream attacks *will* apply).

Care about SSL Certificates

When using any public, hostile or otherwise untrusted network, you must pay careful attention to your browser's padlock icon. If there is any problem with any certificate being presented by an SSL-protected site you've had no issues connecting to in the past, you should assume that somebody is attempting a man-in-the-middle, proxy or imposter Web site attack.

Be Careful with Webmail and On-line Banking

Gmail, Yahoo, Windows Live (Hotmail) and on-line banking sites are all particularly likely for someone to attempt to proxy or spoof. If you must visit such a site from a hostile LAN, again, watch for any certificate weirdness.

If you have your own Webmail server or have access to Webmail from a smaller provider, such as a regional ISP, those may be less likely for someone to attempt to spoof or proxy than one of the "big guys". For maximum paranoia though, using a strong VPN connection really is best.

Conclusion

And with that, we're out of space for this month, but we're done! If I say so myself, it wasn't a bad column's work. My laptop is now hardened for DEFCON WLAN use, and you've hopefully learned a thing or two about Mick's brutally pragmatic approach to desktop security. We'll see whether I end up on the Wall of Sheep this year (if so, maybe I'll admit it, and maybe I won't). Good luck with your own public LAN adventures!■

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".

Spam: the Ham Hack

Check out a few simple spam-fighting tweaks to two of my favorite open-source programs: mutt and Postfix.

KYLE RANKIN

When you think about it, all spam really is, is hacked ham. This is true for both meanings of the word. In the food sense, it is composed of hacked-up bits of pork that are reassembled to resemble (sort of) ham. In the e-mail sense, spam is just hacked-up bits of text that are reassembled so they somewhat resemble a legitimate e-mail you want to read (ham).

Countless articles talk about the open-source tools you can use to reduce the amount of spam in your inbox, so I'm not going to write yet another article about SpamAssassin, Razor/Pyzor, Spam Blackhole Lists (SBLs) or even grey-listing, although I recommend looking into those countermeasures if you haven't already done so. Instead, I assume you already have these measures in place, so I'm going to discuss a few extra tools that make spam management a bit more, well, manageable.

Virtual Addresses in Postfix

I'm not a huge fan of Web-based e-mail, although for the longest time, I did think it was a great tool for spam-catching. I would set up a free Web e-mail account, and whenever I bought something on-line, I used that e-mail address as a contact. Of course, whenever I bought something new, I'd have to go into the account first and purge the mountain of spam that had accumulated since the last time I used the account. The other downside was that I still never knew which companies had sold out my e-mail address and which ones kept it protected.

Virtual addresses in Postfix work much like aliases work in most mail servers.

Since then, I've found an even better solution with virtual addresses in Postfix. Now that I run my own mail server, I can set up as many e-mail addresses as I want for free and have them all land in the same inbox. Not only does this make it easier to find all my on-line receipts later, but also because of the way I set it up, I easily can find out which companies sold me out and block only their e-mail messages.

Virtual addresses in Postfix work much like aliases work in most mail servers. It provides you a way to set up a large series of To addresses that your mail server will accept and map those addresses to one or more real addresses on the server, or even forward e-mail to addresses on a completely different server. All you have to do is set up a new database that defines the mapping between virtual and real addresses, and then tell Postfix to use it.

For this example, let's assume I have a mail server that already accepts mail for example.net, and my personal account is kyle@example.net. Whenever I set up a new account, either on a social network or an on-line retailer (anything that could potentially send me spam), I set up a new virtual address named after them and the year. Let's assume I created an account on CompanyX's site, so I could buy a T-shirt and also registered a new profile on TweetBookSpace—the new hip social-networking-meets-cell-phones-meets-LOLcats site.

First, I would create a regular text file called /etc/postfix/virtual that contained the following entries:

```
# System accounts that should exist
kyle@example.net                  kyle@localhost
root@example.net                  kyle@localhost

# Spam-catching accounts
companyx2009@example.net          kyle@localhost
tweetbookspace2009@example.net    kyle@localhost
```

All of the addresses in the left column correspond to addresses for which Postfix will accept mail, and the right column tells Postfix to which real account to forward the mail. Instead of an @localhost address, I also could forward it to some other external e-mail address, or even list multiple addresses separated by commas. Once I set up the file and whenever I make any changes, I need to run the postmap command against it, so that it creates the custom database file Postfix actually will read:

```
$ sudo postmap /etc/postfix/virtual
```

Finally, I just need to add some new lines to my /etc/postfix/main.cf to define what domains I will use for my virtual aliases and tell it to use the file I just created. I added only one domain here, but if you already have

Polywell Linux Solutions

More Choices, Excellent Service, Great Value!

Serving the Industry for More Than 20 Years

Quiet Performance NAS Storage

8TB $1,999

12TB $2,999

24TB $5,999

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

LD-001

Silent Eco Green PC

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

The Best Terminal PC

Fanless Silent Mini-ITX PC

with Solid State Drive DC Power In

ITX-10A Thin-PC $199

ITX-30A with PCI Riser

Nvidia® ION™ PC

ITX-40A with Blu-Ray or DVD

ITX-1000 with WiFi Option

Excellent for Linux Appliance

4U-24Bay 48TB Storage Server

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

4024SS

Mini-1U Server for Data Center ISP

Intel Dual-Core or Quad-Core Processor, Dual GigaLAN
4GB to 8GB RAM, 2 x 500GB RAID HD
Linux Server Starts at $499

1002SC-1U-945GCL2

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

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

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

POLY
POLYWELL

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

NVIDIA, ION, GeForce and combinations thereof are trademarks of NVIDIA Corporation. Other names are for informational purposes only and may be trademarks of their respective owners.

Well, if you didn't already know from my prior columns, I'm a big fan of mutt, and I didn't want to be left out of all these fancy spam-managing techniques.

multiple domains defined in your mydestination line, move as many as you want managed by this file over to the virtual_alias_domains setting:

```
virtual_alias_domains = example.net
virtual_alias_maps = hash:/etc/postfix/virtual
```

Then, I can run `sudo postfix reload` to reload my settings. If I start to notice that I'm getting spam sent to companyx2009@example.net, all I have to do to block that address is comment out that line in /etc/postfix/virtual and run postmap again. Although it's not necessary to add the year to the e-mail address, I've found that helps when I periodically go through my old throwaway e-mail addresses and comment them out—after all, I always can uncomment them the next time I want to order something.

Spam Tips for Mutt Users

I know plenty of people use whiz-bang graphical e-mail programs, and many of them also have fancy buttons and icons that flash when e-mail might be spam. Well, if you didn't already know from my prior columns, I'm a big fan of mutt, and I didn't want to be left out of all these fancy spam-managing techniques. Once again, mutt's powerful customization comes to the rescue.

Colorize Borderline Spam

Although I do have spam filters set up on my personal account, sometimes messages get through my defenses. It's always a delicate balancing act when you tweak your spam thresholds, so I not only wanted to see how close spam that made it through was to the threshold, but I also wanted to know if any of my legitimate e-mail was close.

I have SpamAssassin configured so that it adds the score to my e-mail headers via the custom X-Spam-Status header. Let's say that my spam threshold was a score of 6; I then set up two rules: one to color any messages with a score of 2 or 3 red and another to color messages with a score of 4 or 5 bright red. That way, both types of messages would stand out—especially the messages right on the tip of my threshold. Here are the folder-hook rules I added to my mutt config:

```
folder-hook . "color index red default '~h
 ➥\"X-Spam-Status:.*score=(2|3)\.\"'"
folder-hook . "color index brightred default '~h
 ➥\"X-Spam-Status:.*score=(4|5)\.\"'"
```

Now, like many people, I have a special spam folder set aside so I can train SpamAssassin. I go in there from time to time to look for any false positives, so I also wanted to highlight any messages that were right above the threshold. The following rule colors any messages that have a score of 6, 7 or 8 magenta:

```
folder-hook . "color index magenta default '~h
 ➥\"X-Spam-Status:.*score=(6|7|8)\.\"'"
```

Quick Macro to Save to the Spam Folder

Now, whenever I go through my inbox and see a message with a suspicious Subject line, if I notice it's colored red or bright red, I might not even bother to open it. Because I know it's close to the threshold, I simply can move it to my spam folder. In mutt, you can do this with just a few keystrokes, but of course, that doesn't stop me from automating it a bit further. After all, why do a few keystrokes when I can bind the S key to save to my spam folder automatically? All I had to do was add the following to my mutt config:

```
# make S automatically save spam to the spam folder
macro index          S "simaps://mail.example.net/INBOX.spam"
macro pager          S "simaps://mail.example.net/INBOX.spam"
```

Of course, change `imaps://mail.example.net/INBOX.spam` so that it points to the spam folder on your IMAP server, but once you do, you either can press S to save an individual message to the spam folder or you can tag all of the spam in your inbox with the T key, and press ;S to save it all to the spam folder at once.

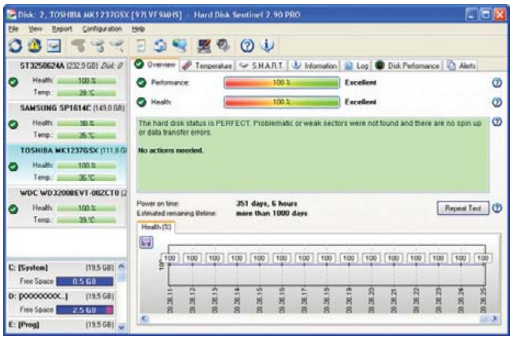
Sure, it would be great if we never had any spam to begin with, but although I can choose what canned food I buy at the grocery store, I may never fully get rid of spam in my inbox. After all, one man's hacked-up pork by-product is another man's tasty canned-ham substitute. If people didn't order those male-enhancement pills, they wouldn't advertise them. At least with a few extra steps, I can make managing spam take less time.■

Kyle Rankin is a Senior Systems Administrator in the San Francisco Bay Area and the author of a number of books, including *Knoppix Hacks* and *Ubuntu Hacks* for O'Reilly Media. He is currently the president of the North Bay Linux Users' Group.

Splunk

Splunk says that version 4 of its IT search application has hit the streets, offering improvements in usability, scalability and performance. Splunk 4 enables users to search, analyze, monitor and report on data from any application, server or network device in real time to troubleshoot outages, investigate security incidents, meet compliance requirements and more—"in minutes instead of hours or days", says the company. Some of the 1,800 enhancements and 50+ new features include 10x faster search and 2x faster indexing, custom dashboards for users of any skill level, more sophisticated enterprise-level management and the Splunk 4 App Framework for creating or leveraging existing apps running on the IT search engine.

www.splunk.com



ST3250624A (232.9 GB) Disk: 0
Health: 100 %
Temp.: 39 °C
SAMSUNG SP1614C (149.0 GB)
Health: 98 %
Temp.: 35 °C
TOSHIBA MK1237GSX (111.8 GB)
Health: 100 %
Temp.: 36 °C
WDC WD3200BEVT-00ZCT0 (2
Health: 100 %
Temp.: 39 °C

C: [System] (19.5 GB)
Free Space 0.5 GB
D: [XXXXXXXX...] (19.5 GB)
Free Space 2.5 GB
E: [Prog] (19.5 GB)

Performance: 100 % Excellent
Health: 100 % Excellent

The hard disk status is PERFECT. Problematic or weak sectors were not found and there are no spin up or data transfer errors.

No actions needed.

Power on time: 351 days, 6 hours
Estimated remaining lifetime: more than 1000 days

H.D.S. Hungary's Hard Disk Sentinel

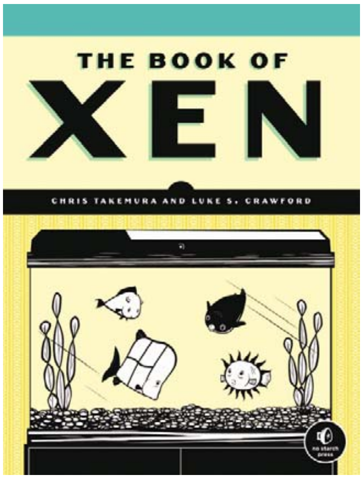
The "ounce of prevention" guys at H.D.S. Hungary have released version 2.9 of Hard Disk Sentinel, a data protection solution that monitors the status of solid-state and hard disks. Hard Disk Sentinel provides detailed disk information, statistics, alerts and backup functions, alerting to present or future disk problems, such as excessive temperature or degradation of disk health, which are signs of imminent hardware failure. The company touts the solution's unique support for a wide range of both internal IDE/SATA/SCSI/SAS and external USB/FireWire/e-Sata hard disks and hard disk enclosures. The new version 2.9 offers deep disk tests to verify hard disk noise, performance and temperature changes. In addition, disk information in RAID arrays connected to 3ware/AMCC and ARECA RAID controllers and solid-state disk features also can be detected. The Enterprise server solution allows monitoring and managing of disk information of remote hosts from a centralized administration console.

www.hdsentinel.com

Kaltura Community Edition

Making the space for on-line video more interesting is Kaltura Community Edition (KCE), which Kaltura dubs "the world's first and only open-source, self-hosted on-line video platform". The freely downloadable KCE allows any site owner or Web developer to integrate customizable video and interactive rich-media functionalities, including video management, publishing, uploading, importing, syndicating, editing, annotating, remixing, sharing and advertising. Kaltura also claims that KCE breaks the "build vs. buy" conundrum and vendor lock-in by allowing publishers and enterprises to build upon and extend an existing robust platform to customize fully their own self-hosted solution on their own servers, behind their own firewalls and at no cost. The company further offers paid support services. KCE runs on Linux, Mac and Windows and is slated to be available on several cloud computing platforms.

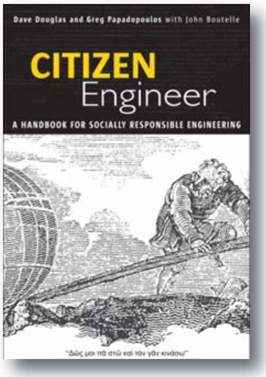
www.kaltura.org



Luke S. Crawford and Chris Takemura's *The Book of Xen* (No Starch Press)

If you are an administrator who has worked with *nix but is new to virtualization, the authorial team of Luke S. Crawford and Chris Takemura has a book for you: *The Book of Xen* from No Starch Press. Xen is a tool that lets administrators run many virtual operating systems on one physical server, including Linux, BSD, OpenSolaris and Microsoft Windows. In the process, users save money on hardware, maintenance and electricity. The book explains everything needed to run Xen, covering installation, networking, virtualized storage, and managing guest and host operating systems. Beyond the basics, it covers profiling and benchmarks, migration, XenSource administration and hardware-assisted virtualization.

www.nostarch.com


Dave Douglas and Greg Papadopoulos with John Boutelle
CITIZEN Engineer
A HANDBOOK FOR SOCIALLY RESPONSIBLE ENGINEERING

David Douglas and Greg Papadopoulos' *Citizen Engineer* (Prentice-Hall)

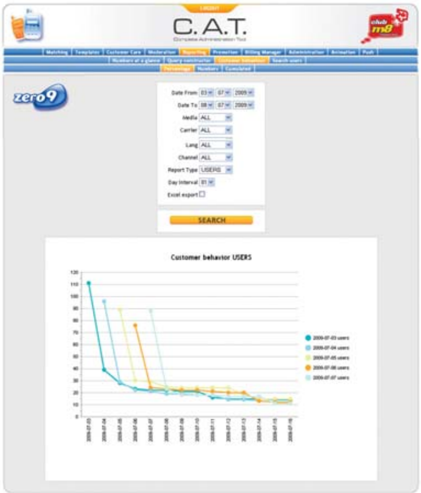
Although the engineering discipline has done many wonderful things for civilization, it has at times been blind to important social and environmental considerations. In order to foster more humane disciplines of engineering, the team of David Douglas and Greg Papadopoulos penned the new book *Citizen Engineer: A Handbook for Socially Responsible Engineering* (Prentice-Hall). *Citizen Engineer* helps engineers of all types to see the full impact of their work beyond design to include ecological, intellectual property, business and sociological perspectives.

www.informit.com

Jedox's ODBO Driver for Palo

Led by the kick-butt motto "Excel without the hell", the company Jedox has announced "the industry's first free ODBO [OLE DB for OLAP] driver" as a part of its open-source OLAP product, Palo. Jedox states that the new ODBO connectivity allows users to carry out advanced OLAP-based Pivot table queries in Excel without the need for expensive licenses for Microsoft SQL Server Analysis Services. Although Pivot tables in Excel are read-only, Palo users have the option to write back values from Excel directly to Palo's OLAP cubes. The company calls "Excel plus Palo" a solution with all the advantages of a centralized BI solution without the cost and time.

www.jedox.com



Date From 03 07 2009
Date To 08 07 2009
Media ALL
Carrier ALL
Lang ALL
Channel ALL
Report Type USERS
Day Interval 01
Excel export □

Zero9 Chat Engine

In the pursuit of bringing us closer together comes the new Zero9 Chat Engine, a product that enables mobile VAS and telco providers to run image- and video-rich chat/dating services via the Web, WAP and SMS. Users can stay in touch with friends via their Web browsers, browsing a WAP site or texting with their cell phones. The engine's core is Zero9's Matching Algorithm, which proposes the ideal and best-matched friends. A back-office suite controls elements, such as CRM, a matching tuner and advanced reporting. The engine is based on the LAMP platform and the Zend framework.

www.zero9group.com

Corsair's Extreme Series Solid-State Drives

The latest offering from Corsair is its Extreme Series X32, X64 and X128 high-performance solid-state drives in 32GB, 64GB and 128GB densities, respectively. The firm says that the drives offer the highest performance currently available on the market, with read speeds of up to 240MB/s and write speeds of up to 170MB/s. Each drive in the Extreme Series utilizes the Indilinx Barefoot controller, Samsung MLC NAND Flash memory and 64MB of onboard cache. Intended uses are as primary drives in desktop and notebooks systems, as well as RAID 0 configurations in high-performance desktops for enthusiasts who want extreme performance.

www.corsair.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

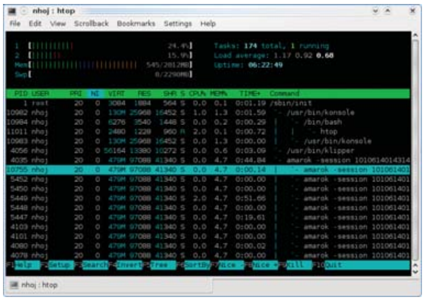
htop—Improved, Intuitive Version of top

htop.sourceforge.net

htop aims to be a better version of the long-lived UNIX program, top. Allowing you to browse through your system processes with the keyboard, htop is a much more intuitive way of doing things than the traditional and archaic UNIX way. The htop Web site gives the best summary with this comparison of htop and top:

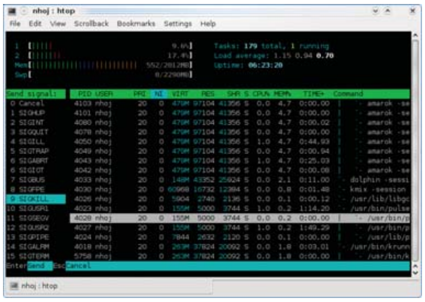
- In htop, you can scroll the list vertically and horizontally to see all processes and complete command lines.
- In top, you are subject to a delay for each unassigned key you press (especially annoying when multikey escape sequences are triggered by accident).
- htop starts faster (top seems to collect data for a while before displaying anything).
- In htop, you don't need to type the process number to kill a process; in top, you do.
- In htop, you don't need to type the process number or the priority value to renice a process; in top, you do.
- htop supports mouse operation; top doesn't.
- top is older, hence, more used and tested.

