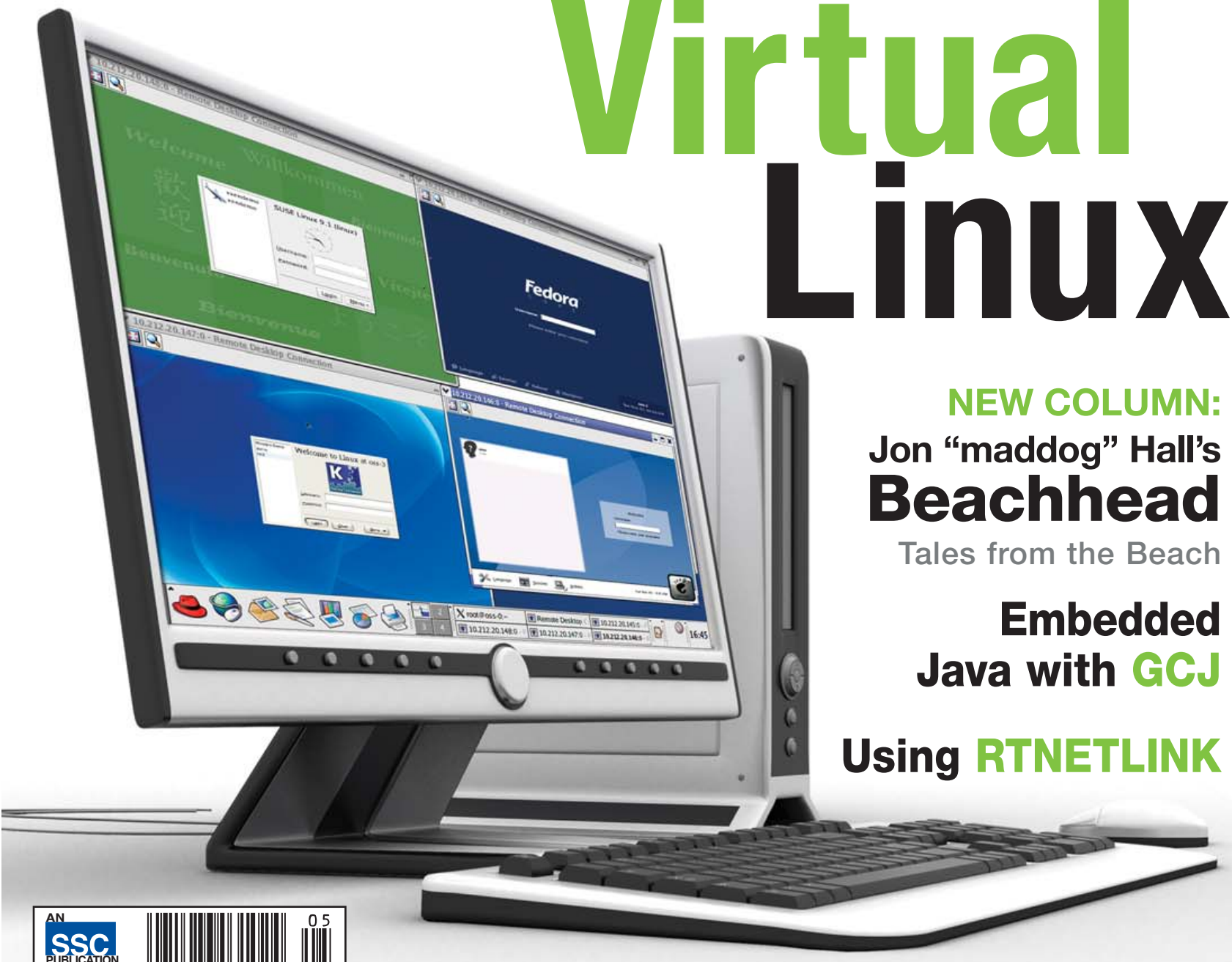
Jon “maddog” Hall | GPL 3 | GCJ | VMware | QEMU | UML | Xen | Wine

LINUX JOURNAL

Since 1994: The Original Magazine of the Linux Community
MAY 2006 | www.linuxjournal.com