Installation If you're chasing binaries, packages are provided in either native or third-party form for GoboLinux, Debian, Fedora, Red Hat, Slackware, Gentoo, ALT Linux, OpenSUSE, Mandriva, KateOS and Zenwalk. For those going with source, head to the Downloads/SVN page, grab the latest tarball and extract it, or SVN the nightly development tree. In terms of requirements for compiling the source, the only oddity I ran into was that I needed to install the development files for ncurses (in my case, libncurses5-dev).



```
  1  [||||||||||                  24.4%]     Tasks: 174 total, 1 running
  2  [||||||                      15.9%]     Load average: 1.17 0.92 0.68
  Mem[||||||||||||||||||||||| 545/2012MB]     Uptime: 06:22:49
  Swp[                           0/2290MB]

  PID USER     PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
    1 root      20   0  3084  1884   564 S  0.0  0.1  0:01.19 /sbin/init
10982 nhoj      20   0  130M 25968 16452 S  1.0  1.3  0:01.59 `- /usr/bin/konsole
10984 nhoj      20   0  6276  3540  1448 S  0.0  0.2  0:00.29 |  `- /bin/bash
11011 nhoj      20   0  2480  1228   960 R  2.0  0.1  0:00.72 |  |  `- htop
10983 nhoj      20   0  130M 25968 16452 S  0.0  1.3  0:00.00 |  `- /usr/bin/konsole
 4056 nhoj      20   0 56164 13380 10272 S  0.0  0.6  0:03.09 `- /usr/bin/klipper
 4035 nhoj      20   0  479M 97088 41340 S  0.0  4.7  0:44.84 `- amarok -session 1010614014314
10755 nhoj      20   0  479M 97088 41340 S  0.0  4.7  0:00.14 |  `- amarok -session 101061401
 5452 nhoj      20   0  479M 97088 41340 S  0.0  4.7  0:00.00 |  `- amarok -session 101061401
 5450 nhoj      20   0  479M 97088 41340 S  0.0  4.7  0:00.00 |  `- amarok -session 101061401
 5449 nhoj      20   0  479M 97088 41340 S  2.0  4.7  0:51.66 |  `- amarok -session 101061401
 5448 nhoj      20   0  479M 97088 41340 S  0.0  4.7  0:00.00 |  `- amarok -session 101061401
 5447 nhoj      20   0  479M 97088 41340 S  0.0  4.7  0:19.61 |  `- amarok -session 101061401
 4103 nhoj      20   0  479M 97088 41340 S  0.0  4.7  0:00.00 |  `- amarok -session 101061401
 4101 nhoj      20   0  479M 97088 41340 S  0.0  4.7  0:00.00 |  `- amarok -session 101061401
 4080 nhoj      20   0  479M 97088 41340 S  0.0  4.7  0:00.02 |  `- amarok -session 101061401
 4078 nhoj      20   0  479M 97088 41340 S  0.0  4.7  0:00.00 |  `- amarok -session 101061401
F1Help  F2Setup F3SearchF4InvertF5Tree  F6SortByF7Nice -F8Nice +F9Kill  F10Quit
```

htop vastly improves on the old top UNIX tool, with an interface that lets you browse through your processes with the keyboard.



```
  1  [||||                         9.6%]     Tasks: 179 total, 1 running
  2  [||||||                      17.4%]     Load average: 1.15 0.94 0.70
  Mem[||||||||||||||||||||||| 552/2012MB]     Uptime: 06:23:20
  Swp[                           0/2290MB]

Send signal:    PID USER     PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
 0 Cancel      4103 nhoj      20   0  479M 97104 41356 S  0.0  4.7  0:00.00 |  `- amarok -se
 1 SIGHUP      4101 nhoj      20   0  479M 97104 41356 S  0.0  4.7  0:00.00 |  `- amarok -se
 2 SIGINT      4080 nhoj      20   0  479M 97104 41356 S  0.0  4.7  0:00.02 |  `- amarok -se
 3 SIGQUIT     4078 nhoj      20   0  479M 97104 41356 S  0.0  4.7  0:00.00 |  `- amarok -se
 4 SIGILL      4050 nhoj      20   0  479M 97104 41356 S  1.0  4.7  0:44.93 |  `- amarok -se
 5 SIGTRAP     4049 nhoj      20   0  479M 97104 41356 S  0.0  4.7  0:00.94 |  `- amarok -se
 6 SIGABRT     4043 nhoj      20   0  479M 97104 41356 S  1.0  4.7  0:25.03 |  `- amarok -se
 6 SIGIOT      4042 nhoj      20   0  479M 97104 41356 S  0.0  4.7  0:00.08 |  `- amarok -se
 7 SIGBUS      4033 nhoj      20   0  148M 43352 25624 S  0.0  2.1  0:11.00 `- dolphin -sessi
 8 SIGFPE      4030 nhoj      20   0 60968 16732 12384 S  0.0  0.8  0:01.48 `- kmix -session
 9 SIGKILL     4026 nhoj      20   0  5904  2740  2136 S  0.0  0.1  0:00.12 `- /usr/lib/libgc
10 SIGUSR1     4023 nhoj      20   0  155M  5000  3744 S  1.0  0.2  1:14.20 `- /usr/bin/pulse
11 SIGSEGV     4028 nhoj      20   0  155M  5000  3744 S  0.0  0.2  0:00.00 |  `- /usr/bin/p
12 SIGUSR2     4027 nhoj      20   0  155M  5000  3744 S  1.0  0.2  1:49.29 |  `- /usr/bin/p
13 SIGPIPE     4024 nhoj      20   0  7844  2632  2120 S  0.0  0.1  0:00.00 |  `- /usr/lib/p
14 SIGALRM     4018 nhoj      20   0  263M 37824 20092 S  0.0  1.8  0:03.01 `- /usr/bin/krunn
15 SIGTERM     5758 nhoj      20   0  263M 37824 20092 S  0.0  1.8  0:00.00 |  `- /usr/bin/k
EnterSend  EscCancel
```

Being able to choose what kind of signal you want to send to a process adds versatility.

Open a terminal in the new source directory (whether from an extracted tarball or SVN), and enter the usual:

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

Once the compilation finishes, run the program by entering:

```
$ htop
```

> Allowing you to browse through your system processes with the keyboard, htop is a much more intuitive way of doing things than the traditional and archaic UNIX way.

Usage Once inside the htop screen, you'll see things arranged in a way that will make instant sense to those used to some Linux mainstays, such as xosview, Midnight Commander and various system monitors, including, of course, top. At the top of the screen is a selection of handy system information, such as CPU usage, free memory, swap usage and so on. In the center of the screen are all of your system's processes waiting to be browsed, and in classic Norton/Midnight Commander style, the functions of the program are sensibly linked to the function keys of your keyboard, laid out clearly on the bottom of the screen. Take note of that in particular, as you'll be using those functions the most.

To get started, you can browse through your processes with the arrow keys along with Page Up and Page Down. If you want to kill or tinker with a process, press either K or F9. At this point, htop brings up a menu of possible signals to send the program, with SIGTERM being the default choice. Scroll through these if you want to play with more-advanced features, but for most users, just pressing F9 and Enter will do the job fine.

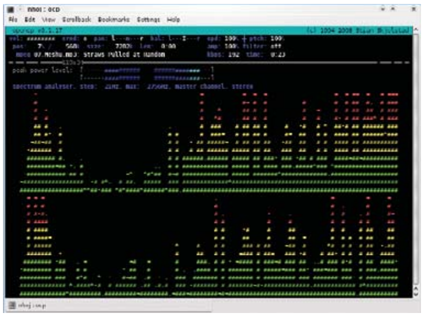
You can kill multiple jobs by pressing the spacebar on each one. The Search function is mapped to F3, allowing you to search for any part of text in the name of the process you're chasing, which comes in handy particularly for overburdened systems with too many processes. The SortBy function, mapped to F6, also is of great help, especially when you want to sort between system- and user-owned processes. And, the Tree function, mapped to F5, is fantastic for seeking things like tricky child processes. This program is fairly loaded with features, so it's worth checking the htop man page and the help screen (assigned to F1) for more information.

Ultimately, htop is a good evolutionary step from unintuitive, older programs like top, and with any luck, it will be included by default with most distros in the years to come. Although the interface still might be a little intimidating for novices, intermediate or advanced users should come to grips with it easily (and let's face it, they're the ones who'll be using it anyway).

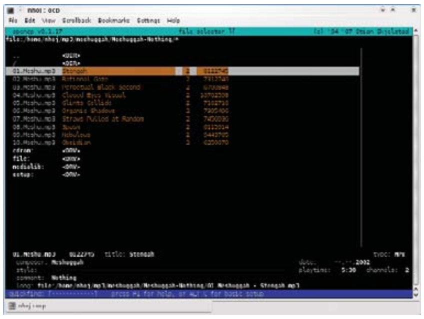
Open Cubic Player

stian.cubic.org/project-ocp.php

Open Cubic Player (OCP) is a text-based audio player that runs in various incarnations on Linux and Windows, and there's even an older DOS version. First appearing in late 1994, the original program was a binary-only freeware version called Cubic Player (running primarily in DOS). It had a reputation for being one of the best module players around, as it supported a great deal of soundcards as well as audio formats. As Windows grew more popular and people demanded GUI-based software, popularity and support for the program died off, as did the project itself.



Open Cubic Player gives wonderful visualizations rendered in real time with pure ASCII.



Thankfully, browsing for audio files is an easy affair, and there are many advanced features that major GUI players don't even have.

Eventually, the source code was opened up to the public in the hope that someone would find it useful, and in late 2003, developer Stian Sebastian Skjelstad started playing around with it, attempting to get the source to compile and run under Linux. After a great deal of tinkering around, Stian eventually got something working, and today, it's available in beta form. And quite frankly, it's a little ripper of a player!

Installation Binary packages are provided for Debian and Ubuntu, as well as some specific information for installation on other systems, but if you're not using the basic .deb packages, you might as well install OCP from source. Grab the latest tarball from the Web site, extract it, and open a terminal in the new folder. As for strange requirements with the source, I had to grab the development files for both ogg and vorbis, which were liboggz1-dev and libvorbis-dev, respectively. Being a wacky console program, you probably need the ncurses development libraries too, but I already had those on my system after compiling htop (see above).

When it comes to compiling the

source, documentation is sorely lacking, but thankfully, compilation is a simple case of the usual:

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

When the compilation is over, run the program by entering:

```
$ ocp
```

Usage Although I'm still coming to grips with the basic controls, playing singular files is a simple affair, as is exploring the program's many functions. When you enter the OCP screen, your first encounter is with the file browser, where you can select your songs, append them to a playlist and so on. I'm not too sure how to operate the playlist functions confidently enough to explain them (you can work out the contents of the manual yourself), but playing a single file is easy. Simply search for the file you want with the up and down arrow keys. Entering directories or playing a file is done by pressing the Enter key.

When a track is playing, this whole project comes to life, and the point becomes clear—you instantly have full visualizations of your music along with neat power-level indicators and all manner of tinkering functions. This is designed for control freaks—seriously. On screen is a load of information, right down to file size, frequency and format information, and so on. However, it's the functions that are the meat of the program. You can alter the panning, balance, speed, pitch, amplification and more. You even can turn on a surround function—not bad for a text-based player!

These functions are mostly spread over the function keys, but the coolest feature (although admittedly a little gimmicky) is actually pausing a song. Press P, and your song winds down and dies like someone has just pulled the plug on an old reel-to-reel player. Unpausing winds it back up to life again. It's really cool and adds genuine charm to the player.

Working your way around this program is initially unintuitive, and the documentation feels as if it's written more for other programmers than new users, but the charm of this program is unavoidable. The beautiful spectrum analyzer patterns rendered in real-time ASCII are enough to bring a tear to any geek's eye, and the advanced controls one expects only of complex, resource-intensive GUI applications will entrench this player firmly in the heart of many a technophile. Awesome stuff—if you can work your way around it!■

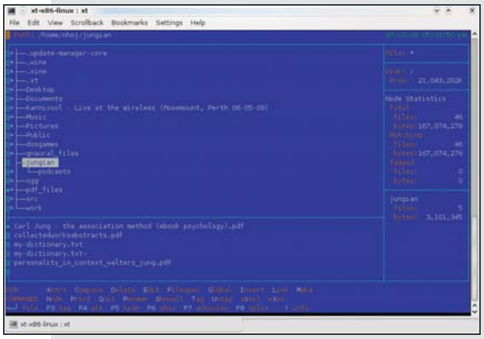
John Knight is a 25-year-old, drumming- and climbing-obsessed maniac from the world's most isolated city—Perth, Western Australia. He can usually be found either buried in an Audacity screen or thrashing a kick-drum beyond recognition.

Projects at a Glance

UnixTree

www.unixtree.org

Much like Midnight Commander captured the feel and essence of the popular DOS application, Norton Commander, UnixTree models itself closely on the once popular XTreeGold. XTreeGold was the introduction that many first PC users had with DOS, which, much like Norton Commander, had a semi-GUI interface to ease the transition into a tricky environment. Although this may not strike an instant chord with command-line purists or full-blown X users, I know a number of people for whom XTreeGold was their primary interface day to day, and hopefully, UnixTree will ease their transition into the UNIX shell in the same way XTreeGold did for DOS. I've had a chance to use it, and I'm quite impressed, especially as certain essential UNIX commands are assigned to single keystrokes to speed up your daily command-line usage.



```
Path: /home/nhoj/jungian
 ├──.update-manager-core                                  File: *
 ├──.wine
 ├──.xine                                                 Disk: /
 ├──.xt                                                    Free:  21,043,292K
 ├──Desktop
 ├──Documents                                             Node Statistics
 ├──Karnivool - Live at the Wireless (Rosemount, Perth 06-05-09)   Total
 ├──Music                                                   Files:          46
 ├──Pictures                                                Bytes:167,074,279
 ├──Public                                                 Matching
 ├──dosgames                                                Files:          46
 ├──gnaural_files                                           Bytes:167,074,279
 ├──jungian                                                Tagged
 │  └──podcasts                                             Files:           0
 ├──ogg                                                     Bytes:           0
 ├──pdf_files
 ├──src                                                    jungian
 └──work                                                    Files:           5
                                                            Bytes:   3,101,345
 Carl Jung - the association method (ebook psychology).pdf
 collectedworksabstracts.pdf
 my-dictionary.txt
 my-dictionary.txt~
 personality_in_context_walters_jung.pdf

DIR       Attrs  Compare  Delete  Edit  Filespec  Global  Invert  List  Make
COMMANDS  Node  Print  Quit  Rename  Showall  Tag  Untag  ^Wall  ^Xec
↵ file  F3 tag  F4 alt  F5 hide  F6 show  F7 autoview  F8 split    ? info
```

UnixTree

Gnake

lightless.org/gnake

Anyone who has a mobile phone will know the time-old classic game, *Snake*. I still play the popular X game *Gnibbles* from time to time, and I've always had a soft spot for the style of gameplay. *Gnake* brings that gameplay to the console in a rather simplified form, but still, it's damn hard. Compilation is easy; simply grab the tarball, enter `make` and then `./gnake`. I'm not sure whether the levels progress (I haven't passed stage one), but some of the options that can be altered include the playground size, speed, number of apples, growing length and the ability to add computer-controlled snakes.

Console Commander

concom.sourceforge.net

Console Commander brings you a selection of information and system tools under an easy-to-navigate group of menus that should be of serious comfort to anyone not familiar with the Linux shell. Some of the clever features group together the sorts of information I've always had to dig through large GUI programs to reach, like Kinfocenter. Information, such as CPU type, free memory, partition usage, distro info and so on, is usually a pain to hunt down individually. Combined with tools that automate tasks like package and repository upgrades, how-tos and many more features, this is a handy program indeed. Although there's nothing technically amazing happening here, it has no pretenses about what it is and will save time for many users. I love it.

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

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

coreboot AT YOUR SERVICE!

Don’t let your PC’s closed-source BIOS stop you from doing what you want with your hardware.

ANTON BORISOV

As people started to build large computing clusters from ordinary PCs, the shortcomings of existing PC BIOSes for certain tasks became more obvious. Like any other computer, on occasion, a cluster’s nodes need to be rebooted; however, most of the original PC BIOSes halted on boot if no keyboard was attached. Obviously, adding a keyboard and monitor to every node in a large cluster is not feasible. These days, this particular problem has been fixed by most PC BIOSes. They contain an option that tells the system to continue booting even if there is no keyboard. Other problems persist, of course, such as how to reboot and adjust the BIOS settings remotely.

One of the first people to try to fix these problems was Ron Minnich from Advanced Computing Lab, LANL, who in 1999 started the open-source BIOS project named LinuxBIOS. In 2008, the project’s name was changed to coreboot.

Development Phases

The project has had three different phases: LinuxBIOS v1, LinuxBIOS v2 (or coreboot v2) and coreboot v3.

The first phase, LinuxBIOS v1, began in 1998–1999, and it became a "finished" product in 2000. At this point in the project, the BIOS consisted of some simple hardware initialization code, and the rest was a stripped-down version of the Linux kernel to do the real initialization. Because the Linux kernel does a lot of hardware initialization during its boot process (tests memory, sets up interrupts and so forth), it seemed like a reasonable choice to use the existing Linux kernel—hence the name LinuxBIOS.

The main problem in LinuxBIOS v1 was writing correct code so that the multitude of available motherboards were initialized properly. The code was far from "universal". Among other things, each motherboard had its own unique memory initialization sequence, and most of the motherboard initialization code was written in assembly.

In the second phase, the developers took a new approach. They left the assembly code to enter protected mode untouched, but they rewrote everything else in C. There was a bit of a problem though. Normally, code generated by a C compiler assumes a stack is available, but because memory has not yet been initialized, there is no stack available. To get around this problem, Eric Biederman created a new C compiler called ROMCC. As you may have guessed, ROMCC generates machine code from C that uses only CPU registers—meaning machine code that needs no stack and, therefore, no initialized RAM! Plus, the CPU's cache is used as RAM. This technique is now known as CAR (Cache-As-RAM).

Although, LinuxBIOS v2 fixed some of the original design's problems, others remained. For instance, in order to add or remove a "payload"—the code that is actually responsible for loading the operating system—you had to recompile LinuxBIOS.

Around 2006, the developers refined their approach again. This, the current phase, is coreboot v3. Coreboot v3 uses the Kconfig facility to set all configuration settings—the same way you recompile a "normal" Linux kernel. The coreboot image is now an archive file that allows modules to be added to and/or removed from an image more easily. Also of note in coreboot v3 is the dropping of ROMCC—all code is compiled with gcc. Due to marketing reasons, the project's name was changed from LinuxBIOS to coreboot.

LinuxBIOS v1 supported 64 motherboards, and LinuxBIOS v2 supported about 120. The current version, coreboot v3, is still young, and at the time of this writing, it supports only 16 different motherboards.

In Practice

My lab contains a VIA EPIA-M II for test purposes. It was manufactured a few years ago, but it's supported by coreboot. Let's take a look at how it is easy to replace its closed-source, proprietary BIOS with the open-sourced coreboot.

Because the EPIA-M II is not yet supported by coreboot v3, I'm going to cover installing v2 for this example. First, make sure you have GCC, binutils, Python, bash, pciutils-devel and subversion installed. Now, check out coreboot v2 code from the repository:

```
$ svn co svn://coreboot.org/repos/trunk/coreboot-v2
```

Next, fetch a payload:

```
$ svn co svn://coreboot.org/filo
```

I decided to use FILO, which is almost the same as LILO, but it uses no BIOS calls. You also may use GRUB2 if you like; it's completely compatible with coreboot.

You also need a special library named libpayload, because

Listing 1. Configuration Process for libpayload

```
*
* Libpayload Configuration
*
* Architecture Options
*
Multiboot header support (MULTIBOOT) [Y / n]
*
* Standard Libraries
*
Enable C library support (LIBC) [Y / n]
Enable tinycurses support (TINYCURSES) [Y / n]
*
* Console Options
*
See output on the serial port console (SERIAL_CONSOLE) [Y / n]
  I/O base for the serial port (SERIAL_IOBASE) [0x3f8]
  Override the serial console baud rate (SERIAL_SET_SPEED) [N / y]
  Use plain ASCII characters for ACS (SERIAL_ACS_FALLBACK) [N / y /?]
See output on a video console (VIDEO_CONSOLE) [Y / n]
  VGA video console driver (VGA_VIDEO_CONSOLE) [Y / n]
  Geode LX video console driver (GEODELX_VIDEO_CONSOLE) [N / y]
Allow input from a PC keyboard (PC_KEYBOARD) [Y / n]
  English (US) keyboard layout (PC_KEYBOARD_LAYOUT_US) [Y / n]
  German keyboard layout (PC_KEYBOARD_LAYOUT_DE) [N / y]
*
* Drivers
*
Support for PCI devices (PCI) [Y / n]
Support for reading / writing NVRAM bytes (NVRAM) [Y / n]
Extended RTC ports are 0x74/0x75 (RTC_PORT_EXTENDED_VIA) [N / y /?]
Support for PC speaker (SPEAKER) [Y / n]
USB Support (USB) [N / y]
```

Listing 2. Configuration Process for FILO

```
*
* FILO Configuration
*
* Interface Options
*
Use GRUB like interface (USE_GRUB) [Y / n /?]
  Command line prompt (PROMPT) [filo]
  GRUB menu.lst filename (MENULST_FILE) [hda3:/boot/grub/menu.lst]
  Timeout for loading menu.lst (MENULST_TIMEOUT) [0]
  Use MD5 passwords in menu.lst? (USE_MD5_PASSWORDS) [Y / n /?]
*
* Drivers
*
IDE DISK support (IDE_DISK) [Y / n /?]
  IDE disk poll delay (IDE_DISK_POLL_DELAY) [0]
  Extra delay for SATA (SLOW_SATA) [N / y /?]
  PCMCIA CF (Epia) support (PCMCIA_CF) [N / y /?]
new USB Stack (USB_NEW_DISK) [Y / n /?]
USB Stack (obsolete?) (USB_DISK) [N / y /?]
NAND Flash support (FLASH_DISK) [N / y /?]
PCI support (SUPPORT_PCI) [Y / n]
  Scan all PCI busses (PCI_BRUTE_SCAN) [N / y /?]
  Sound Support (SUPPORT_SOUND) [N / y]
*
* Filesystems
*
EXT2 filesystem (FSYS_EXT2FS) [Y / n]
FAT (MSDOS) filesystem (FSYS_FAT) [Y / n]
JFS (FSYS_JFS) [N / y]
Minix filesystem (FSYS_MINIX) [N / y]
ReiserFS (FSYS_REISERFS) [Y / n]
XFS (FSYS_XFS) [N / y]
ISO9660 filesystem (FSYS_ISO9660) [Y / n]
  El Torito bootable CDROMs (ELTORITO) [Y / n /?]
Compressed RAM filesystem (CRAMFS) (FSYS_CRAMFS) [N / y]
Squash filesystem (FSYS_SQUASHFS) [N / y]
*
* Loaders
*
Standard Linux Loader (LINUX_LOADER) [Y / n /?]
Windows CE Loader (WINCE_LOADER) [N / y /?]
Artec Loader (ARTEC_BOOT) [N / y /?]
*
* Debugging & Experimental
*
Enable experimental features (EXPERIMENTAL) [N / y /?]
DEBUG_ALL (DEBUG_ALL) [N / y]
DEBUG_ELFBOOT (DEBUG_ELFBOOT) [N / y]
DEBUG_ELFNOTE (DEBUG_ELFNOTE) [N / y]
DEBUG_SEGMENT (DEBUG_SEGMENT) [N / y]
DEBUG_SYS_INFO (DEBUG_SYS_INFO) [N / y]
DEBUG_BLOCKDEV (DEBUG_BLOCKDEV) [N / y]
DEBUG_VFS (DEBUG_VFS) [N / y]
DEBUG_FSYS_EXT2FS (DEBUG_FSYS_EXT2FS) [N / y]
DEBUG_PCI (DEBUG_PCI) [N / y]
DEBUG_LINUXLOAD (DEBUG_LINUXLOAD) [N / y]
DEBUG_IDE (DEBUG_IDE) [N / y]
DEBUG_ELTORITO (DEBUG_ELTORITO) [N / y]
Developer Tools (DEVELOPER_TOOLS) [Y / n /?]
```

FILO depends on it. Check it out, and then run `make`, which first will run through the configuration:

```
$ svn co svn://coreboot.org/repos/trunk/payloads/libpayload
$ cd libpayload
$ make
```

Listing 1 shows the output from the configuration process. Simply press Enter for all options. The value chosen is the default, which is the capitalized value in square brackets [...] if it's a yes/no option; otherwise, it's the value in brackets.

Once the configuration parameters are set, run make again to compile the library:

```
$ cd ../filo
$ make
```

Now you can set the options for FILO. Again, simply press Enter for all the prompts and accept the defaults.

Obviously, not all of the above options actually are needed. For instance, you don't need XFS, JFS or Minix support if your system boots off an ext2/ext3 partition. Once you've gotten everything running, you can come back and switch off the options you don't need, which will reduce the size of the coreboot image.

Notice the following line near the top of Listing 2:

```
GRUB menu.lst filename (MENULST_FILE) [hda3:/boot/grub/menu.lst]
```

My test EPIA-M II system has OpenSUSE 11.0 installed and uses the GRUB bootloader. I chose to include GRUB's interface

Listing 3. If your distribution works with LILO, you can switch off the GRUB interface in FILO.

```
*
* FILO Configuration
*
* Interface Options
*
Use GRUB like interface (USE_GRUB) [Y / n /?]  n
  Autoboot a command line after timeout? (USE_AUTOBOOT) [Y / n]
    Kernel filename and parameters (AUTOBOOT_FILE)
        ➥[hda1:/boot/vmlinuz root=/dev/hda3 console=tty0
         ➥console=ttyS0, 115200]
    Time in seconds before booting (AUTOBOOT_DELAY) [2]
```

support inside FILO, and this is the place to specify the location of GRUB's menu file. If you don't plan to use the GRUB interface (for instance, if your Linux distribution uses LILO for booting), you need to specify the correct line to load the kernel and initrd, as shown in Listing 3.

After setting the FILO's configuration parameters, compile FILO by running make again. The compiled loader is placed here: filo/build/filo.elf.

At this point, you've prepared the payload. Now, you need to generate a coreboot image. First, let's take a look at the config file that is used during the coreboot build (Listing 4):

```
$ cd coreboot-v2/targets/via/epia-m
$ vi Config.lb
```

Lines 1 and 2 define the board and board manufacturer that makes the board we're targeting. Lines 3–5 set the logging level. Higher values give you more information, and logging information comes out on a serial (RS-232) port.

Line 6 specifies the size of the Flash (ROM) memory chip on your board.

Line 7 indicates that coreboot may access CMOS memory for getting any parameters—in particular, the boot sequence.

Listing 4. Configuration Parameters for Coreboot v2

```
 1 target epia-m
 2 mainboard via/epia-m

 3 option MAXIMUM_CONSOLE_LOGLEVEL=8
 4 option DEFAULT_CONSOLE_LOGLEVEL=8
 5 option CONFIG_CONSOLE_SERIAL8250=1

 6 option ROM_SIZE=256*1024

 7 option HAVE_OPTION_TABLE=1
 8 option CONFIG_ROM_PAYLOAD=1
 9 option HAVE_FALLBACK_BOOT=1
10 option CONFIG_COMPRESSED_PAYLOAD_NRV2B=1

11 option FALLBACK_SIZE=131072

12 option _RAMBASE=0x00004000

13 romimage "normal"
14 option USE_FALLBACK_IMAGE=0
15 option ROM_IMAGE_SIZE=64*1024
16 option COREBOOT_EXTRA_VERSION=".0-Normal"
17 payload $ (HOME)/filo/build/filo.elf
18 end

19 romimage "fallback"
20 option USE_FALLBACK_IMAGE=1
21 option ROM_IMAGE_SIZE=60*1024
22 option COREBOOT_EXTRA_VERSION=".0-Fallback"
23 payload $ (HOME)/filo/build/filo.elf
24 end

25 buildrom ./coreboot.rom ROM_SIZE "normal" "fallback"
```

Line 8 specifies that the boot image (payload) is located in ROM. In some situations you will want to load the payload via a serial port. For those cases, use this:

```
CONFIG_SERIAL_PAYLOAD=1
```

Line 9 sets the strategy used to start coreboot. For example, if the checksum from CMOS-memory is not valid, instead of loading the "normal" part, coreboot must start the backup part—that is, "fallback".

Line 10 specifies the compression method (NRV2B). Because Flash chip sizes are somewhat limited, you can (or may have to) use a compressed payload. Instead of NRV2B, you can use LZMA—a more-advanced method:

```
CONFIG_COMPRESSED_PAYLOAD_LZMA=1
```

Line 11 specifies the size of the backup (fallback) part: 128kB, half the size of the Flash chip.

Line 12 indicates where exactly in RAM the compressed coreboot will be placed upon power-up.

Lines 13–18 and 19–24 are almost identical except for name and ID. Here you define the "normal" and "fallback" parts. If coreboot can't start the "normal" part for some reason, it will start the reserved, "fallback" part instead.

The last line specifies how the build tool must combine both parts into a single file. See Resources for more information on all of these options.

That's all for the configuration; now compile coreboot for the EPIA-M:

```
$ cd coreboot-v2/
$ ./buildtarget via/epia-m
$ cd via/epia-m/epia-m/
$ make
```

The coreboot image is ready. The next step is writing it into the Flash chip. To do this, you need a special tool, flashrom, which comes with the coreboot sources:

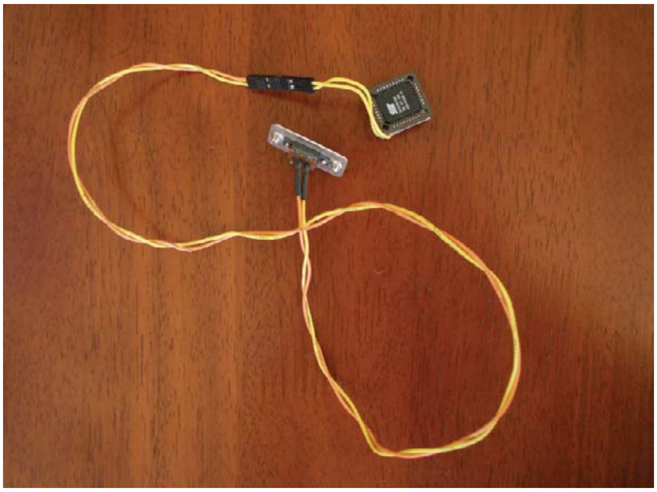


Figure 1. BIOS Savior is a must-have tool.

```
$ cd coreboot-v2/util/flashrom/
$ make
```

Before proceeding, take note, if problems occur when writing to the Flash or if you've configured coreboot improperly (such as forgetting to include a payload), you can brick your hardware. Therefore, it's highly recommended that you have a way to restore your BIOS, such as by using BIOS Savior from IOSS (Figure 1).

To write to the Flash chip, execute the following command:

```
# ./flashrom -w ~/coreboot-v2/targets/via/epia-m/epia-m/coreboot.rom
```

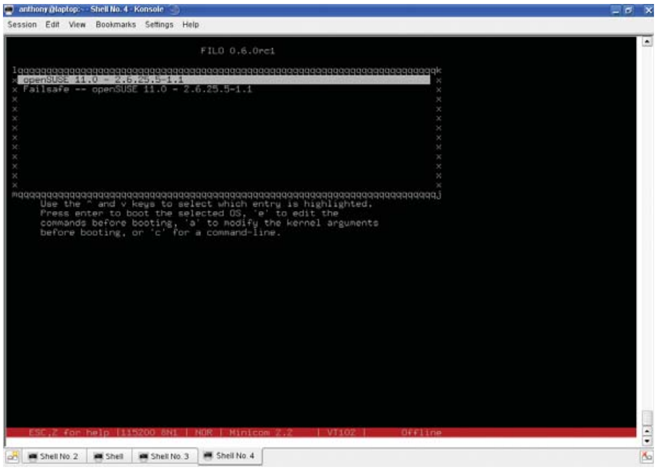
Then, verify that Flash has been written correctly:

```
# ./flashrom -v ~/coreboot-v2/targets/via/epia-m/epia-m/coreboot.rom
```

In order to see boot messages with OpenSUSE 11.0, I first need to modify my GRUB configuration to set the serial line to a speed of 115200 (Listing 5). Now, when I start my EPIA-M, I will be able to see coreboot's output in minicom.

Listing 5. Modifications added to GRUB's menu.lst in order to redirect output to serial port COM1.

```
serial --unit=0 --speed=115200
terminal serial

default 0
timeout 8
gfxmenu (hd0,2)/boot/message

title openSUSE 11.0 - 2.6.25.5-1.1
    root (hd0,2)
    kernel /boot/vmlinuz-2.6.25.5-1.1-default
               ➥root=/dev/sda3 resume=/dev/sda5
               ➥splash=silent showopts vga=0x317
               ➥console=ttyS0,115200n8
    initrd /boot/initrd-2.6.25.5-1.1-default
```


anthony@laptop:~ - Shell No. 4 - Konsole
Session Edit View Bookmarks Settings Help
Shell No. 2
Shell
Shell No. 3
Shell No. 4

```
                          FILO 0.6.0rc1

lqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqk
x openSUSE 11.0 - 2.6.25.5-1.1                                                x
x Failsafe -- openSUSE 11.0 - 2.6.25.5-1.1                                    x
x                                                                             x
x                                                                             x
x                                                                             x
x                                                                             x
x                                                                             x
x                                                                             x
x                                                                             x
x                                                                             x
x                                                                             x
mqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqj
      Use the ^ and v keys to select which entry is highlighted.
      Press enter to boot the selected OS, 'e' to edit the
      commands before booting, 'a' to modify the kernel arguments
      before booting, or 'c' for a command-line.

    ESC,Z for help |115200 8N1 | NOR | Minicom 2.2    | VT102 |      Offline
```

Figure 2. You can control the booting process via minicom.

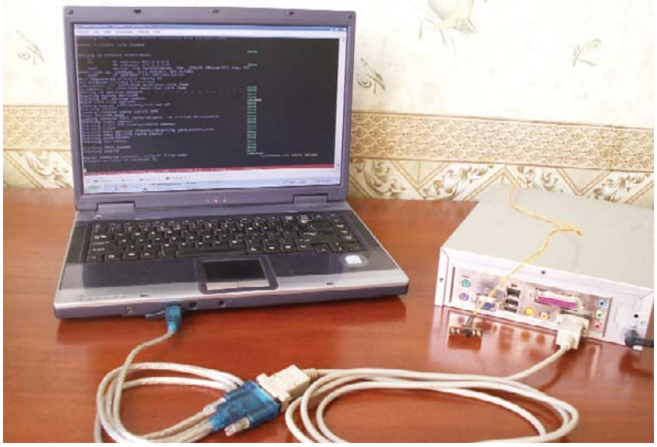


Figure 3. The operating system writes to serial port.

You now should be ready to reboot, so shut down the EPIA-M, connect a null-modem serial cable, and run minicom:

```
# minicom -o -8 ttyUSB
```

Next, restart the EPIA-M, and minicom should show you a GRUB-like boot menu (Figure 2). As the system boots, the operating systems' boot messages also appear in minicom (Figure 3).

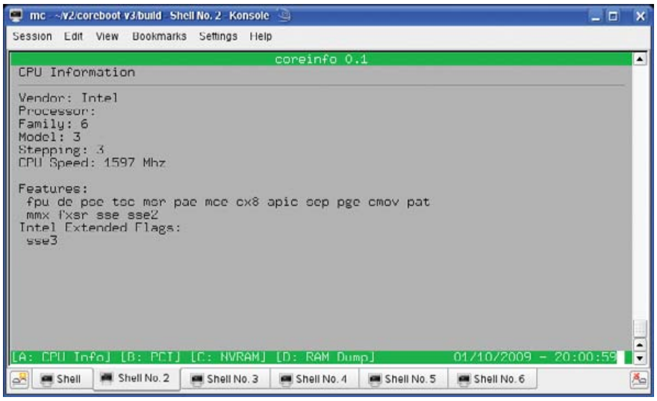
QEMU and Coreboot

Both coreboot v2 and v3 allow you to use the QEMU emulator for doing all the above steps without worrying about bricking your hardware. Using an emulator also is handy when you want to develop a new payload or re-implement some feature of coreboot.

The steps for configuring and compiling coreboot for an emulated system are similar to those for the EPIA-M:

```
$ cd coreboot-v2/
$ ./buildtarget emulation/qemu-x86
$ cd emulation/qemu-x86/qemu-x86/
$ make
```

Before you can run the emulator, you need to to have a copy of a Video ROM/BIOS patched for use with QEMU, such as the one for Cirrus Logic card (**www.coreboot.org/**


mc - ~/v2/coreboot v3/build - Shell No. 2 - Konsole
Session Edit View Bookmarks Settings Help
Shell
Shell No. 2
Shell No. 3
Shell No. 4
Shell No. 5
Shell No. 6

```
                              coreinfo 0.1
CPU Information
──────────────────────────────────────────────────────────────
Vendor: Intel
Processor:
Family: 6
Model: 3
Stepping: 3
CPU Speed: 1597 Mhz

Features:
 fpu de pse tsc msr pae mce cx8 apic sep pge cmov pat
 mmx fxsr sse sse2
Intel Extended Flags:
 sse3

[A: CPU Info] [B: PCI] [C: NVRAM] [D: RAM Dump]          01/10/2009 - 20:00:59
```

Figure 4. QEMU is a powerful tool that greatly helps to develop coreboot.

images/0/0d/Vgabios-cirrus.zip). Download the file, unzip it in the current directory and then run QEMU:

```
$ qemu -L . -hda /dev/zero
```

The latest version of coreboot (v3) doesn't include support for many motherboards yet, but anyone can evaluate coreboot v3 with an emulated system (Figure 4).

Coreboot v3

So, what are the major differences that distinguish coreboot v3 from the previous releases? First, the configuration mechanism is greatly revised. In previous releases, you had to edit configuration files manually. Now, you configure coreboot just as you configure the Linux kernel—using `make menuconfig` or `make xconfig`. Second, the coreboot image itself is nothing but a LAR archive. LAR is a coreboot-specific archiver. It allows you to add, edit and delete payloads in a single step; there's no need to recompile the entire image. Third, the process for producing the code has been simplified and is much more elegant than before. Fourth, the use of ROMCC has been dropped, and all C code now is compiled with gcc. Fifth, there is a growing community and improved documentation on the Web site, as well as feedback from some silicon companies.

So, let's take a look at coreboot v3. Get the source code from the repository and configure it:

```
$ svn co svn://coreboot.org/repos/trunk/coreboot-v3
$ make menuconfig
...
$ make
```

Once this completes, the coreboot image is ready and can

Listing 6. Output of LAR Archiver (Coreboot Image Contents)

```
normal/option_table      (932 bytes @ 0x50);
                         loadaddress 0x0 entry 0x0
normal/initram/segment0  (420 bytes @ 0x450);
                         loadaddress 0x0 entry 0x0x40
normal/stage2/segment0   (194,780 bytes,
                              zeroes compressed to 1 bytes @ 0x650);
                         loadaddress 0x0xd7b0 entry 0x0x2000
normal/stage2/segment1   (34560 bytes,
                              lzma compressed to 18320 bytes @ 0x6b0);
                         loadaddress 0x0x2000 entry 0x0x2000
normal/stage2/segment2   (6076 bytes,
                              lzma compressed to 356 bytes @ 0x4e90);
                         loadaddress 0x0xbff4 entry 0x0x2000
normal/payload/segment0  (183,984 bytes,
                              zeroes compressed to 1 bytes @ 0x5050);
                         loadaddress 0x0x318e0 entry 0x0x19000
normal/payload/segment1  (100,552 bytes,
                              lzma compressed to 78196 bytes @ 0x50b0);
                         loadaddress 0x0x19000 entry 0x0x19000
bootblock                (20480 bytes @ 0x3b000)

Total size = 119314B 116KB (0x1d212)
```

be found in build/coreboot.rom. To view the contents of coreboot.rom, you can use the LAR archiver (Listing 6):

```
$ build/util/lar/lar -l build/coreboot.rom
```

As you can see, coreboot.rom is really just an archive file, so it can be “disassembled” and “reassembled”. Compiling from scratch is not required.

Sophisticated Boot

Most contemporary, proprietary BIOSes contain a BIOS setup program, where you can configure different settings, ranging from RAM parameters to the boot strategy. Currently, there is no such thing available for coreboot, but to illustrate the flexibility of coreboot, let’s examine bayou.

Bayou was developed by AMD and contributed to the coreboot project last year. Bayou is a payload that itself is a container for further payloads, thereby allowing coreboot to choose among payloads at boot time via a menu. For instance, you could include a memtest payload, a FILO payload and even a tint payload (tint is a *Tetris* clone). As an example, let’s build bayou with a tint payload.