REVIEWED VMware 5.5

Virtual Linux



NEW COLUMN:
Jon “maddog” Hall’s
Beachhead
Tales from the Beach

Embedded Java with GCJ

Using RTNETLINK

AN SSC PUBLICATION
USA $5.00
CAN $6.50
0 71486 03102 4 05

PLUS **Is Linux VoIP Ready?**

Let's go Xtreme

Introducing Appro *Xtreme*Servers & Workstation with 8 DIMM Sockets per CPU

- 2-way or 4-way, Single or Dual-Core AMD Opteron™ processors
- **Largest memory capacity - 8 DIMM Sockets per CPU, up to 128GB**
- PCI-Express technology to increase I/O bandwidth and reduce system latency
- Outstanding Remote Management – IPMI 2.0 compliant
- Cable-free design, ready to run, simple to install, service and maintain
- Support for Windows® or Linux OS
- Ideal for memory-intensive and I/O-intensive applications

1U / 2U / 3U Servers and Workstation

AMD Opteron™ Processors
- AMD64 dual-core technology reduces memory latency and increases data throughput
- Dual-core processors with Direct Connect Architecture deliver the best performance per watt with little or no increase in power consumption or heat dissipation.

APPRO
HPC Cluster Solutions

Appro delivers high-performance computing solutions to help you maximize productivity for a solid ROI. On-site maintenance and installation services are also available.

For more information, please visit www.appro.com
or call Appro Sales at 800.927.5464, 408.941.8100.

AMD 64 Opteron™

LOST

in a sea of server problems?

Get Rescued Today with the AlterPath™ OnBoard!

Download a FREE White Paper on Service Processor Management at
www.cyclades.com/spm

The Next Big Innovation in Out-of-Band Management

Remote management of servers and other IT assets is critical in today's increasingly complex data centers. Service processor management (HP iLO, IPMI, Dell DRAC, IBM RSA, Sun ALOM) as part of a comprehensive out-of-band infrastructure (OOBI) solution helps enterprises streamline the management of servers from multiple vendors and reduces operating expenses.

The AlterPath OnBoard is a service processor manager that consolidates IP connections, user access and server health information from multiple servers to provide secure, efficient rack-level management with seamless integration into the OOBI.

www.cyclades.com/lja
1.888.cyclades · sales@cyclades.com

cyclades

CONTENTS

MAY 2006
Issue 145

ON THE COVER: IMAGE BY AYAAZ RATTANSI

FEATURES

ON THE COVER

The competition doesn't stand a chance.

If you base deployment decisions on performance and price, Coyote Point's for you. We've cornered that market.

To prove it we asked The Tolly Group to evaluate our E350si application traffic manager against the competition. The results speak for themselves.

Throughput? Almost 40% more than others in our space. Cost of transactions per second? Up to four times less. Connection rate? In some cases, one-sixth the cost. One-sixth! And we're told Coyote Point is the #1 choice for today's open source networks.

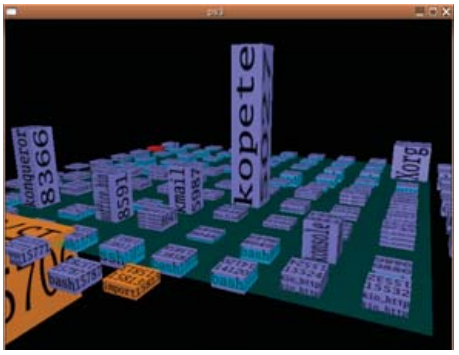
But don't just take our word for it. Get the facts. Call 1.877.367.2696 or write info@coyotepoint.com for your free copy of the full Tolly Report.

COYOTE POINT SYSTEMS INC.

www.coyotepoint.com

CONTENTS MAY 2006 Issue 145

COLUMNS



REVIEW

INDEPTH

IN EVERY ISSUE

80 IS LINUX VOICE OVER IP READY?

Next Month

STORAGE

Next month, we're going fishing. Why not? Did you know that you can use fish:// for secure access to files over a network? In fact, we'll tell you about a number of kioslaves that KDE uses to make network storage transparency a breeze. While you're at it, you may want to dump your reliance on NFS or Windows (Samba) shares in favor of SSHFS, a user-mode filesystem that works over the secure shell, SSH.