To build tint, get the source and patch it:

```
$ wget http://ftp.debian.org/debian/pool/main/t/tint/
➥tint_0.03b.tar.gz
$ tar xfvz tint_0.03b.tar.gz
$ cd tint-0.03b
$ svn export svn://coreboot.org/repos/trunk/payloads/
➥external/tint/libpayload_tint.patch
$ patch -p1 < libpayload_tint.patch
$ make
```

Then, get the bayou payload:

```
$ svn co svn://coreboot.org/repos/trunk/payloads/bayou
$ cd bayou
```

Edit the bayou configuration file (bayou.xml), and add the required payloads (Listing 7).

Listing 7. Configuration for Bayou Payload

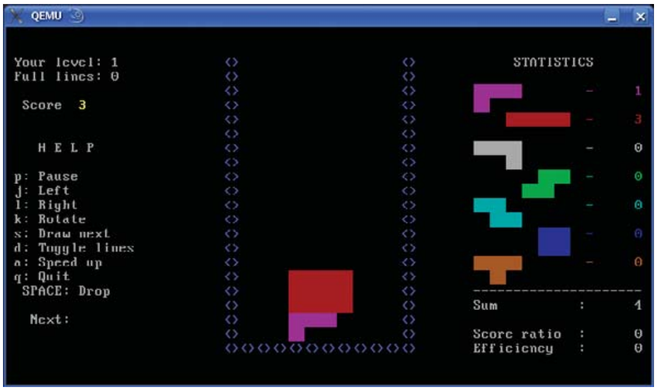
```
<BayouConfig>
    <global>
        <timeout>5</timeout>
    </global>
    <payloads>
        <payload type="chooser" flags="default">
            <file>payloads/filo.elf</file>
        </payload>
        <payload type="chooser">
            <file>payloads/coreinfo.elf</file>
        </payload>
        <payload type="chooser" flags="default">
            <file>payloads/tint.elf</file>
        </payload>
    </payloads>
</BayouConfig>
```

The config file is fairly straightforward. The default payload starts after five seconds, and if nothing is chosen, the default is FILO. The other options are tint or coreinfo (Figure 4 shows it running in QEMU).

Next, make a directory for the payloads, put them into it, and then run make:

```
$ ls -l payloads

-rwxr-xr-x 1 ab users 47004 2009-01-03 11:59 coreinfo.elf
-rwxr-xr-x 1 ab users 71440 2009-01-09 21:35 filo.elf
-rwxr-xr-x 1 ab users 49298 2009-01-10 09:40 tint.elf
-rwxr-xr-x 1 ab users 74334 2009-01-10 19:22 seabios.elf

$ make
```

Now, run image in QEMU, and you can play *Tetris* from the BIOS (Figure 5).



```
Your level: 1                                   STATISTICS
Full lines: 0
                                                      -   1
 Score   3                                            -   3

   H E L P                                            -   0
                                                      -   0
p: Pause                                              -   0
j: Left                                               -   0
l: Right                                              -   0
k: Rotate
s: Draw next
d: Toggle lines
a: Speed up
q: Quit                                         -------------------
 SPACE: Drop                                    Sum          :   4

  Next:                                         Score ratio  :   0
                                                Efficiency   :   0
```

Figure 5. Take a rest—*Tetris* running from the BIOS.

Conclusion

Coreboot allows you to replace your motherboard’s proprietary BIOS with a free and open-source BIOS. Coreboot already can boot Windows XP and Windows Vista as well as FreeBSD, Plan9 and, of course, Linux. Operating systems can be started from local disks, from a network connection or even from a serial port. Although it’s not yet feature-complete, coreboot provides a base for building more flexible BIOSes.■

Anton Borisov lives and works in Russia. Always fond of low-level programming, he has devoted his PhD work to the economic analysis of the advantages and ROI of custom-made firmware.

Resources

Coreboot: **www.coreboot.org**

Coreboot Options: **www.coreboot.org/Coreboot_Options**

IOSS: **www.ioss.com.tw**

QEMU: **bellard.org/qemu**

FPGA PROGRAMMING WITH LINUX

Short of opening your own chip fab, you can't get much closer to the metal than FPGA programming.

Free software licenses and operating systems like GNU/Linux make it possible to learn programming and customize state-of-the-art software in countless ways. Hacking software, however, isn't the last frontier anymore. What if you could hack integrated circuits directly—that is, tell a chip to connect its internal transistors to create exactly the custom, real-time digital hardware you want? This is precisely what you can do with Field Programmable Gate Arrays (FPGAs). In this article, I explain how to do it with nothing else but your Linux computer and an inexpensive development board.

This isn't the first time *Linux Journal* has covered FPGAs (see Resources), but these devices have made huge progress since those articles were written. Plus, prices for powerful development boards have come down dramatically. Today, you can do really cool things with a $200–$300 FPGA-based board, a typical personal computer and one square foot of desk space. Finally, the FPGA community is now big enough and stable enough to make life much easier for beginners. The major FPGA makers offer text or video tutorials and forums where even newbies can find support. Above all, Web sites like Opencores.org codevelop and release Linux-compatible FPGAs and boards like the EUS 100LX with the same spirit (and licenses) as free software. In summary, the barrier to entry is much lower now than it was even just a couple years ago, making FPGA design both a cool hobby and an affordable and interesting addition to the offerings of even high schools.

MARCO FIORETTI

An FPGA Is Not a Microprocessor

A digital integrated circuit (IC) is a chip that deals only with binary digits—meaning signals that can assume only one of two states: 0 or 1, high or low voltage and so on. An FPGA is an IC consisting of one array of digital logic gates. These basic circuits, made up of a few transistors each, instantiate either a flip-flop or lookup tables capable of implementing any boolean function of up to four binary signals. The magic of FPGAs is that the connections among the logic gates (the actual circuit you need) are made at power-up by reading the configuration instructions written into a bit file. Changing the file changes the function of the FPGA.

As flexible as they are, microprocessors are always and only microprocessors: single-purpose hardware that is capable only of executing (relatively slowly) instructions in one machine language. An FPGA, instead, becomes whatever hardware you need. It can morph into a microprocessor, a game console, a real-time IP switch or encryption device, an antitheft server or anything else you can imagine. The only limits are that your circuit cannot require more transistors or external pins than those physically present on the chip, and it can't go faster than the intrinsic propagation delay from gate to gate.

That said, modern FPGAs are powerful enough to let you squeeze several Linux-compatible microprocessors, like the Nios, the PowerPC or the Microblaze, inside them and still have lots of room for your own custom circuits. In many cases, you can load certified CPU designs from libraries and place them on silicon with simple commands, creating very flexible, complete systems inside just one chip.

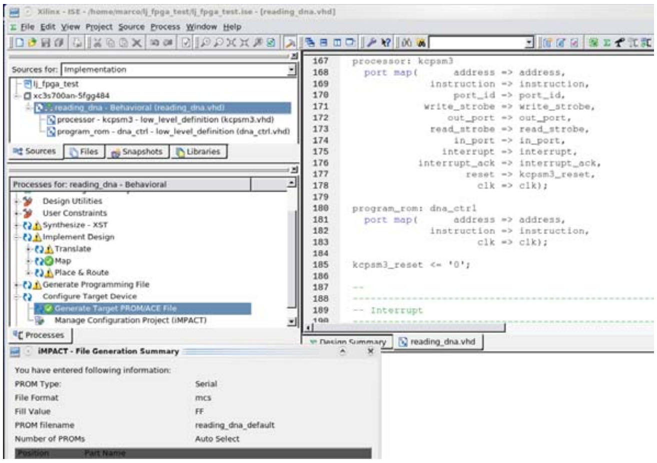
The major manufacturers of FPGAs and other programmable ICs are Altera and Xilinx, followed by Lattice and Atmel. Although the example in this article uses Xilinx products, the general procedure is the same with all vendors, and all of them have similar boards. In all cases, the design software is closed-source and often expensive, but it is possible to download either free trial versions valid for one or two months, or free versions with reduced functionality but free upgrades and no expiration date.

The current way to design FPGAs is to write a behavioral model in a Hardware Description Language (HDL), like Verilog or VHDL, which supports concurrency and synchronous circuits. Concurrency allows you to create fully parallel, independent processes, each describing how to update some variables continuously. Synchronous circuits, instead, are those made of flip-flops that change their state only on the edge of some clock signal.

After the design has been written and verified with an HDL simulator, a compiler creates a list of all the logic gates and the wires (nets) that must connect them to reproduce the functionality of the HDL model. After this logic synthesis, layout programs read the netlist and several constraints files to find out which logic gates inside the FPGAs must be used and which physical, internal wires must connect them to each other. The end result is the bit file that the FPGA reads at power-up.

The official Xilinx design suite is called ISE Foundation (**www.xilinx.com/ise**), and the reduced functionality version is called Webpack. Both programs run on Windows, Red Hat Enterprise and SUSE Linux Enterprise (32- or 64-bit). Other Linux distributions may work too, but there is no guarantee.

ISE has a graphical installer where you must accept the software license and enter the key you got after a free registration on the Xilinx Web site. After it's finished, you'll find a script called settings32.sh or settings64.sh in the installation directory—that's the one you have to source to add the Xilinx software to your path. After this, type `ise` at the prompt to launch the Project Navigator (Figure 1). This is a front end to a bunch of specialized programs, one for each design phase. You also can run most of these back-end utilities from the command line. The Navigator includes a Tcl prompt and, if you select Project→Generate Tcl Script, it will save all the commands you entered through the GUI as a Tcl script.


Xilinx - ISE - /home/marco/lj_fpga_test/lj_fpga_test.ise - [reading_dna.vhd]
File Edit View Project Source Process Window Help
Sources for: Implementation
lj_fpga_test
xc3s700an-5fgg484
reading_dna - Behavioral (reading_dna.vhd)
processor - kcpsm3 - low_level_definition (kcpsm3.vhd)
program_rom - dna_ctrl - low_level_definition (dna_ctrl.vhd)
Sources
Files
Snapshots
Libraries
Processes for: reading_dna - Behavioral
Design Utilities
User Constraints
Synthesize - XST
Implement Design
Translate
Map
Place & Route
Generate Programming File
Configure Target Device
Generate Target PROM/ACE File
Manage Configuration Project (iMPACT)
Processes
Design Summary
reading_dna.vhd
iMPACT - File Generation Summary

```
processor: kcpsm3
  port map(      address => address,
             instruction => instruction,
                 port_id => port_id,
            write_strobe => write_strobe,
                out_port => out_port,
             read_strobe => read_strobe,
                 in_port => in_port,
               interrupt => interrupt,
           interrupt_ack => interrupt_ack,
                   reset => kcpsm3_reset,
                     clk => clk);

program_rom: dna_ctrl
  port map(      address => address,
             instruction => instruction,
                     clk => clk);

kcpsm3_reset <= '0';

--
------------------------------------------------------------
-- Interrupt
------------------------------------------------------------
```

You have entered following information:

PROM Type:	Serial
File Format	mcs
Fill Value	FF
PROM filename	reading_dna_default
Number of PROMs	Auto Select
Position	Part Name

Figure 1. Xilinx Design Suite

Other ISE components, like the simulator, the FPGA Editor and ChipScope, have graphical interfaces. You'd use the FPGA Editor to place and connect single gates manually when the software fails to do it according to your specs. ChipScope is like a software oscilloscope with a USB probe. During synthesis, you can add special circuits to your design that will buffer the internal signals you want

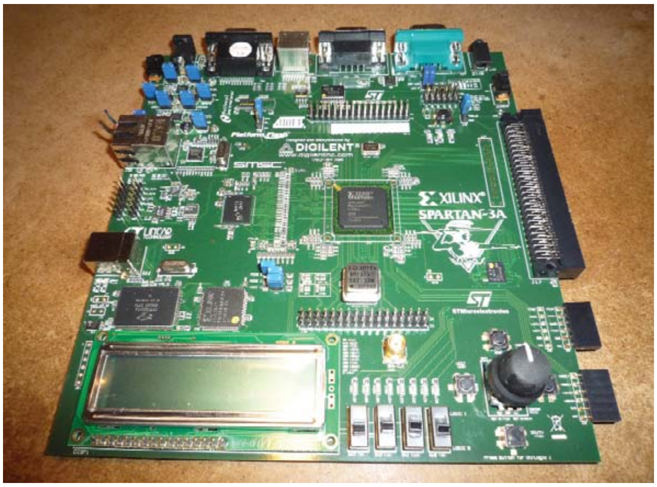


Figure 2. Spartan-3AN Starter Kit

Figure 3. Spartan XC3S700AN FPGA

Figure 4. Stereo Mini-Jack, Serial M, Serial F, Keyboard, VGA, Power (Ethernet/USB on Other Edge)

to see and send them over a USB cable to the ChipScope software for display. We'll see the ISE HDL simulator at work in a moment.

The board I got from Xilinx for this article is the Spartan-3AN Starter Kit (Figure 2), based on the Spartan XC3S700AN FPGA (Figure 3), which contains about 700K system gates. Around it there are several memory chips, a 50MHz onboard clock, a connector for an external clock and several extra components, from D/A and A/D converters to generic I/O pins, assorted LEDs, sliders and push buttons and, finally, a two-line LCD display. The ports (Figure 4), are enough to make a full-custom Linux PC out of this board: 10/100 Ethernet/PHY, USB, keyboard, VGA, serial and stereo mini-jack for PWM audio. A universal power adapter and USB cable are included, as are four different bit files that demonstrate the capabilities of the FPGA. The corresponding design files are freely downloadable from the Xilinx Web site.

Today, you can do really cool things with a $200–$300 FPGA-based board, a typical personal computer and one square foot of desk space.

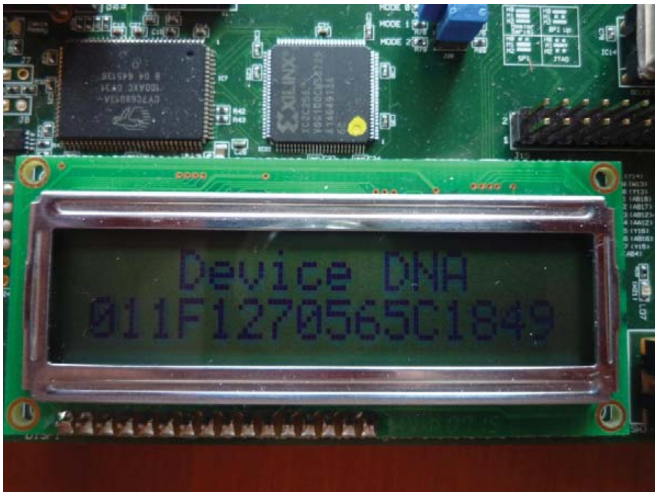


Figure 5. Starter Kit LCD Screen

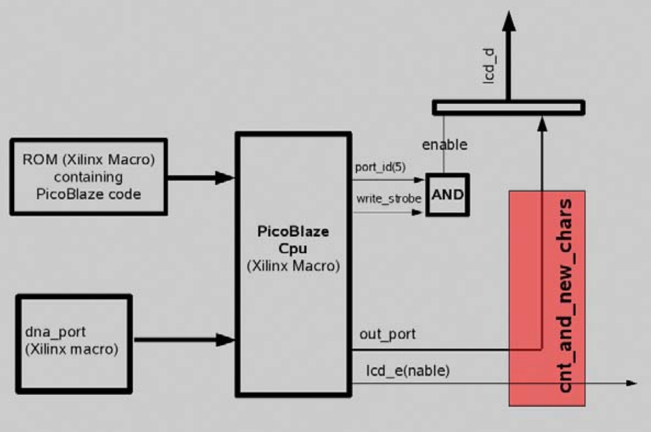


Figure 6. Simplified Circuit Diagram of the DNA Reader

Let's Create New Hardware

In order to show you what it's like to design custom digital hardware and how FPGA development software works, I've modified one of the demo circuits loaded into the Starter Kit, the DNA reader by Xilinx Senior Engineer Ken Chapman.

Spartan FPGAs have a unique ID number, called DNA. The DNA reader displays the intro string "DNA Reader by Ken Chapman", and then this number is displayed on the LCD screen (Figure 5), working as shown in Figure 6. An Xilinx hardware macro called dna_port reads the DNA ID from the silicon. A PicoBlaze processor first displays the intro string, then gets the DNA ID from dna_port and finally sends it, one character at a time, to the LCD interface through the lcd_d data bus. The PicoBlaze code is stored into the dna_ctrl ROM.

My modification consists of a small extra circuit that overwrites the default intro string on the fly with one saying "M Fioretti Linux Journal". Be warned that this is a hack made only for demo purposes. In the real world, if you actually needed to change that string, it would make much more sense to rewrite the PicoBlaze assembly code. Because this is an article about HDL design in FPGAs, however, I went for a solution

October 24, 2009
8 AM - 5 PM
Radssion Resort Orlando-Celebration
2900 Parkway Blvd
Kissimmee, Florida 34747

http://www.floridalinuxshow.com/

Join us for the hottest Linux show in Florida this coming October!

Donald L. Corbet "TheLinuxGuy®"
2009 Show Chairman

Keynote Speaker:
Mr. Robin "Roblimo" Miller

Bryan Smith
2009 Technical Director

Chris Crisafulli
2009 Community Relations Director

Dan Trevino
2009 Web Operations Director

Rod Sharp
2009 Show Director

Charitable Outreach

Children's Miracle Network®

We believe it's important to support worthwhile charitable causes. This year, Florida Linux Show will donate $1.00 of every $15.00 show admission to the Children's Miracle Network. Donation jars will also be available during the show for anyone who wishes to make an additional contribution.

Linux Presentation Areas
- Plug Computing
- Security
- Ubuntu, Ubuntu, Ubuntu
- Linux Thin Clients
- Virtualization
- Cloud Computing
- Interoperability
- Handheld Devices

Open Source Presentation Areas
- Open Source Software
- Open Source Solutions
- Solutions for Businesses
- Open Source Resources
- Open Source PBXs

Official Sponsors

GOLD

GSA PROPOSAL.COM

SILVER

O'REILLY®

BRONZE

TSHIRTBORDELLO.COM
Your place for dirty laundry

MEDIA

LINUX® JOURNAL

LINUX PRO MAGAZINE

LinuxQuestions.org

Supporting Florida LUGs: GatorLUG / Linux enthusiasts & professionals / Pensacola Linux User's Group / Florida Linux Users Group / Linux Users Group of Tallahassee/ Thomasville / Greater Orlando LUG

Florida Ubuntu Loco Team

OSS Atlanta.org

Open Source Software

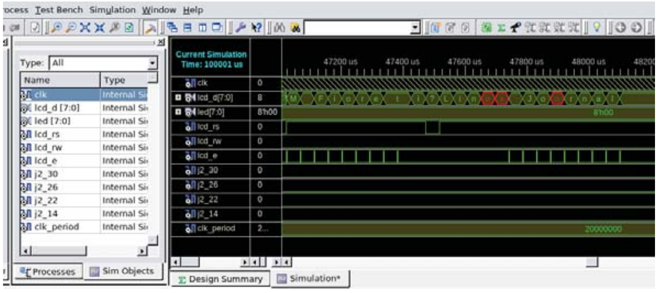
based on easy-to-read HDL code whose effect is easily visible in one picture.

My extra circuit is shown in red in Figure 6: a counter and decoder that detect when the PicoBlaze is driving the LCD data port (lcd_d) and send different characters to it. The VHDL source code corresponding to this extra hardware is shown in Listing 1, which is not the complete, working VHDL file I used, but only an excerpt meant to give you an idea of how HDL coding works.

Lines 1–12 define input and output ports of the top-level circuit, the reading_dna module. You must declare all internal registers and wires before using them (lines 17–23). HDLs support hierarchy; you can instantiate other modules by declaring them and connecting all their ports to the right signals (lines 26–49). Line 53 shows a first example of a synchronous process. Depending on the value of the cnt_ops counter, whenever there is a positive edge of the clock (line 56) and the processor sets the signals write_strobe and port_id(6) high, the lcd_output_data register loads the character from the processor or the one from my extra logic (lines 62–68). The

Listing 1. VHDL Source Code

```
 1  entity reading_dna is
 2      Port (  led    : out std_logic_vector(7 downto 0);
 3              lcd_d  : inout std_logic_vector(7 downto 0);
 4              lcd_rs : out std_logic;
 5              lcd_rw : out std_logic;
 6              lcd_e  : out std_logic;
 7              j2_30  : out std_logic;
 8              j2_26  : out std_logic;
 9              j2_22  : out std_logic;
10              j2_14  : out std_logic;
11              clk    : in std_logic);
12      end reading_dna;
13  --
14  architecture Behavioral of reading_dna is
15  --
16
17  -- start extra signals for LJ demo
18  signal   lcd_e_copy         : std_logic;
19  signal   lcd_e_del_1        : std_logic;
20  signal   lcd_e_del_2        : std_logic;
21  signal   current_character  : std_logic_vector(7 downto 0);
22  signal   cnt_ops            : integer range 0 to 49999999 := 0;
23  -- end extra signals for LJ demo
24  begin
25
26    device_dna: dna_port
27      port map(    din => dna_din,
28                  read => dna_read,
29                 shift => dna_shift,
30                  dout => dna_dout,
31                   clk => dna_clk);
32
33    processor: kcpsm3
34      port map(        address => address,
35                   instruction => instruction,
36                       port_id => port_id,
37                  write_strobe => write_strobe,
38                      out_port => out_port,
39                   read_strobe => read_strobe,
40                       in_port => in_port,
41                     interrupt => interrupt,
42                 interrupt_ack => interrupt_ack,
43                         reset => kcpsm3_reset,
44                           clk => clk);
45
46  program_rom: dna_ctrl
47    port map(       address => address,
48                instruction => instruction,
49                        clk => clk);
50
51  kcpsm3_reset <= '0';
52
53  output_ports: process(clk)
54  begin
55
56    if clk'event and clk='1' then
57      if write_strobe='1' then
58
59        -- 8-bit LCD data output address 40 hex.
60
61        if port_id(6)='1' then
62          -- lcd_output_data <= out_port;
63                        --extra code for LJ demo
64                  if ((cnt_ops >=  8 and cnt_ops <= 17) or
65                       (cnt_ops >= 19 and cnt_ops <= 32))  then
66                    lcd_output_data <= current_character;
67                  else
68             lcd_output_data <= out_port;
69          end if; --end extra code for LJ demo
70        end if;
71
72      end if;
73
74    end if;
75
76  end process output_ports;
77
78  -- LCD interface
79
80  cnt_and_new_chars: process(clk)
81  begin
82    if clk'event and clk='1' then
83
84              if port_id(5)='1' and write_strobe='1' then
85                lcd_e_copy <= out_port(0);
86              end if;
87
88      lcd_e_del_1 <= lcd_e_copy;
89      lcd_e_del_2 <= lcd_e_del_1;
90
```

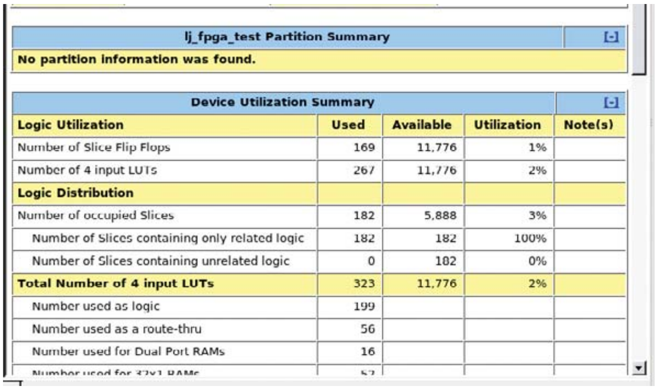


Name	Type
clk	Internal Si
lcd_d [7:0]	Internal Si
led [7:0]	Internal Si
lcd_rs	Internal Si
lcd_rw	Internal Si
lcd_e	Internal Si
j2_30	Internal Si
j2_26	Internal Si
j2_22	Internal Si
j2_14	Internal Si
clk_period	Internal Si

```
clk          0
lcd_d[7:0]   8
led[7:0]     8'h00
lcd_rs       0
lcd_rw       0
lcd_e        0
j2_30        0
j2_26        0
j2_22        0
j2_14        0
clk_period   2...
```

Figure 7. Simulation of Listing 1

```
          if (lcd_e_copy ='1' and lcd_e_del_1='0') then -- posedge
            if cnt_ops=49999999 then                      -- inc counter
              cnt_ops <= 0;
            else
              cnt_ops <= cnt_ops + 1;
            end if; -- if cnt_ops=49999999
                 end if; -- end (lcd_e_copy ='1' and lcd_e_del_1='0')

          if (lcd_e_del_1 ='1' and lcd_e_del_2='0') then  -- posedge
            case cnt_ops is                 -- character generator
                   when  8 => current_character <= "01001101"; -- M
                   when  9 => current_character <= "00100000"; -- space
                   when 10 => current_character <= "01000110"; -- F
                   when 11 => current_character <= "01101001"; -- i
                   when 12 => current_character <= "01101111"; -- o
                   when 13 => current_character <= "01110010"; -- r
                   when 14 => current_character <= "01100101"; -- e
                   when 15 => current_character <= "01110100"; -- t
                   when 16 => current_character <= "01110100"; -- t
                   when 17 => current_character <= "01101001"; -- i

                   when 19 => current_character <= "01001100"; -- L
                   when 20 => current_character <= "01101001"; -- i
                   when 21 => current_character <= "01101110"; -- n
                   when 22 => current_character <= "01110101"; -- u
                   when 23 => current_character <= "01111000"; -- x
                   when 24 => current_character <= "00100000"; -- space
                   when 25 => current_character <= "01001010"; -- J
                   when 26 => current_character <= "01101111"; -- o
                   when 27 => current_character <= "01110101"; -- u
                   when 28 => current_character <= "01110010"; -- r
                   when 29 => current_character <= "01101110"; -- n
                   when 30 => current_character <= "01100001"; -- a
                   when 31 => current_character <= "01101100"; -- l
                   when 32 => current_character <= "00100000"; -- space

                   when others => current_character <= "00100000"; -- space

                   end case;
          end if; -- end (lcd_e_del_1 ='1' and lcd_e_del_2='0')


        end if; -- clk'event and clk='1'
      end process cnt_and_new_chars;
```



lj_fpga_test Partition Summary				[-]
No partition information was found.				

Device Utilization Summary				[-]
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	169	11,776	1%	
Number of 4 input LUTs	267	11,776	2%	
Logic Distribution				
Number of occupied Slices	182	5,888	3%	
Number of Slices containing only related logic	182	182	100%	
Number of Slices containing unrelated logic	0	182	0%	
Total Number of 4 input LUTs	323	11,776	2%	
Number used as logic	199			
Number used as a route-thru	56			
Number used for Dual Port RAMs	16			
Number used for 32x1 RAMs	52			

Figure 8. Final Output Report

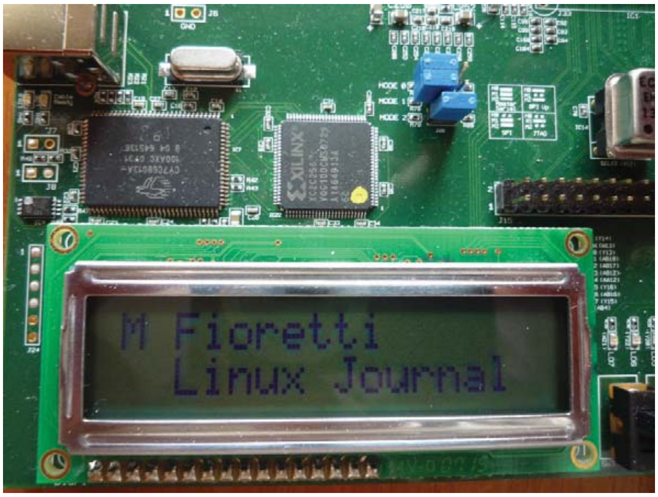


Figure 9. Program Running on FPGA Hardware

cnt_and_new_chars process starting at line 80 does the real work. First, it samples the LCD enable signal to count (line 91) the write accesses to the LCD. One cycle after a write occurs, working with the new counter value (line 99), the process calculates the next current_character that should be displayed. If you look at lines 101–125 you'll see that, instead of the DNA number, the display should show the ASCII string "M Fioretti Linux Journal". A quick simulation (Figure 7) proves that the new process sends those characters to the display at the right times—that is, when the lcd_rs signal is high (low would indicate LCD configuration commands).

The procedure to transform this really simple HDL model into properly connected gates on silicon is equally simple.

The procedure to transform this really simple HDL model into properly connected gates on silicon is equally simple. Double-click, one at a time, the icons in the left-center pane of the Project Navigator shown in Figure 1: synthesize, Implement Design, Generate Programming File and Configure Target Device. If you clicked directly on the last one, ISE would do all the previous steps in the right order anyway, but doing it in steps is a better way to learn. Eventually, you'll get the bit file and a final report like the one shown in Figure 8, showing how much silicon was used. Remember, what we just did is actual hardware—that is, transistors directly connected to do, in real time, what we ordered them to do. All that remains to make it actually happen is to load the bit file in the FPGA. Figure 9 shows the result.

Conclusion

Due to space constraints, I have given only a very limited view of the FPGA design flow. Pushing FPGAs to the limits requires lots of skill and experience. I have said nothing about floor planning, optimization or simulation strategies, nor have I gone into how to run Linux inside FPGAs. All these are excellent topics for future articles.

My goal with this article was simply to show that it is very easy to start learning these skills, and that there already is a strong community to help you. Students, for example, might consider whether FPGAs are what they need to become the next Linus or Steve Jobs. In my opinion, any high school already teaching programming should add FPGA to its courses. If yours is already doing it, please let me know.

Acknowledgements

I wish to thank K. Chapman and F. Porpora at Xilinx, and the FPGA Gurus of DekItalia.com, who helped me greatly in preparing this article.■

Marco (mfioretti.com) is a freelance writer, activist and teacher, concentrating on open digital standards and technologies and their relations and impact on civil rights and education. He's also the author of the Family Guide to Digital Freedom (digifreedom.net).

Resources

Xilinx FPGA Design Tools for Linux by Michael Baxter: **www.linuxjournal.com/article/6857**

Embedded System à la Carte by Michael Baxter: **www.linuxjournal.com/article/6073**

Spartan Starter Kit Datasheet: **www.xilinx.com/products/devkits/HW-SPAR3AN-SK-UNI-G.htm**

OpenCore's Linux and Xilinx FPGA Dev Board: **www.opencores.org/?do=project&who=eus100lx**

Xilinx Linux Forum: **forums.xilinx.com/xlnx/board?board.id=ELINUX**

Xilinx Documentation: **www.xilinx.com/design**

Linux KVM as a Learning Tool

Low-level system programming is a difficult task, but with Linux KVM, it's a whole lot easier.

DUILIO JAVIER PROTTI

Low-level system programming is a difficult task, and acquiring expertise in the areas of interrupt handling and memory segmentation/paging can be a time-consuming and frustrating process if you're working right down on the metal. An alternative choice is to use a virtual machine or the Linux KVM module to create and run your own mini-kernels from scratch quickly.

The KVM Module

The KVM (Kernel-based Virtual Machine) module turns a Linux host into a VMM (Virtual Machine Monitor), and it has been included in the mainline Linux kernel since version 2.6.20. A VMM allows multiple operating systems to run concurrently on a computer. These guest operating systems execute on the real (physical) processor, but the VMM (or hypervisor) retains selective control over certain real system resources, such as the physical memory and the I/O capabilities.

When a guest tries to perform an action on a controlled resource, the VMM takes control from the guest and executes the action in a fashion that keeps it from interfering with other guest operating systems. As far as the guest knows, it thinks it is running on a platform with no VMM—that is, it has the illusion of running on a real machine. For example, the guest can do memory paging and segmentation and interrupt manipulation without interfering with the same mechanisms within other guest operating systems or within the VMM itself.

A normal Linux process has two modes of execution: kernel mode and user mode. KVM adds a third one: guest mode (Figure 1). When a guest process is executing non-I/O guest code, it will run in guest mode or perhaps better-named guest-user mode. This is the "user" mode shown inside the "Guest mode" box in Figure 1. In kernel mode (guest-kernel), the process handles exits from guest-user mode due to I/O or other special instructions. This is the "kernel" mode shown inside the "Guest mode" box in Figure 1. In user mode, the process performs I/O on behalf of the guest. This is the "I/O Ops" box shown within the normal "User mode" box in Figure 1. For more on how KVM itself operates, see the KVM Web site and the many on-line articles about it.

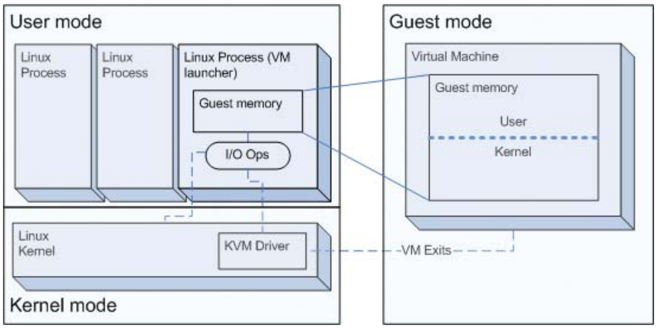


Figure 1. KVM Modes of Execution

The examples presented here require a recent Linux kernel with the KVM module installed and the LibKVM library to interact with the module from userspace. You can install the corresponding package(s) from your favorite distribution or compile the KVM source package (from SourceForge) to create both the module and LibKVM library. Note that the KVM module works only on platforms with hardware support for virtualization; most newer Intel and AMD 64-bit-capable processors have this support.

The rest of this article shows how to build a series of guest-mode programs (kernels) as well as a user-mode program to emulate their I/O (a virtual machine launcher).

What Gets Virtualized?

The basic components of contemporaneous computer machines are memory, one or more CPUs and one or more I/O devices. Therefore, a virtual computer machine should have these three kinds of components. Linux KVM has the ability to handle the virtual machine's memory and CPUs (with hardware help). The third ingredient, I/O, currently is left to the programmer and has to be handled in a custom way.

For instance, the KVM distribution comes with qemu-kvm, a modified QEMU program that builds virtual machines using LibKVM and emulates various I/O devices, such as a VGA card, PS/2 mouse and keyboard and an IDE disk. We are not going to use qemu-kvm here, but rather we will code a virtual machine launcher from scratch to keep our first examples simple and to learn how a program like qemu-kvm does its work.

How to Create a Virtual Machine Launcher

The KVM module exposes a character device (/dev/kvm) for interaction with userspace. For simplicity, we won't access this device directly but instead through LibKVM (API defined in libkvm.h). Use the methods shown in Listing 1 to build the virtual machine launcher (code based on Avi Kivity's test driver program included in the KVM sources).

To start, create a KVM context with kvm_init(). The first argument is a kvm_callbacks structure to specify the handlers to be called when I/O or some system-sensitive instructions are executed inside the virtual machine—for example, when the guest executes something like this:

```
mov     $0x0a,%al
outb    %al,$0xf1    // output value 0x0a to I/O port 0xf1
```

the guest will exit from guest mode, and the configured outb() callback function is called in user mode (with values 0xf1 and

Listing 1. LibKVM Methods Used for Our Launcher

```
kvm_context_t  kvm_init(struct kvm_callbacks    *callbacks,
                        void                    *opaque);

int             kvm_create(kvm_context_t   kvm,
                           unsigned long   phys_mem_bytes,
                           void            **phys_mem);

int             kvm_create_vcpu(kvm_context_t  kvm,
                                int            slot);

void            *kvm_create_phys_mem(kvm_context_t    kvm,
                                     unsigned long    phys_start,
                                     unsigned long    len,
                                     int              log,
                                     int              writable);

int             kvm_run(kvm_context_t    kvm,
                        int              vcpu);
```

Listing 2. I/O Callbacks (used in launcher.c)

```
static int my_inb(void *opaque, int16_t addr, uint8_t *data)
                     { puts ("inb"); return 0; }

static int my_inw(void *opaque, uint16_t addr, uint16_t *data)
                     { puts ("inw"); return 0; }

static int my_inl(void *opaque, uint16_t addr, uint32_t *data)
                     { puts ("inl"); return 0; }

static int my_outb(void *opaque, uint16_t addr, uint8_t data)
                     { puts ("outb"); return 0; }

static int my_outw(void *opaque, uint16_t addr, uint16_t data)
                     { puts ("outw"); return 0; }

static int my_outl (void *opaque, uint16_t addr, uint32_t data)
                     { puts ("outl"); return 0; }

static int my_pre_kvm_run(void *opaque, int vcpu)
                     { return 0; }

  ... and similar for my_mmio_read, my_mmio_write,
      my_debug, my_halt, my_shutdown, my_io_window,
      my_try_push_interrupts, my_try_push_nmi,
      my_post_kvm_run, and my_tpr_access

static struct kvm_callbacks my_callbacks = {
    .inb                  = my_inb,
    .inw                  = my_inw,
    .inl                  = my_inl,
    .outb                 = my_outb,
    .outw                 = my_outw,
    .outl                 = my_outl,
    .mmio_read            = my_mmio_read,
    .mmio_write           = my_mmio_write,
    .debug                = my_debug,
    .halt                 = my_halt,
    .io_window            = my_io_window,
    .try_push_interrupts  = my_try_push_interrupts,
    .try_push_nmi         = my_try_push_nmi,  // added in kvm-77
    .post_kvm_run         = my_post_kvm_run,
    .pre_kvm_run          = my_pre_kvm_run,
    .tpr_access           = my_tpr_access
};
```

0x0a for its second and third parameters, respectively).

Initially, use dummy callbacks. Create and reference them in a variable called my_callbacks, as shown in Listing 2. Most field names are self-explanatory, but for a brief description of each of them, refer to the comments in the structure definition in libkvm.h.

To create the virtual machine itself, use kvm_create(), whose second argument is the amount of RAM in bytes desired for it, and the third argument is the address of a location that will in turn contain the address of the beginning of the memory space reserved for the virtual machine (the "guest memory" box in Figure 1). Note that kvm_create() does *not* allocate memory for the virtual machine.

To create the first virtual CPU, use kvm_create_vcpu() with a value of 0 for the slot parameter—versions less than 65 create the first virtual CPU during the call to kvm_create().

There are several methods to allocate memory for the virtual machine—for example, kvm_create_phys_mem(). The second argument of kvm_create_phys_mem() is the starting physical address of the requested region *in the guest memory* (in the pseudo-"physical memory" of the virtual machine, not in the physical memory of the host). The third argument is the length, in bytes, of the region. The fourth indicates whether dirty page logging should be activated in the requested region, and the fifth argument indicates whether the pages may be written. On success, it returns the location of the allocated memory area as an address in the virtual address space of the calling process.

Invoke the functions of Listing 1 within the same KVM context to create your first virtual machine, and execute it with kvm_run(). This function will return only if an I/O handler pointed in my_callbacks returns a nonzero value or an exception occurs that neither the guest OS nor KVM can handle.

Listing 3 contains the code for the launcher, including the load_file() function to copy the guest kernel image from a file to the virtual machine's memory space. Why is this image copied at offset 0xf0000 of the guest's memory space? Because of the way real-mode works, as explained in the next section.

16-Bit Real-Address Mode

Processors compatible with the x86 architecture can support different operating modes. Two of them are 16-bit real-address mode. The most frequently used, these days at least, is 32-bit protected mode. The processor starts in real-address mode after a power-up or reset (so platform initialization code has to be written for this mode) and jumps to the instruction at address 0xFFFF0. Usually, the BIOS's initialization routine is located here. The first instruction of our simple kernel will be located there to take control of the platform as soon as it boots. Although with KVM it is possible to start a virtual machine directly in protected mode, our launcher won't do that in order to learn how to manipulate a PC just after power-up.

The 16-bit real-address mode is a legacy mode inherited from the Intel 8086 processor, which is able to address up to 1Mb of memory. 1Mb is 2^{20} bytes, so addresses require 20 bits. Given that the 8086's registers are only 16-bit wide, addresses are built by pairing two values. The first value is used as a selector (stored in a segment register), and the second value is used as an offset. With these, physical addresses are computed by the formula: 16 * selector + offset.