Also, *Linux Journal* tells you about installing Yellow Dog Linux on an iPod, security features in Red Hat Enterprise 4 and recovering RAID and LVM2 volumes. We'll introduce you to Gambas, tell you how to set up and use Tripwire and how to tap in to Linux kernel hotplug events.

TYAN

Power of 4+4

TYAN Matchless 8-Processor Server

M4881
Processor Expansion board for S4881

- Up to four AMD Opteron™ processor MP
- Up to 64GB of Registered (ECC) DDR400/333 166 DIMM sockets
- 12"x13" dimension with special architecture
- Two Hyper TransportTM v1.0 Connectors

Thunder S4881
8-Processor AMD Opteron™ Motherboard

- Up to eight or four AMD Opteron™ processor
- Up to 64GB of Registered (ECC) DDR400/333
- (4)SATA support RAID 0,1, 0+1
- Broadcom® BCM5704C Dual-Channel GbE controller
- IEEE 1394a (FireWire) ports
- ATI® RAGE™ XL with 8MB

Transport VX50 B4881
8-Processor AMD Opteron™ 5U Server

- 4/8-Processor HPC Computing Platform
- Support up to eight (8) AMD Opteron™ 8xx series
- 4-P model (B4881-4P): Sixteen (16) DIMMs,supporting max. 64GB registered, ECC DDR400/333 memory
- 8-P model (B4881-8P): Thirty-two (32) DIMMs,supporting max. 128GB registered, ECC DDR400/333 memory
- Two (2) gigabit Ethernet LANs, Broadcom® BCM5704 controller
- Three (3)64-bit, 133/100/66MHz PCI-X slots , Two (2) PCI-E slots (1*x16 + 1*x4 signal) Total 5 usable PCI expansion slots

TYAN COMPUTER CORP.

Tyan Computer USA
3288 Laurelview Court
Fremont, CA 94538 USA
Tel: +1-510-651-8868 Fax: +1-510-651-7688
Pre-Sales Tel: +1-510-651-8868 x5120
Email: marketing@tyan.com

For more information about this and other Tyan products, please contact Tyan Pre-Sales at (510) 651-8868 x5120, or contact your local Tyan system integrator/reseller.

www.tyan.com

LINUX JOURNAL

Editor in Chief
Nick Petreley, ljeditor@ssc.com

Executive Editor	Jill Franklin jill@ssc.com
Senior Editor	Doc Searls doc@ssc.com
Web Editor	Heather Mead heather@ssc.com
Art Director	Garrick Antikajian garrick@ssc.com
Editor Emeritus	Don Marti dmarti@ssc.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
Products Editor	James Turner newproducts@ssc.com

Contributing Editors
David A. Bandel • Greg Kroah-Hartman • Ibrahim Haddad • Robert Love • Zack Brown • Dave Phillips • Marco Fioretti • Ludovic Marcotte • Paul Barry • Paul McKenney

Proofreader	Geri Gale
VP of Sales and Marketing	Carlie Fairchild carlie@ssc.com
Marketing Manager	Rebecca Cassity rebecca@ssc.com
International Market Analyst	James Gray jgray@ssc.com
Sales Coordinator	Lana Newlander ads@ssc.com

Regional Advertising Sales
NORTHERN USA: Joseph Krack, +1 866-423-7722 (toll-free)
EASTERN USA: Martin Seto, +1 416-907-6562
SOUTHERN USA: Laura Whiteman, +1 206-782-7733 x119
INTERNATIONAL: Annie Tiemann, +1 866-965-6646 (toll-free)

Advertising Inquiries ads@ssc.com

Publisher	Phil Hughes phil@ssc.com
Accountant	Candy Beauchamp acct@ssc.com

Linux Journal **is published by, and is a registered trade name of, SSC Media Corp.**
PO Box 980985, Houston, TX 77253-3587 USA

Subscriptions
E-MAIL: subs@ssc.com
URL: www.linuxjournal.com
PHONE: +1 713-589-3503
FAX: +1 713-589-2677
TOLL-FREE: 1-888-66-LINUX
MAIL: PO Box 980985, Houston, TX 77253-3587 USA
Please allow 4–6 weeks for processing address changes and orders
PRINTED IN USA