For example, the selector:offset 0xDEAD:0xBEEF represents the physical address 0xEA9BF. To multiply the selector (0xDEAD) by 16, simply add a 0 to the right side of the number (0xDEAD0). The addition then becomes the following:

```
  0xDEAD0
+ 0x0BEEF
  -------
  0xEA9BF
```

Listing 3. Our First Virtual Machine Launcher (launcher.c)

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <libkvm.h>

/* callback definitions as shown in Listing 2 go here */

void load_file(void *mem, const char *filename)
{
    int  fd;
    int  nr;

    fd = open(filename, O_RDONLY);
    if (fd == -1) {
        fprintf(stderr, "Cannot open %s", filename);
        perror("open");
        exit(1);
    }
    while ((nr = read(fd, mem, 4096)) != -1  &&  nr != 0)
        mem += nr;

    if (nr == -1) {
        perror("read");
        exit(1);
    }
    close(fd);
}

#define MEMORY_SIZE     (0x1000000)     /* 16 Mb */
#define FIRST_VCPU      (0)

int main(int argc, char *argv[])
{
    kvm_context_t   kvm;
    void            *memory_area;

    /* Second argument is an opaque, we don't use it yet */
    kvm = kvm_init(&my_callbacks, NULL);
    if (!kvm) {
        fprintf(stderr, "KVM init failed");
        exit(1);
    }
    if (kvm_create(kvm, MEMORY_SIZE, &memory_area) != 0) {
        fprintf(stderr, "VM creation failed");
        exit(1);
    }
#ifndef KVM_VERSION_LESS_THAN_65
    if (kvm_create_vcpu(kvm, FIRST_VCPU)) {
        fprintf(stderr, "VCPU creation failed");
        exit(1);
    }
#endif
    memory_area = kvm_create_phys_mem(kvm, 0, MEMORY_SIZE, 0, 1);
    load_file(memory_area + 0xf0000, argv[1]);

    kvm_run(kvm, FIRST_VCPU);

    return 0;
}
```

Note that given a fixed value for the selector, it is possible to reference only 64Kb of memory (the offset's allowed range). Programs bigger than 64Kb must use multi-segment code. We will keep our kernel simple and make it fit into a single 64Kb segment. Our launcher will put the kernel image in the last segment (where the 0xFFFF0 entry point resides). The last segment starts at 0xF0000 as shown by the following calculation:

```
Start of the last segment
    = (Maximum 8086 Memory) - (Segment Size)
    = 1MB - 64KB
    = 0x100000 - 0x10000 = 0xF0000
```

A memory map of this is shown in Figure 2.

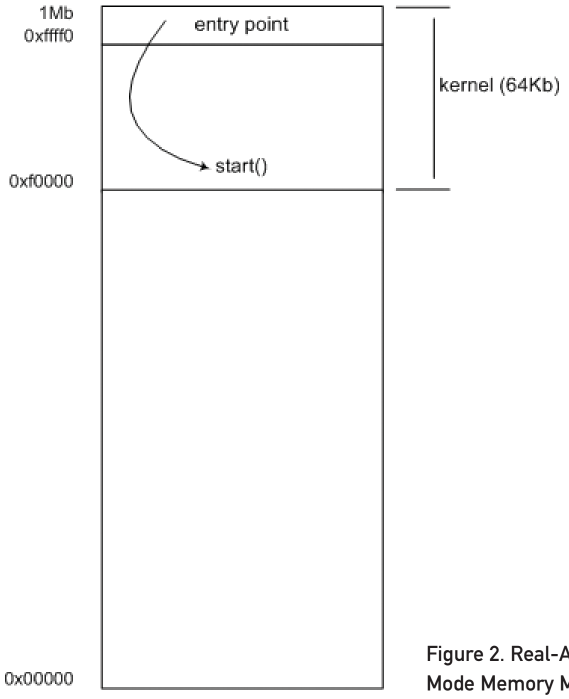


Figure 2. Real-Address Mode Memory Map

Our 16-Bit Real-Address Mode Kernel

We now can write a kernel in assembler with its first instruction at offset 0xFFFF0. Note that unlike many processors, the x86 processor does not have a reset "vector". It does not *use* the value at 0xFFFF0 as the location of the reset code; rather, it begins executing the code that is *at* 0xFFFF0. Therefore, the "normal" code to place at 0xFFFF0 is a jump to the actual reset code.

Our first kernel is shown in Listing 4. It merely sets the AX

Listing 4. kernel1.S

```
.code16                    // Generate 16-bit code
start:                     // Kernel's main routine
        xor %ax, %ax
1:
        jmp 1b             // Loop forever

. = 0xfff0                 // Entry point
        ljmp    $0xf000, $start
```

register to 0 and then loops forever.

In the second to the last line, the dot (.) refers to the current location counter. Therefore, when we write:

```
. = 0xfff0
```

we instruct the assembler to set the current location to address 0xFFF0. In real-mode, address 0xFFF0 is relative to the current segment. Where does the segment offset get specified? It comes from the call to load_file() in Listing 3. It loads the kernel at offset 0xF0000. This, combined with the assembler offset, will place the ljmp at address 0xFFFF0, as required.

How to Build It

The kernel binary should be a raw 64Kb 16-bit real-address mode image, and not a normal ELF binary (the standard binary format used by Linux). To do this, we need a special linker script. We use GNU ld for this, of course, which accepts script files to provide explicit control over the linking process.

A linker is a program that combines input binary files into a single output file. Each file is expected to have, among other things, a list of sections, sometimes with an associated block of data. The linker's function is to map input sections into output sections. GNU ld uses, by default, a linker script specific for the host platform, which you can view by using the -verbose flag:

Although with KVM it is possible to start a virtual machine directly in protected mode, our launcher won't do that in order to learn how to manipulate a PC just after power-up.

```
$ gcc -Wl,-verbose hello-world.c
```

To build our kernel, we don't use the default script but instead the simple script kernel16.lds, shown in Listing 5.

The SECTIONS command controls how to make the mapping and how to place the output sections in memory.

Listing 5. Linker Script kernel16.lds

```
OUTPUT_FORMAT(binary)

SECTIONS {
        . = 0;
        .text : { *(.init) *(.text) }
        . = ALIGN(4K);
        .data : { *(.data) }
        . = ALIGN(16);
        .bss : { *(.bss) }
        . = ALIGN(4K);
        .edata = .;
}
```

Listing 6. Building a 16-Bit Kernel Image

```
$ gcc -nostdlib -Wl,-T,kernel16.lds kernel1.S -o kernel1
$ ls -oh kernel1
-rwxr-xr-x 1 djprotti 64K 2008-10-17 19:09 kernel1
```

Directives follow the syntax:

```
.output-section : [optional-args]
                  { input-section, input-section, ... }
```

The kernel16.lds script sets the current location at offset 0x0. Then, the output .text section will start there and will contain the contents of any .init and .text input sections.

Next, we align the current location to a 4KB boundary and create the .data and .bss output sections. Use kernel16.lds to generate the kernel image as shown in Listing 6.

The -nostdlib flag avoids linking the standard system startup files and libraries (these will not be available inside our virtual machines). After this, we have our 64Kb 16-bit real-address kernel image.

How to Test It All

The Makefile in Listing 7 contains the commands to build both the kernel and the launcher.

Launch the virtual machine with kernel1 as guest with the following command:

```
$ ./launcher kernel1
```

If everything goes well, you will see no output, and the guest kernel should be consuming all of its available CPU. If you run the top command in another console, and you see output similar to that of Listing 8 (100% CPU usage for the launcher process), you have your kernel running in your first KVM virtual machine!

An Improved Kernel

Now, let's build a kernel that communicates with the world. First, choose one of the I/O ports and use it to implement a "serial port”. Name the chosen port as IO_PORT_PSEUDO_SERIAL (as shown in Listing 10), then modify the outb callback in the launcher to interpret bytes sent to this port as characters

Listing 7. Makefile

```
# If KVM was compiled from sources and you have errors about
# missing asm/kvm*.h files, copy them from
# kvm-XX/kernel/include/asm/* to {prefix}/include/asm/
CC=gcc
KERNEL16_CFLAGS=-nostdlib -ffreestanding -Wl,-T,kernel16.lds

all:     launcher kernel1

launcher: launcher.o
        $(CC) launcher.o /usr/lib/libkvm.a -o launcher

launcher.o:

kernel1: kernel1.S
        $(CC) $(KERNEL16_CFLAGS) kernel1.S -o kernel1

clean:
        rm *.o launcher kernel1
```

Listing 8. Output of top While Our Launcher Is Running

```
 PID USER     S %CPU %MEM    TIME+  COMMAND
8002 djprotti R  100  0.8   1:53.19 launcher
7428 djprotti S    0  0.8   0:04.45 gnome-terminal
8005 djprotti R    0  0.0   0:00.02 top
   1 root     S    0  0.0   0:03.92 init
   2 root     S    0  0.0   0:00.00 kthreadd
   3 root     S    0  0.0   0:00.12 migration/0
   4 root     S    0  0.0   0:02.76 ksoftirqd/0
   5 root     S    0  0.0   0:00.01 watchdog/0
```

Listing 9. Pseudo-Serial Port Implementation in launcher.c

```
#include "runtime.h"

static int my_outb (void *opaque, uint16_t addr, uint8_t data)
{
    if (addr == IO_PORT_PSEUDO_SERIAL)
        if (isprint(data) || data == '\n')
            putchar(data);
        else
            putchar('.');
    else
        printf("outb: %x, %d\n", addr, data);
    fflush (NULL);

    return 0;
}
```

Advertiser Index

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

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

Advertiser	Page #
1&1 Internet, Inc. www.oneandone.com	1
Aberdeen, LLC www.aberdeeninc.com	7
ApacheCon www.us.apachecon.com/us2008	37
Archie McPhee www.mcphee.com	78
ASA Computers, Inc. www.asacomputers.com	49
Cari.net www.cari.net	71
Digi-Key Corporation www.digi-key.com	78
Emac, Inc. www.emacinc.com	72
EmperorLinux www.emperorlinux.com	33
Florida Linux Show www.floridalinuxshow.com	55
Gecad Technologies/Axigen www.axigen.com	79
Genstor Systems, Inc. www.genstor.com	57
iXsystems, Inc. www.ixsystems.com	3
Logic Supply, Inc. www.logicsupply.com	75
Lullabot www.lullabot.com	13, 19
Microway, Inc. www.microway.com	C4, 5
Ohio Linux Fest www.ohiolinux.org	59
Parallels, Inc. www.parallels.com	27, 79
Polywell Computers, Inc. www.polywell.com	35, 79
Rackspace Managed Hosting www.rackspace.com	C3
Saint Arnold Brewing Company www.saintarnold.com	78
ServerBeach www.serverbeach.com	41
Servers Direct www.serversdirect.com	9
Silicon Mechanics www.siliconmechanics.com	23, 61
StrayTats www.straytats.com	78
Supercomputing SC09 sc09.supercomputing.org	51
Technologic Systems www.embeddedx86.com	69
USENIX LISA www.usenix.org/events/usenix07	25
Ubiquiti Networks, Inc. www.ubnt.com	C2
Universitat Oberta de Catalunya www.otsmediasa.com	21
Utah Open Source Convention 2008.utosc.com/pages/registration	67
Utilikilts www.utilikilts.com	78
Verio www.verio.com	11, 79

ATTENTION ADVERTISERS

January 2010 Issue #189 Deadlines

Space Close: October 26; **Material Close:** November 3

Theme: Security

BONUS DISTRIBUTIONS:
Do It with Drupal, New Mexico GNU/LinuxFest, SharePoint Tech Conference

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

Listing 10. kernel2.S

```
#include "runtime.h"

.code16
start:
    mov     $0x48,%al     // H
    outb    %al,$IO_PORT_PSEUDO_SERIAL
    mov     $0x65,%al     // e
    outb    %al,$IO_PORT_PSEUDO_SERIAL
    mov     $0x6c,%al     // l
    outb    %al,$IO_PORT_PSEUDO_SERIAL
    mov     $0x6c,%al     // l
    outb    %al,$IO_PORT_PSEUDO_SERIAL
    mov     $0x6f,%al     // o
    outb    %al,$IO_PORT_PSEUDO_SERIAL
    mov     $0x0a,%al     // new_line
    outb    %al,$IO_PORT_PSEUDO_SERIAL

    hlt                   // halt the processor

. = 0xfff0
    ljmp    $0xf000, $start
```

Listing 11. runtime.h

```
#ifndef __RUNTIME_H__
#define __RUNTIME_H__

// port to use for general purpose output
#define IO_PORT_PSEUDO_SERIAL   0xf1

#endif /* __RUNTIME_H_ */
```

printed to a serial console, and redirect them to launcher's standard output as shown in Listing 9.

Then, build a second kernel (kernel2) whose only task is to print "Hello\n" to its pseudo-serial port and then halt, as shown in Listing 10.

Build both the launcher and kernel2, and run them as usual. The output should be similar to this:

```
$ ./launcher kernel2
Hello
```

Now the top command should show 0% CPU usage for the launcher process, because its virtual CPU is halted.

As a last example, an improved kernel is shown in Listing 12, using the OUTSB string output instruction and the REP prefix to repeat it the number of times specified by CX. Interestingly, this code generates *only one* I/O exit to output the entire string. Compare this against the previous kernel2, which generates one I/O exit for each outb execution, with the associated overhead due to context switches. You can use the kvm_stat Python script from the KVM sources to see this and other behaviours of the virtual machines.

Listing 12. kernel3.S (output using OUTSB)

```
#include "runtime.h"

.code16
start:
    mov      $(IO_PORT_PSEUDO_SERIAL), %dx
    cs lea   greeting, %si
    mov      $14, %cx
    cs rep/outsb     // kvm_stat reports only
                     // *one* io_exit using this
    hlt

.align 16
greeting:
    .asciz   "Hello, World!\n"

. = 0xfff0
    ljmp     $0xf000, $start
```

The CS prefix before the LEA and OUTSB instructions are needed to fetch data (greeting string) from the code segment.

What's Next?

At this point, you have the basis to experiment with all kinds of real-mode code. You can extend the examples to set an IDT and handle interrupts or add more I/O devices. A good starting point is interrupts to learn the constraints of interrupt context, and another one is to investigate the rest of LibKVM's methods.

However, real mode is not enough to learn all the things that current kernels do on the x86 platform. For this reason, in a follow-up article, we will extend our launcher a little in order to handle kernels running in 32-bit protected mode. This change will give us the ability to write kernels in the C language, allowing for rapid development of bigger kernels. It also will open the door for experimenting with segmentation, paging, privilege levels (two or more rings) and more.

Remember, low-level system programming is a challenging task, but with Linux KVM, it can be easy. So, go ahead and code, have fun and you will learn a lot about how computer systems work in the process!■

Duilio Javier Protti (duilio.j.protti@intel.com) is a software engineer with Intel Corp., in Cordoba, Argentina. He currently is working on a team specializing in virtualization technology. Before joining Intel, he wrote LibCMT (a library for composable memory transactions), was the maintainer of the Infinity XMMS plugin and contributed to various open-source projects, such as Nmap, Libvisual and others.

Resources

A Good Book on PC Assembly by Dr Paul Carter: **drpaulcarter.com/pcasm**

KVM Sources: **sourceforge.net/projects/kvm**

Utah Open Source Conference 2009

(afford|scal|reli)ability

The growth of Utah Open Source Conference is a testament to the strength of the free and open source software community!
--Clint Savage,
Executive Director, Utah Open Source

...I, for one, won't be surprised when UTOSC quickly becomes one of the defining FOSS community shows for the entire USA.
--Paul Frields, Fedora Project Leader

UTOSC 2009 will bring developers, beginners and businesses together to show the affordability, scalability, reliability or any ability of free and open source software.

If you are interested in free and open source software, this is the place to be in 2009!

Register now!

http://utos.com

Three full days of Open Source Software presentations and activities.

Technology Job Fair and Networking opportunities

One incredible day dedicated to families and open source

Brought to you by:

Salt Lake Community College

Novell.

XMISSION
INTERNET

Android Everywhere!

Put Android where you'd least expect it—from phones to virtual machines.

BILL CHILDERS

It seems you can't hit a tech news site or read a magazine these days without encountering some mention of Android. If you've not been keeping up on the news, Android is a Linux-based OS, designed by Google that's geared to run on lightweight devices like cellular phones and Webpads. One of Android's key features is that developers can write code for the OS in Java, making it a very easy platform for developers to work with.

The first Android-powered product was the T-Mobile G1, made by HTC and known as the Dream. The Dream has a 528MHz ARM11 CPU, 192MB of RAM and 256MB of Flash, so it's a capable smartphone, and it's part of an open standards effort from the Open Handset Alliance and Google. As a result of Google's involvement, it's been touted as "The Google Phone" by the press.

For this article, I set out to see how many devices I could put Android on and how difficult each one was to get running. Because we're hearing buzz about Android-powered Webpads, phones and even Netbooks, I wanted to discover just what the hype was about. I elected to skip past the gloss and dive into the OS itself and see exactly what it takes to get it running on a device.

Exploring Android on the G1

The HTC Dream/T-Mobile G1 phone (Figure 1) comes in a developer version that allows unsigned binaries to be run, and it does a few other things that the regular G1 doesn't do. Because I had a regular G1, I figured a good place to start my Android exploration would be to see if I could get the developer OS running on a release device. Not surprisingly, T-Mobile frowns upon anyone doing this and puts roadblocks in the device to prevent it from happening. Also surprising, it turned out to be really easy, as there are holes in the firmware that allow you to gain root access on the phone.

Once you get root, you pretty much can do what you want to the device, including flashing the developer version of the OS. The "Hacking Your G1/Dream" link in the Resources section of this article contains the details, but basically the steps are mostly standard Linux command-line fare, taking advantage of a bug in the firmware where everything you type at the keyboard is sent to the OS. (Try typing `reboot` on an older G1 at any time. It will reboot spontaneously!)

You most likely will have to downgrade your firmware to a version that has the known exploit, and then take advantage of the exploit to gain root, but once that's done, you can reflash the device with any firmware you choose, using the standard update method. If you choose to do this, standard disclaimers and waivers apply about breaking your hardware (see the Disclaimers and Waivers sidebar), as you're definitely doing something that has the potential to turn your several-hundred-dollar smartphone into an expensive brick. If you do decide to do this, however, I recommend JF's excellent 1.51 ADP build, as that retains root capabilities and allows you to run unsigned binaries (see the link to JF's Blog in the Resources section).

Once the latest build of the OS (code-named Cupcake) is on the now-rooted phone, you can build your own binaries for it, if you're a coder type, or grab things others have done from the Internet. Of course, if you *do* download someone else's binaries, standard disclaimers apply there too. Can you imagine the data charges that could be possible if you had a rootkit or trojan on your always-connected

Figure 1. The HTC Dream/T-Mobile G1 Smartphone

Disclaimers and Waivers

I can't stress enough: take extreme care when hacking mobile devices. Besides the obvious peril of bricking the device and making it unusable, there is the even greater danger of causing yourself an extremely large cellular bill. This is a really good way to test to see if your "unlimited" data plan truly is unlimited. You'll find out the hard way that most aren't. Take precautions, and try to find hacks that other people have done and reported success doing. Unless you have unlimited funds, blazing a trail in this area can become expensive quickly.

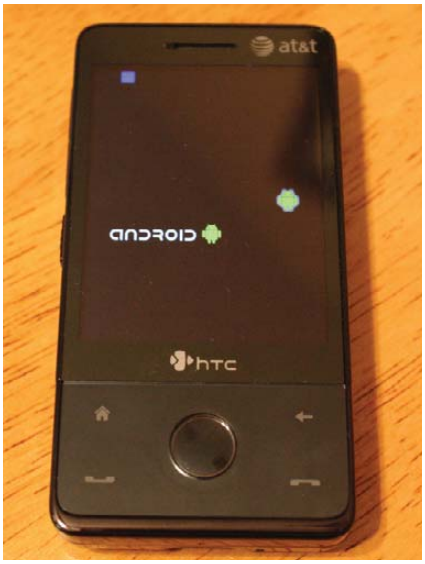


Figure 2. Android Booting on a WinMo Phone

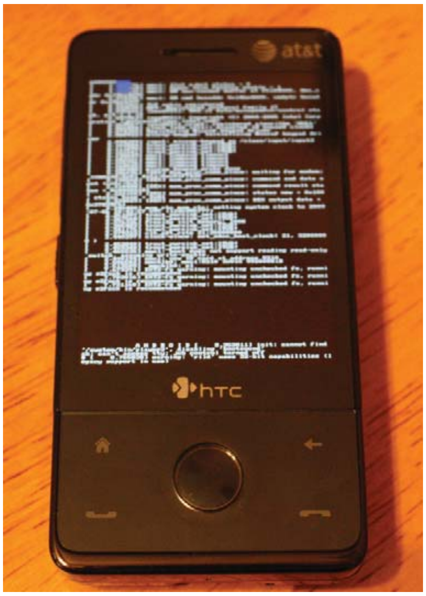


Figure 3. Kernel Messages on Windows Mobile?

mobile device? This is exactly why T-Mobile doesn't want the devices hacked, as it could congest its network.

Going Boldly Where No One Has Gone Before...

As getting control of a G1 was relatively easy, I started wondering about installing Android on other devices. A quick scan of my desk revealed an unused AT&T Fuze cell phone, otherwise known as the HTC Touch Pro (code-named Raphael100). The HTC Touch runs Windows Mobile, not Android, but the units are both made by HTC and seemed to have similar hardware. I began to wonder if it would be possible to run Android on that phone, because they had the same manufacturer.

I started researching the feasibility of running Android on the Touch Pro, and I discovered that a group of enterprising developers already had done this very thing. Luckily for me, they made their distribution available as well (see Resources), so getting Android running on the HTC Touch Pro was almost as easy as getting it going on the G1 (Figure 2).

As it turns out, getting Android running on the Touch Pro was as easy as downloading a .zip file of the distribution and unzipping the contents of that file to a MicroSD card. Once that was done, I put the card into the phone and used the Windows Mobile file manager to navigate to a directory on the card called tmp. Within that directory was a program called haret.exe. I ran that, and the screen on the phone went black, and then it showed me the familiar Linux kernel messages as it began to boot Android (Figure 3).

Just like the loadlin days, when a DOS program could bootstrap the Linux kernel into booting, haret.exe bootstraps Android from the Windows Mobile environment. Before long, I was greeted with the Android desktop environment. However, all was not right with this port of Android. Although I could launch some of the applications, like the contact manager and browser, the 3G modem inside the phone was not operational, nor were the microphone or speaker. About the only thing I could do was send and receive SMS messages, though it did do that exactly like the G1. Yes, just like the early days of Linux, it seems that device drivers for various pieces of hardware don't exist or don't work properly. However, this is a rapidly moving target, and the Android developers are working hard to make progress in this area.

Where Are You Using Android?

We want to know what devices you're running Android on. Share with our editors in the Android Everywhere forum at **www.linuxjournal.com/forums/hot-topics** where author and *LJ* Virtual Editor Bill Childers will be moderating.

Android on a Netbook—Is It Usable?

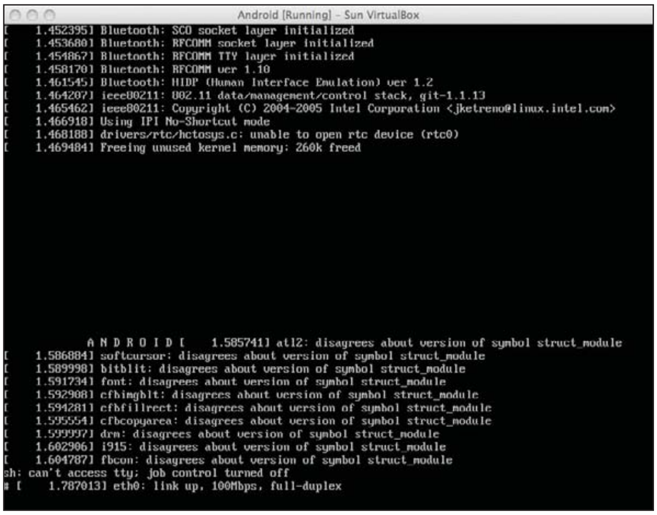
Because I had such relative success with Android on a Windows Mobile device, I proceeded on to see what it would take to get it running on my Netbook, an Acer Aspire One. I found that there is a project underway to port Android to x86 platforms, so I started to investigate. This seemed semi-straightforward, even though it required me to custom build the distribution myself.

The x86 porting project allows you to build either a VirtualBox virtual machine or an installer for an ASUS Eee PC as the target. I found the main Android code repository, which had excellent instructions on how to set up your build environment and get all the various libraries installed (see Resources). If you're running Ubuntu, all that's needed is a simple apt-get statement, and all the build dependencies and libraries are installed in a snap. However, once you've done that, you shouldn't get the source from the main Android repository.

Although you can get it from the main repository and patch it yourself, there is a prepatched source tree available at the x86 porting site via SVN (see Resources). I spent a lot of time trying to patch the main source of Android for running on x86 only to have it fail near the end of the three-hour-long build cycle, or worse, fail to boot the OS image after it reported a clean build.

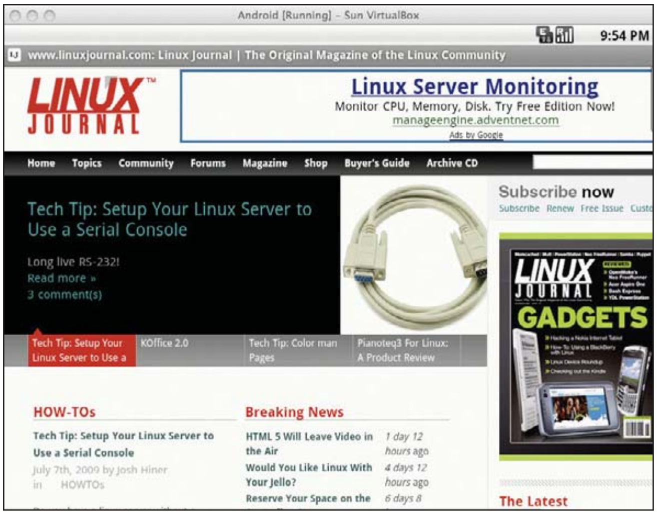
The prepatched code built correctly on the first try, and I was able to get the VirtualBox virtual machine going with little hassle (Figure 4). One thing I did discover is that the VirtualBox virtual machine failed to boot unless the VM was configured to have a serial port.

Once my Android VM was up and running, I started playing with it. Unlike the HTC Touch Pro port, this port had