USPS *LINUX JOURNAL* (ISSN 1075-3583) is published monthly by SSC Media Corp., 2825 NW Market Street #208, Seattle, WA 98107. Periodicals postage paid at Seattle, Washington and at additional mailing offices. Cover price is $5 US. Subscription rate is $25/year in the United States, $32 in Canada and Mexico, $62 elsewhere. POSTMASTER: Please send address changes to *Linux Journal*, PO Box 980985, Houston, TX 77253-3587. Subscriptions start with the next issue.

routerOS: routerBOARD.com

User Management

- support more than 3000 PPPoE or HotSpot clients
  - full RADIUS support for user parameters
    - tx/rx speed, address, filter rules
    - supports RADIUS real time modification of parameters while users are online
- Peer to Peer protocol control (P2P)
  - per client P2P tx/rx rules with burst support
  - P2P pool
  - complete blocking of P2P

Wireless AP and Backbone

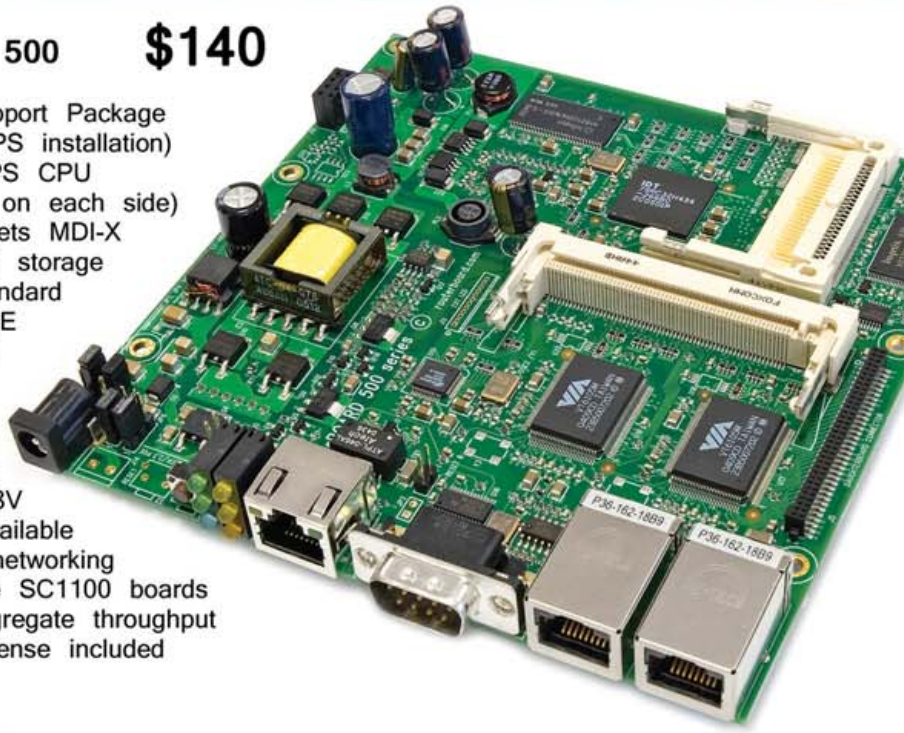
- Wireless monitoring
  - Frequency scanning with detailed report
  - Raw wireless packet sniffer
    - streaming option to Ethereal analyzer
    - option to save to a file format supported by Ethereal
  - Snooper packet inspection
    - analyzes all raw frames received for wireless parameters
    - monitors a single channel or all channels

- Nstreme wireless polling protocol
  - no decrease in speed over long distances (as seen with the 802.11 ACK packet bottleneck)
  - polling improves speed and eliminates contention for access to the wireless bandwidth
  - access point control over Nstreme clients tx data to optimize use of the wireless medium
  - RADIUS support for the access control list including bandwidth settings for wireless clients

- Full 802.11a/b/g support

The above is a brief description of a few features, for more information and a fully featured 24 hour demo go to:

RouterBOARD 500 $140



- Linux Board Support Package (full Debian MIPS installation)
- 266-400MHz MIPS CPU
- 2 miniPCI (one on each side)
- 3 10/100 Ethernets MDI-X
- 64/128MB NAND storage
- PoE 802.3af standard and passive PoE (also 12V PoE)
- Compact Flash
- Low power
- 32MB DDR (64MB optional)
- 6-24V and 25-48V power mode available
- 2-3x faster for networking than the Geode SC1100 boards
- 200-300Mb/s aggregate throughput
- L3 RouterOS license included

$95

- MDI-X
- 4 separate 10/100 ports

RouterBOARD 44

For the Router Builder !

- rackmount servers and routers
- up to 24 Ethernet ports in a PC
- no more straight/cross cable problems
- server quality VIA VT6105 chips

$195

RouterBOARD 230

No feature left behind !

Integrated router with various interfaces. Use as an AP on a tower with up to 500ft PoE. Includes IDE/CF, miniPCI, USB, PCMCIA, UART, PCI, GPIO, LCD controller, Linux SDK, and more.

$120 $65

Eight ports Four ports

RouterBOARD 11/14/18

Multi radio tower !

MiniPCI to PCI adapters for multi radio system. Tested with sixteen radios in one Router/AP.

$240

- PoE
- 10-56V input
- 9x 10/100
- 6x mPCI

RouterBOARD 500 & RouterBOARD 564

The Wireless Switchboard !

For a complete multi-radio tower system, the RouterBOARD 500 can carry a daughterboard (RouterBOARD 564) which adds six ethernets and four miniPCI.

contact sales@routerboard.com or go to www.routerboard.com

YOUR HIGH PERFORMANCE COMPUTING SOLUTION HAS ARRIVED.

VXRACK™ with the Intel® Xeon™ processor helps you simplify computing operations, accelerate performance and accomplish more in less time.

1 Choose one of the 3 convenient rack sizes

VXR-72
Rack accomodating up to
72 VXBlades/**144 Processors**
27TB of aggregated **Storage**
864GB of Global Memory
Power Distribution Included
Patented Architecture
Advanced Cooling System
Integrated InfiniBand Cable Mgnt.

VXR-96
Rack accomodating up to
96 VXBlades/**192 Processors**
36TB of aggregated **Storage**
1.15TB of Global Memory
Power Distribution Included
Patented Architecture
Advanced Cooling System
Integrated InfiniBand Cable Mgnt.

CIARA TECHNOLOGIES

For more Information call
or visit us at

VXB-7230NH
Intel SE7230NH Motherboard
800MHz Front Side Bus
Intel® Pentium® D 3.2GHz
1GB DDR2 533 Memory
Single 80GB 7200RPM SATA Drive
One **PCI/Express Slot Available**
Dual 10/100/1000 Intel Lan Port
350W Power Supply

VXB-7520J
Intel SE7520J Motherboard
800MHz Front Side Bus
2 x Intel® EM64T Xeon™ 3.2GHz
2GB DDR2 400 ECC Reg.Mem
Single 40GB 7200RPM ATA Drive
One **PCI/Express Slot Available**
Dual 10/100/1000 Intel Lan Port
500W Power Supply

2 Choose one or more type of VXBlade

3 Add, Mutiply,That's it. Easy as 1, 2, 3...

For example you choose the following: One VXR-96 with 48 Dual Intel® EM64T Xeon™ and 40 Single Intel® Pentium®4. You take 1 (VXR-96) + 48 (VXB-7520J) + 40 (VXB-7221B)...That's it

VXRACK

THE FUTURE OF CLUSTER TECHNOLOGY

CIARA TECHNOLOGIES...A GLOBAL SOLUTION PROVIDER.

Ciara Technologies is a world-class computer systems manufacturer. Ciara designs, develops, manufactures, markets, services, and supports a variety of computer systems including graphic workstations, rackmount and tower servers, networked storage and the newly acclaimed VXRACK™ Cluster Technology. The company's state of the art supercomputer cluster is based on the Intel IA32 and IA64 architectures and utilizes Linux operating systems. We are proud to be recognized by Intel as an "Intel Premier Provider". Choosing Ciara is choosing a single point of contact for all your IT requirements. All our products are built under the ISO 9001 standards and regulations. The growth of Ciara enabled the company to move its 300+ employees,in February 2003, to an ultra-modern plant of 576,000 sqft.. Ciara now has the capability of producing more than 500,000 units per year.

866-7VX-RACK (866-789-7225)
WWW.VXRACK.COM

intel Xeon® inside™

letters

Kudos on the Redesign

I received my copy of the March 2006 issue of *LJ*, and I was surprised to see the graphic design. I think the new design is awesome. Thanks for your work.