```
[    1.452395] Bluetooth: SCO socket layer initialized
[    1.453680] Bluetooth: RFCOMM socket layer initialized
[    1.454867] Bluetooth: RFCOMM TTY layer initialized
[    1.458170] Bluetooth: RFCOMM ver 1.10
[    1.461545] Bluetooth: HIDP (Human Interface Emulation) ver 1.2
[    1.464207] ieee80211: 802.11 data/management/control stack, git-1.1.13
[    1.465462] ieee80211: Copyright (C) 2004-2005 Intel Corporation <jketreno@linux.intel.com>
[    1.466918] Using IPI No-Shortcut mode
[    1.468188] drivers/rtc/hctosys.c: unable to open rtc device (rtc0)
[    1.469484] Freeing unused kernel memory: 260k freed




          A N D R O I D [    1.585741] atl2: disagrees about version of symbol struct_module
[    1.586884] softcursor: disagrees about version of symbol struct_module
[    1.589998] bitblit: disagrees about version of symbol struct_module
[    1.591734] font: disagrees about version of symbol struct_module
[    1.592908] cfbimgblt: disagrees about version of symbol struct_module
[    1.594281] cfbfillrect: disagrees about version of symbol struct_module
[    1.595554] cfbcopyarea: disagrees about version of symbol struct_module
[    1.599997] drm: disagrees about version of symbol struct_module
[    1.602906] i915: disagrees about version of symbol struct_module
[    1.604787] fbcon: disagrees about version of symbol struct_module
sh: can't access tty; job control turned off
# [    1.787013] eth0: link up, 100Mbps, full-duplex
```

Figure 4. Android Booting in a VM!



www.linuxjournal.com: Linux Journal | The Original Magazine of the Linux Community

Linux Server Monitoring
Monitor CPU, Memory, Disk. Try Free Edition Now!
manageengine.adventnet.com
Ads by Google

Subscribe now

Tech Tip: Setup Your Linux Server to Use a Serial Console

Long live RS-232!
Read more »
3 comment(s)

Tech Tip: Setup Your Linux Server to Use a | KOffice 2.0 | Tech Tip: Color man Pages | Pianoteq3 For Linux: A Product Review

HOW-TOs

Tech Tip: Setup Your Linux Server to Use a Serial Console
July 7th, 2009 by Josh Hiner
in HOWTOs

Breaking News

HTML 5 Will Leave Video in the Air — 1 day 12 hours ago
Would You Like Linux With Your Jello? — 4 days 12 hours ago
Reserve Your Space on the — 6 days 8

The Latest

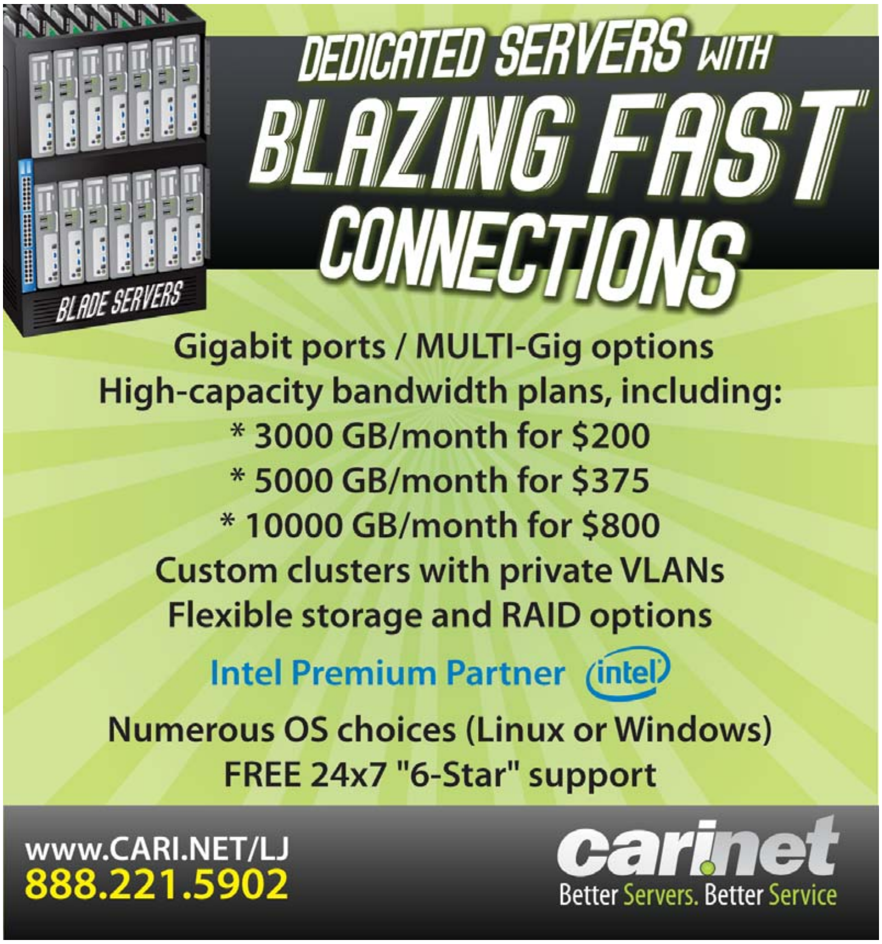
Figure 5. *Linux Journal*—Android Style

full networking. The first thing I did was fire up the browser (Figure 5) and was that ever surprising! It's possibly the fastest browser I've ever seen. Granted, it's optimized for a 500MHz ARM, and I was running it on a 2GHz dual-core CPU, but it was lightning fast. It's still a mobile browser, however, so it's not full-featured by any stretch of the word.

However, aside from surfing the Web, mobile style, and playing with the Terminal application, there wasn't much of interest with Android on a VM. The applications aren't compelling enough to run there. As I had it built on a VM, I tried to port that to my Aspire One to see any performance differences, but there were enough hardware incompatibilities that I reached the point of diminishing returns. I figured I'd learned enough from the virtual machine, and going through another round of extensive troubleshooting just wasn't worth the effort.

x86 Specifics

I had a difficult time getting Android going on my Netbook, but that doesn't mean you will. In particular, the ASUS Eee PCs (701, 900, 901, 904 and 1000) are known to work fairly well under Android. Brock Tice has made a ready-to-go USB installer image for the Eee PCs, although the Wi-Fi support in his build doesn't work at the time of this writing. If you have a different model of Netbook (like my Acer Aspire One) your experience may vary. This is still very much proof-of-concept code.



Conclusion

My exploration of Android was a fun and educational exercise, even if it was very frustrating at times. Android proves that Linux has come a long way from the days of the Sharp Zaurus on mobile devices, and on the T-Mobile G1, it's smooth, fun and easy to use. The G1 definitely is a contender for top smartphone, and when you're competing with the likes of the iPhone and BlackBerry, that's saying a lot.

Android as an alternative to Windows Mobile holds a lot of promise. Its open architecture means it can take advantage of the Open Source movement to roll in new features, and it can give many current Windows users their first taste of how sweet Linux and open source can be. Unfortunately, it's not ready for prime time now, but that'll change as developers figure out the hardware and get driver support for the various handsets—much like Linux's desktop support has grown during the past few years. This is a moving target, and a lot can change quickly. I plan to keep an eye on this space for more news on Android on formerly Windows Mobile handsets.

Android on a Netbook, however, gives me pause. After spending some time with the VirtualBox virtual machine, I can't really see how this is better than, say, Ubuntu Netbook Remix or even regular Ubuntu. Android is optimized for small touchscreens and tiny CPUs. Most Netbooks have a lot more CPU than is required to run Android, yet they don't have the touchscreen. Android is designed for use with a minimal or onscreen keyboard, and Netbooks have full, functioning keyboards. After fiddling with it, the use case for Android on a Netbook just doesn't seem very compelling. I think there's more value in using a Netbook as a tiny, full-featured laptop than using it as a large mobile Internet tablet. However, don't let my opinion sway you. Go test it for yourself! You'll learn a lot, and if you have the skills to improve the port, contribute some fixes. The developers probably would appreciate the help, and you'd be contributing to something that could touch a lot of people.■

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.

Breaking News

Just before press time, a team of Android developers released a live CD for Android. If you want to test drive Android on your Netbook, laptop or even a virtual machine, it's extremely easy now. Simply download the image, burn it to CD, and boot from it—no compiling needed. Check it out—you may get hooked on it enough to buy a G1 phone!

Resources

The Open Handset Alliance: **www.openhandsetalliance.com/android_overview.html**

JF's Blog (Android Developer Extraordinaire): **jf.andblogs.net**

Hacking Your G1/Dream: **forum.xda-developers.com/showthread.php?t=442480**

Android for the HTC Touch Pro: **connect-utb.com/index.php?option=com_rokdownloads&view=folder&Itemid=68&id=3:htc-raphael**

Porting Android to x86: **code.google.com/p/patch-hosting-for-android-x86-support**

Setting Up Your Machine to Build Android: **source.android.com/download**

Getting the Patched x86 Version of Android: `svn checkout http://patch-hosting-for-android-x86-support.googlecode.com/svn/trunk/ patch-hosting-for-android-x86-support-read-only`

Brock Tice's Eee PC Build of Android: **virtuallyshocking.com/2008/12/20/building-android-for-the-asus-eeepc-701**

Innovative Interfaces with Clutter

Use Clutter to develop OpenGL applications with rich 2-D and 3-D interfaces that include object rotation, scaling, texturing and more. ALEX CRITS-CHRISTOPH

Meet one of the most revolutionary toolkits available for Linux: Clutter. Clutter is an OpenGL-based toolkit, described as an "open-source software library for creating fast, visually rich and animated graphical user interfaces". Clutter provides a simple API for powerful three-dimensional and two-dimensional manipulation. Creating interactive games, 3-D media and animated applications for Linux systems with Clutter is cleaner, easier and quicker than coding an OpenGL application with more conventional methods.

Clutter comes with many built-in tools and effects. Rendering object rotation, scale, texture and opacity are built right in and can be accomplished with a few lines of code. Rendering and controlling the GStreamer multimedia API also is easy with an additional library. There even are Webkit bindings, so manipulating Web pages in a Clutter program is simple.

Clutter has been used in many successful applications and open-source projects. Take, for example, the open-source Elisa media center. Developed by Fluendo, Elisa is a 3-D media center—one of the most sophisticated alternatives to software such as Windows Media Center available for Linux. Elisa makes use of Clutter's animation and 3-D API in its elegant interface.

The Ubuntu Mobile Internet Device Edition, developed by the Ubuntu Mobile community, also uses Clutter for its main user interface. Additionally, the Moblin Project plans to use Clutter in its software platform. Clutter's use is widespread across Linux systems and is becoming more and more popular every day.

Installing Clutter on Linux systems is extremely easy with the use of binary package managers. Install Clutter, the Cairo add-on, the GStreamer add-on and the Python bindings. Using your distribution's package manager, install the following packages: libClutter, libClutter-cairo, libClutter-gst and python-Clutter.

Different distributions will have different versions available, and it is recommended that you install the latest possible version. However, you need 0.8.0 or 0.8.2 of the libClutter packages to follow the examples and run the code given in this article. If the version number in your package manager is different from either 0.8.0 or 0.8.2, you should install Clutter from source. See Resources for the URL to Clutter's source files.

In this article, I'm using Clutter's Python bindings to work with Clutter. More can be done with Python in just a few lines of code, so using that language makes it easier to explore and understand Clutter. To test your install of Clutter, simply run Python and do the following:

```
import clutter
```

If you get a blank prompt back with no errors, the Clutter module was imported successfully, and you've installed python-Clutter correctly.

Now, let's start with a simple "Hello World!" Clutter application with Python. You probably should turn off any desktop effects or compositing window managers, such as Compiz Fusion. Most Linux video drivers will not allow multiple OpenGL or 3-D processes to run simultaneously with a compositing window manager, which includes Clutter, because of its 3-D capabilities.

To start the program, you need to import the Clutter module and define your main Class and an initialization function. Create a stage in the initialization function. The stage is the base of any Clutter interface. On the stage, objects called actors can be seen and manipulated. Clutter uses the term actors to describe any objects that exist on the stage.

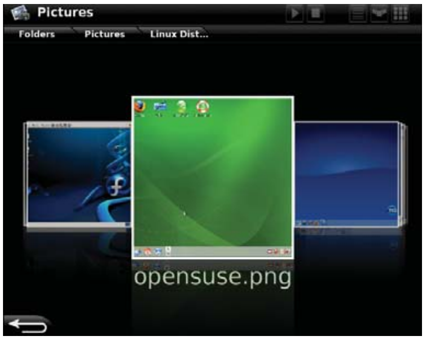


Figure 1. Elisa, Fluendo's media center application, utilizes Clutter's advanced library to add 3-D animation to its GUI.

The sample program is shown in Listing 1, and the output window is shown in Figure 2.

After creating the stage, set the stage's properties and some properties of the window containing it.

Set color of the stage to black by accessing Clutter's predefined colors using `Clutter.color_parse()`.

Next, set the size of the stage, which will set the size of the window. Also set the title of the window.

To show our "Hello world" message on the stage, you need to create an actor—in this case, a label. Set the font type of the label, the text to display, and the color of the label. Here, let's set the color manually rather than using a predefined color. Once the label is set, add the actor (the label) to the stage.

Labels work similarly to GTK+ widgets, but Clutter is *not* widget-based in the same way GTK+ is. Although both have similar functions and parts, Clutter contains only a handful of built-in "widgets", which are called actors. Clutter's actors are limited to rectangles, labels, images, video textures and a few other items.

To finish the example, tell the stage to

What’s Wrong with Compiz Fusion?

Compiz Fusion is an OpenGL compositing window manager, capable of delivering 3-D and smooth animation to the desktop. It is widespread on Linux desktops and is available on almost all modern Linux distributions. Often, Compiz Fusion is enabled by default.

However, Compiz Fusion does not play nice with Clutter. In fact, on almost all video cards, most OpenGL- or SDL-based applications are slow and prone to flickering if Compiz is running. Some of the programs affected include Google Earth, Blender and most 3-D games.

This is due to the X Window System’s inability to render OpenGL applications along with a compositing window manager, such as Compiz Fusion, simultaneously on most video cards. However, DRI2 aims to fix this problem. DRI2 (Direct Rendering Infrastructure 2) will allow several OpenGL applications to run at once by directly rendering redirected windows. In time, DRI2 will ship along with most X.Org video drivers.

With the implementation of DRI2 in most video drivers, the X Window System finally will be able to handle OpenGL with a compositing manager. However, currently on most video cards and using most drivers, Clutter conflicts with Compiz Fusion.

show all of its contents and call the main Clutter loop, which will display the interface. The last step is to tell Python to create an instance of your class.

Now, let’s take a look at a more useful Clutter program. Our program will use the Clutter GStreamer library to display and control a video file on the stage. Connect GStreamer’s video output to a video texture, which is then displayed on the stage and can be manipulated as an actor.

Listing 1. “Hello World” Using Clutter