—

Stefano

Thank our artist extraordinaire, Garrick Antikajian, for the spectacular new design!—Ed.

Separation of Church and Ads

I'm not so sure I like the new look for *LJ*. It's difficult to tell where the ads end and the articles begin. I feel that it acts as a distraction to the reader and hides valuable information among the noise of advertising.

—

William W. Atkinson

Blackjack in Fewer Draws

I just read the "Writing a Shell Game" by Dave Taylor in the March 2006 issue of *Linux Journal*, and the technique he used for shuffling cards seems somewhat inefficient and incorrect. Taylor is randomly selecting cards from a fixed array. A far better algorithm would be a swap-based shuffle, where you walk along the deck, swapping each card with a randomly chosen other card. The code to implement this is shorter than Taylor's "choose until bored, then scoop up the rest" method. In bash, you could do it with:

```
# Shuffle the deck
i=52
while [ $i -gt 0 ]; do
    swap_i=$(( ( $RANDOM % $i ) + 1 ))
    swap=${deck[$swap_i]}
    deck[$swap_i]=${deck[$i]}
    deck[$i]=$swap
    i=$(( $i - 1 ))
done
```

No need for creating a second deck from a first—this shuffles in-place. Still, I'd like to thank Dave for the article, because although I know I've encountered bash arrays before, somehow I just could never get the hang of the syntax. His article spells it out in an understandable and useful way.

—

Steve Fink

Dave Taylor replies: *Hey! That's some cool code you've written there, Steve. Thanks for sharing it. In terms of whether my algorithm was optimal, well, um, err, I've been too busy figuring out whether we were going to implement Atlantic City or Vegas rules to worry about how well the shuffle worked.*

Substantiate Your Dislikes about GNOME

I'm a KDE refugee and much prefer GNOME, so I was surprised at the depth of [Nicholas Petreley's] dislike [see the March 2006 etc/rant column]. It'd be nice to know what *objective* standards he's measuring GNOME against. I'm not suggesting that GNOME is "the best"—for example, I too dislike Nautilus and usually install another file manager.

—

Sonia Hamilton

I will be more specific in a future rant. In the meantime, as far as what a user sees, GNOME is a window manager, panels and file manager. The minimalistic window manager, Metacity, doesn't have much to do with GNOME. The panels get more minimalistic with each release. The only substantial component of GNOME is Nautilus, and you replace it. What does that tell you?—Ed.

What a GNOME Needs

I think the lesson to be learned from both open-source and commercial software is that everyone has different needs. [GNOME should come up with a method] that would allow people to choose just what level of features are available to users, what the defaults are and so on. As distro makers get a better sense of the varying classes of users, they can continue to fine-tune their policies without having to be full GUI developers. And as users themselves grow, they can be given the option of moving to a more advanced GUI policy. This would be an elegant way of satisfying everyone's needs. And it keeps in line with what I think of as real choice in the traditional UNIX sense.

—

Danny

GNOME Does Not Offer Choice

The only real complaint I have against GNOME is that it doesn't support no-auto-raise-on-focus. You Google it, and you find letters that are six or seven years old asking when it will be available in GNOME, with the GNOME developers pretty much ignoring the request or saying, "never". So to the people who say that GNOME offers choice, ask them why you can't choose to have the focus in a window that is partially occluded? If you get the Microsoftian response of "Why would you want that?", then you have demonstrated that they really don't care about choice. And here I find myself at the end of the letter, apparently disagreeing with Nick! Choice does matter, and GNOME bites precisely because it fails to offer it.

—

Bill

My sentiments exactly. As per the previous letter, I'd like to see a desktop that "just works" but makes it easy to customize it to the Nth degree if I so choose.—Ed.


intel
Leap ahead™
Remember when
the sky was the limit?
With Intel® Software Development Products, the Swinburne Center for Astrophysics is showing today's kids a universe filled with unlimited possibilities.
intel
Software
HPC APPLICATIONS HELP UNRAVEL THE ORIGINS OF THE UNIVERSE.
Swinburne University's Center for Astrophysics and Supercomputing in Melbourne, Australia, is helping develop the next generation radio telescope in order to collect enough data to perform modeling and simulations of our entire galaxy. Their goal is to make realistic, 3D, virtual-reality animations available to the general public, particularly school children. To do this, they used the Intel® compilers to deliver application performance improvements — saving them valuable development time and money[1]. Whether you build applications for physics or financial analysis, Intel® Software Development Products help your HPC applications reach for the stars.
Visit www.intel.com/software/products for more information.
Copyright © Intel Corporation, 2006. All rights reserved. Intel, the Intel logo, are trademarks of Intel Corporation or its subsidiaries in the United States and other countries. *Other names and brands may be claimed as the property of others. [1] Performance and benchmark information were provided by Swinburne University. Buyers should consult other sources of information to evaluate the performance of systems or components they are considering purchasing.