```
import clutter

class HelloWorld:
    def __init__ (self):
        # Create stage and set its properties.
        self.stage = clutter.Stage()
        self.stage.set_color(clutter.color_parse('Black'))
        self.stage.set_size(500, 400)
        self.stage.set_title('Clutter Hello World')

        # Create label and set its properties.
        color = clutter.Color(0xff, 0xcc, 0xcc, 0xdd)
        hello = clutter.Label()
        hello.set_font_name('Mono 32')
        hello.set_text("Hello There!")
        hello.set_color(color)
        hello.set_position(100, 200)

        # Add label to stage.
        self.stage.add(hello)

        # Start main clutter loop.
        self.stage.show_all()
        clutter.main()


# Run program.
main = HelloWorld()
```

There are three main actors on the stage: the play button, the pause button and the GStreamer video. When the pause button is pressed, the video pauses, and it will continue playing when the play button is pressed. The program is shown in Listing 2, and the window is shown in Figure 3.

After the initial setup, define a signal for responding to mouse clicks. Clutter uses signals to respond to events in the same way GTK+ does. The signal dictates that when any button on the mouse is clicked, the specified function is called.

Next, create the buttons by creating rectangle actors for the button shape and text actors for the button text. Remember that Clutter is not widget-based, and there are no default button widgets as part of the API.

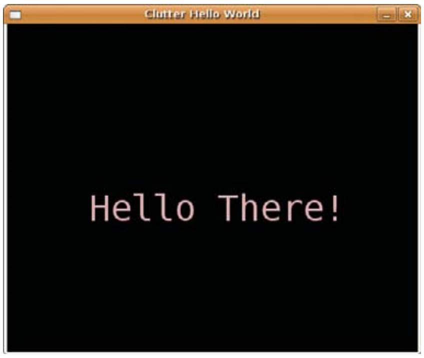
To create the video, you need a video texture. A video texture is a physical plane on which GStreamer can display the video. The video texture can be manipulated on the stage like an ordinary actor.

You also need two other GStreamer elements to play the video: a playbin and a pipeline. Clutter-Gstreamer will play any file type that GStreamer can play. After creating the playbin, set the location of the video to play and add the playbin to the pipeline.

Then, set the position of the video texture, add it to the stage and tell GStreamer to start playing the video.

Finally, create the mouseClick function. This function is called when the mouse is clicked anywhere on the stage. Check to see if the left mouse button was clicked inside one of the buttons and if it was, change the size and color of the buttons to give visual feedback of the click. Tell GStreamer to start or stop the video depending on which button was pressed.

An important feature of Clutter is its animation API. Animating any actors



Hello There!

Figure 2. The simple “Hello World” Clutter Program Running on Ubuntu

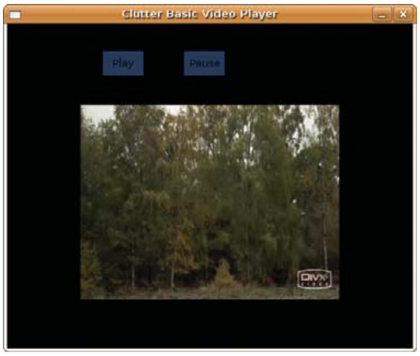


Figure 3. Clutter Video Player Playing the Monty Python Skit "How Not to Be Seen"

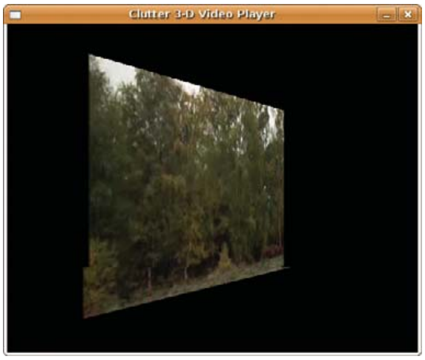


Figure 4. The 3-D Clutter Video Player, Partway through Rotating the Video

on the stage is easy with Clutter. Using the animation API, you can add smooth animations and effects to your Clutter application.

In the next example, let's take the GStreamer video texture and manipulate it in three dimensions. The GStreamer texture is rotating on the y-axis constantly.

The program is shown in Listing 3, and the window is shown in Figure 4.

After the initial setup, create a timeline. Timelines are used in Clutter to control animation and time events. The example timeline lasts for 100 frames at ten frames per second, and the timeline is set to loop forever.

Next create an alpha (see The Alpha Functions sidebar) for the animation, assign it to your timeline and give it a smooth step decreasing function. In simplest terms, the alpha is used to control the speed of the animation. The smooth step function causes the animation to speed up, then slow down, come to a halt, and then start up again. Clutter has several functions built in that you can use with the alpha, including sine, exponential and ramp functions.

Next, define the Rotation behavior the animation uses. Clutter uses behavior effects to describe animations. The Opacity behavior, for example, can change the visual alpha of an actor, making it transparent or opaque. Other behaviors include Scale, Path, Depth, B-Spline and Ellipse.

In this example, we tell our Rotation behavior to rotate over the x-axis, rotate clockwise, start rotating at an angle of zero, end at 360, and finally, tell it to use the alpha created earlier, respectively. After that, the rotational center, or the point the actor rotates around, is set to the approximate center of the GStreamer texture and the Rotation behaviour is applied to the video texture.

In addition to the normal startup steps, the timeline must be started.

Hopefully, you've learned a good deal about how Clutter works, and you can start developing and programming using the Clutter API. Using just the features you've seen here, you'll be able to create any interface that uses text, buttons,

Listing 2. Clutter-Based Video Player

```
import clutter
import gst
from clutter import cluttergst

class HelloWorld:
    def __init__ (self):
        # Create stage and set its properties.
        self.stage = clutter.Stage()
        self.stage.set_color(clutter.color_parse('Black'))
        self.stage.set_size(500, 400)
        self.stage.set_title('Clutter Basic Video Player')

        # Create signal for handling mouse clicks.
        self.stage.connect('button-press-event', self.mouseClick)

        # Create play button shape.
        self.playBtn = clutter.Rectangle()
        self.playBtn.set_color(clutter.Color(66, 99, 150, 0x99))
        self.playBtn.set_size(50, 30)
        self.playBtn.set_position(118, 34)
        self.stage.add(self.playBtn)

        # Create play button text
        # and overlay the rectangle.
        playTxt = clutter.Label()
        playTxt.set_text("Play")
        playTxt.set_color(clutter.color_parse('Black'))
        playTxt.set_position(130, 40)
        self.stage.add(playTxt)

                # Same for stop button.
        self.stopBtn = clutter.Rectangle()
        self.stopBtn.set_color(clutter.Color(66, 99, 150, 0x99))
        self.stopBtn.set_size(50, 30)
        self.stopBtn.set_position(218, 34)
        self.stage.add(self.stopBtn)

        StopTxt = clutter.Label()
        StopTxt.set_text("Pause")
        StopTxt.set_color(clutter.color_parse('Black'))
        StopTxt.set_position(225, 40)
        self.stage.add(StopTxt)

        # Create video texture.
        video_tex = cluttergst.VideoTexture()

        self.pipeline = gst.Pipeline("mypipe")
        playbin = video_tex.get_playbin()

        # Specify video file to play.
        movfile = "file:///home/user/Videos/Video.mov"
        playbin.set_property('uri', movfile)

        # Add to playbin to the pipeline.
        self.pipeline.add(playbin)

        # Set position and start playing the video.
        video_tex.set_position(90,100)
        self.stage.add(video_tex)
        self.pipeline.set_state(gst.STATE_PLAYING)

        self.stage.show_all()
        clutter.main()

    def mouseClick (self, stage, event):
        # Mouse click function, called when the moused
        # is clicked *anywhere* on the stage, we check
        # the mouse coordinates manually to see if the
        # click occurred inside a button.

        # Check for left mouse button.
        if event.button == 1:

            # Check to see if stop button was pressed.
            if event.x > 218 and event.x < 268 and \
               event.y > 34  and event.y < 64:


self.stopBtn.set_color(clutter.Color(33,50,150,0x89))
self.playBtn.set_color(clutter.Color(66,99,150,0x99))
                  self.stopBtn.set_size(49, 29)
                  self.playBtn.set_size(50, 30)
self.pipeline.set_state(gst.STATE_PAUSED)

            # Check to see if the play button was pressed.
            if event.x > 118 and event.x < 168 and \
               event.y > 34  and event.y < 64:


self.playBtn.set_color(clutter.Color(33,50,150,0x89))
self.stopBtn.set_color(clutter.Color(66,99,150,0x99))
                 self.playBtn.set_size(49, 29)
                 self.stopBtn.set_size(50, 30)
self.pipeline.set_state(gst.STATE_PLAYING)


# Run program.
main = HelloWorld()
```

images and video with Clutter. Of course, after learning the basics, the more advanced UI elements will become easier to understand and work with.

In the future, the Clutter developers will continue to improve and update the API, and many new improvements are expected in the Clutter 1.0 release. You can learn more about the Clutter development process from the Web site (see Resources). Clutter is going to power many innovative open-source applications in the future.■

The Alpha Functions

At first, especially to those who've forgotten their Calculus and Algebra, the alpha functions may seem unpredictable or confusing. There is a large list of the number of available functions: exp_dec_func, exp_inc_func, ramp_dec_func, ramp_func, ramp_inc_func, sine_dec_func, sine_func, sine_half_func, sine_inc_func, smoothstep_dec_func, smoothstep_inc_func and square_func. Here's a brief explanation of each type:

- Exponential functions: depending on whether you're using a decaying function or an increasing function, exponential functions make the animation speed up or slow down at an exponential rate.
- Ramp functions: ramp functions animate at a constant speed. However, the full ramp function animates at both a negative and a positive constant speed by switching directions.
- Sine functions: sine functions make the animation reverse. Like the graph of a sine function, the animation would speed up, slow down, change directions, speed up in the reverse direction, and then slow down again.
- Smooth step functions: the smooth step function works logistically. It starts slowly, then quickly increases and finally slows down toward the end of the animation.
- Square functions: square functions follow a step pattern, which results in quick changes between two constant animation speeds.

Resources

Clutter Home Page: **clutter-project.org**

Elisa Project Page: **elisa.fluendo.com**

Clutter Source Files: **www.clutter-project.org/sources**

Listing 3. Clutter Rotating a Video

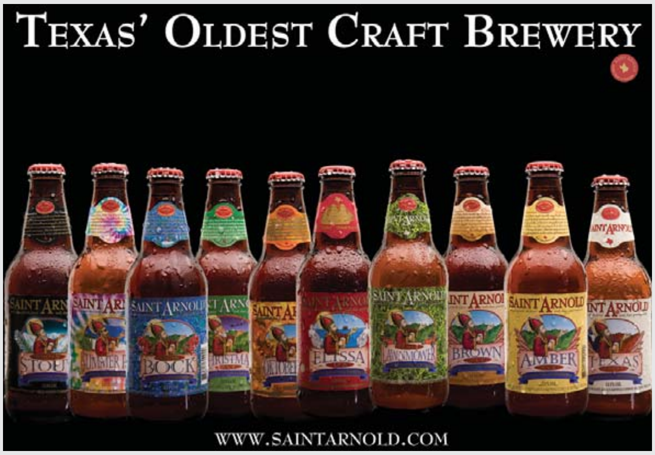
```
import clutter
import gst
from clutter import cluttergst

class HelloWorld:
    def __init__ (self):
        self.stage = clutter.Stage()
        self.stage.set_color(clutter.color_parse('Black'))
        self.stage.set_size(500, 400)
        self.stage.set_title('Clutter 3-D Video Player')

        # Setup video.
        video_tex = cluttergst.VideoTexture()
        self.pipeline = gst.Pipeline("mypipe")
        playbin = video_tex.get_playbin()
        movfile = "file:///home/user/Videos/Video.mov"
        playbin.set_property('uri', movfile)
        self.pipeline.add(playbin)
        video_tex.set_position(90,80)
        self.stage.add(video_tex)
        self.pipeline.set_state(gst.STATE_PLAYING)

        # Create timeline that lasts for 100 frames
        # at ten frames per second.
        timeline = clutter.Timeline(100, 10)

        # Set timeline to loop forever.
        timeline.set_loop(True)

        # Create an alpha.
        alpha = clutter.Alpha(timeline, clutter.smoothstep_dec_func)

        # Set up rotation.
        Rotation = clutter.BehaviourRotate(
                                  axis=clutter.Y_AXIS,
                                  direction=clutter.ROTATE_CW,
                                  angle_start=0,
                                  angle_end=360,
                                  alpha=alpha)
        Rotation.set_center(160, 160, 0)
        Rotation.apply(video_tex)

        # Start it all up.
        timeline.start()
        self.stage.show_all()
        clutter.main()


# Run program.
main = HelloWorld()
```

Alex Crits-Christoph has been working with Linux for some time now. He enjoys developing and designing open-source graphical user interfaces.

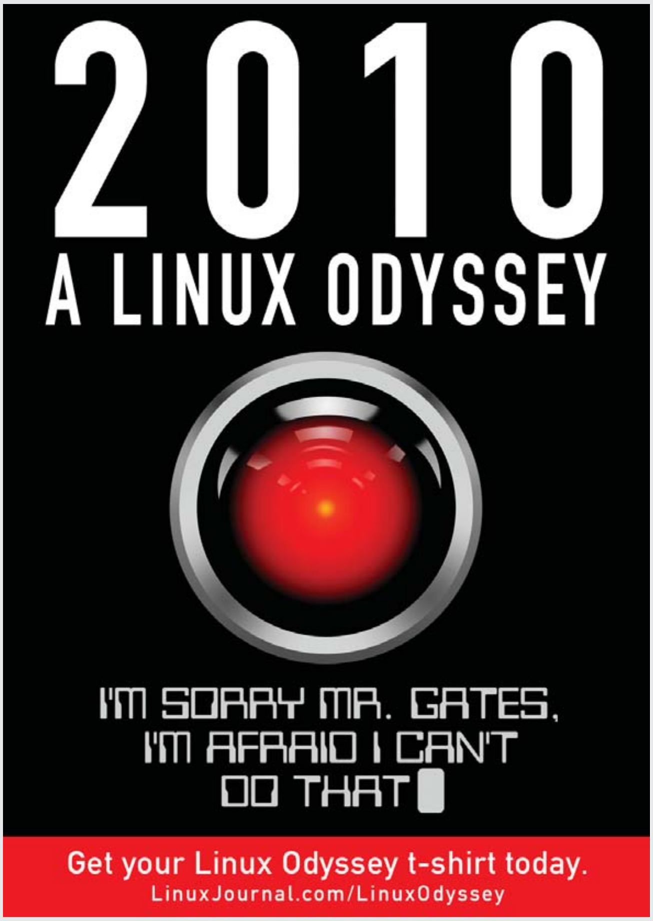
INNOVATION ON THE GO
ORDER YOUR BEAGLE BOARD FROM DIGIKEY.COM
AVAILABLE EXCLUSIVELY AT DIGI-KEY
beagleboard
only $149.00
LOW-COST, NO FAN, SINGLE-BOARD COMPUTER
Digi-Key® CORPORATION
www.digikey.com

Archie McPhee
Supplying the world with impractical weirdness for over 25 years!
mcphee.com

GET NOTICED!
With custom, temporary tattoos! Any image!
5% Off With This Code:
linuxjournal
Great for conventions!
StrayTats.com


TEXAS' OLDEST CRAFT BREWERY
WWW.SAINTARNOLD.COM

American made Utility Kilts for Everyday Wear
UTILIKILTS.com


2010
A LINUX ODYSSEY
I'M SORRY MR. GATES,
I'M AFRAID I CAN'T
DO THAT
Get your Linux Odyssey t-shirt today.
LinuxJournal.com/LinuxOdyssey

Turning the Internet Outside In

Let's hack an open Internet, starting at home.

DOC SEARLS

You can only hack what's hackable. We owe Linux to the fact that operating systems are hackable, and that they can run on common hardware, much of which is also hackable. We also owe Linux to the Internet, which is a hack on wiring and data trafficking.

For PCs and mobile devices, Linux is a defaulted choice. It's at GandhiCon 4. That and the first three GandhiCons are implicit in the Mohandas Gandhi quote, "First they ignore you, then they laugh at you, then they fight you, then you win."

The Internet, however, is another matter. The Internet Protocol (IP) arrived at GandhiCon 4 by 1981 (with IPv4). That's because it was created as what we might call a public protocol, connecting devices using just about any kind of network wiring, hardware and data link protocols (Ethernet, Token Ring, FDDI and so on), without prejudice. This made it easy and cheap for anybody to use.

By design, the Internet Protocol was decentralized. It reduced network complexity inside the network as far as possible, while relying on intelligence at its end nodes. It was even agnostic toward addressing schemes, leaving choices up to implementations at higher levels in the stack and resolution up to the Address Resolution Protocol (ARP).

Alas, what most people know best about the Internet is not its decentralized, depoliticized and free (as in both freedom and beer) public nature, but rather its centralized, politicized and costly (as in both freedom and beer) private one. This is the Internet of domain names that are privately owned (actually, rented), controlled by a central naming authority (the Internet Corporation for Assigned Names and Numbers, or ICANN) and filled with "pipes" mostly owned by private interests and highly cartelized. This is not an Internet to which we can simply connect. Instead, it's one we can "access" only through Internet Service Providers—a class of businesses that was born when small independent companies found ways to make the Internet available to anybody with a land line and has since become the tertiary service of phone and cable companies selling "broadband" or "high-speed Internet" as the third act in a "triple play".

Although the former Internet is hackable, the latter one is not.

As it happens, I've been living in the hackproof hell of the private, centralized Internet for the last two weeks, during which time my home connection here in Santa Barbara has been intermittently plagued by high latencies and packet losses. My ping and traceroute tests clearly isolate the problem somewhere between my cable modem and the first IP address my packets encounter: a gateway downtown that's also owned by the cable company. Cable company technicians that have come to my house (four so far) have excused from blame my cable modem and all wiring between it and the service pole. They know the problem is somewhere in their system. They still have not solved it, and neither can I, even with help from many friends far geekier than myself.

So here is a radical proposition. Let's build the Internet we want—a free, open and hackable Internet—from the outside in.

This is something Bob Frankston has been advocating for many years. What Bob wants is simple connectivity between any points floating on the vast resource he calls our "sea of bits". His latest label for this is "ambient connectivity". In his essay "Opportunity for Innovation", Bob writes, "Once we can assume connectivity we can start taking advantage of the opportunities. It's not just about high-value applications like education, commerce and entertainment. It's about basic infrastructure. We won't discover the real value until we've had a chance to experience ambient connectivity." In a follow-up essay titled "Zero Marginal Cost", he adds:

> The idea that we can create our own solutions using raw, unreliable bits is at the heart of the Internet's generativity....
>
> We've already seen the power of zero marginal cost. It was the availability of unmeasured local phone service that gave the United States the lead in adopting the Internet in the 1990s. We rejected digital phone service because the phone companies chose to charge a premium for that service. We just worked around it using modems because there was zero marginal cost for using the existing infrastructure.

Bob's model of the Internet is home networking, expanded outward through converging communities. In my interview with Bob for the March 2008 issue of *Linux Journal* (**www.linuxjournal.com/article/10033**), he said, "The networks in our homes are a good example. You 'just' print without worry about negotiating for the printing provider."

As it happens, I'm also shopping for home networking gear—in particular, for a router/switch to connect the 16 Ethernet jacks scattered about the house. Cat-6 wiring runs from each of those jacks to a patch panel in a wiring closet. The cable company's modem is in there too.

Lemme tell ya, if there's a category ripe for disruption, it's home networking. I've been looking at Belkin, Cisco/Linksys, D-Link, Netgear and others—none of which are especially helpful. The 8-port device I'm replacing is a Netgear router/switch that was billed as a "VPN Firewall" but failed at the essentials: its gears were stripped by the cable company's new 20Mb downstream data speeds.

So let's look at making the Net hackable from the outside in. VCs always are asking about market size and "pain points" in need of relief. I can't think of a bigger, or more ideally hackable, pain than the one we find right at home.■

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.