In Defense of GNOME

On the defensive, GNOME has the ability to do everything I need/want. It's a simple and elegant design, which means I don't need to customize it and rip out applications to slim it down. For me, at least, it is very intuitive when I need to make a change or add something to it. The biggest priority for me is system resources; it is not uncommon for my computer to be pushing the max limit because of the magnitude of the programs I run.

—

Chris Stackpole

You make a good point about GNOME using fewer resources than KDE. However, when I really want to minimize the resources used by a desktop, I run IceWM or Fluxbox. To each his own.—Ed.

Make Innovation Not Duplication

[There is] a real challenge to those who design user interfaces for Linux. In order for Linux truly to eclipse Microsoft on the desktop, it is not enough to be "as good as Windows". It has to set a new paradigm, one in which productivity surpasses anything possible with Windows.

To paraphrase your comments [see etc/rant, March 2006]: choice is nice, but it's not an end in itself. If you have only crap to choose from, then choice means nothing. We don't need to have a choice among user interfaces that are basically patterned after what Microsoft has established. We need to be able to choose a better way of doing things with a computer in general. We need a better paradigm of user interface. We need something that really emulates the way we think as humans, and not how Microsoft believes we should think.

—

Ken Peterson

Make Raves Not Rants

I've been a subscriber for five years or so. If the badgering rants of Petreley continue, I'm canceling. In fact, I don't want to read any rants. I like raves.

—

John Elliott

We print raves. See the raves about the rant column, for example.—Ed.

KOffice Live Links

I'm a happy reader of *LJ* next to also being a core programmer of KWord and several of the other KOffice components. As your article went over a core feature [see the February 2006 etc/rant column], and you apparently missed some aspects of it, I would like to invite you to do a more thorough review of the capabilities of KOffice. In direct reference to your article, I can inform you that in KOffice the OLE kind of embedding is actually done quite innovatively (since 1999 already), and much like you have witnessed in EIOffice, there is a way to keep the embedded document external so that the chart and spreadsheet data is updated whenever the external document is edited.

—

Thomas Zander

I couldn't find the way to do what you described. It obviously isn't as simple as the EIOffice method, which is simply "copy" and then "paste link".—Ed.

Boobies out of Place

I found the cartoon with reference to "boobies" out of place in a professional magazine. It was hard for me to enjoy reading the March 2006 issue of *Linux Journal*. My mind kept wandering to such places as pondering if my coworkers were thinking about my "boobies" while I was trying to convey sophisticated technical material, perhaps in defense of our Linux operating systems.

—

Linda Hedges

Why Isn't etc/rant LSB-Compliant?

Wouldn't it be more appropriate to call your column "var/rant" rather than "etc/rant"? Given the context of the column, it's not implied that one should take the content as fact, but rather coincidental output based on certain circumstances. Just a thought....

—

Ken Peterson

You're probably right, but you gave it much more thought than I did.—Ed.

Skype Hype

In the January 2006 issue of *Linux Journal*, (Home Projects) there was an article regarding using a Linux-based Skype Server for your home telephone service [see Andrew Sheppard's "Build a Linux-Based Skype Server for Your Home Phone System"]. I set up the server using Fedora Core 3, following the article every step, until I completed installing SkypeMate. Long story short, it didn't work. Please help.

—

Jimmy

Andrew Sheppard replies: *Linux is, sadly, the poor cousin of Windows in terms of hardware driver support. However, there's an open-source project for Linux to provide independent drivers and support for the Yealink B2K USB/PSTN phone adapter. Here are the links:* **savannah.nongnu.org/projects/usbb2k-api** *and* **memeteau.free.fr/usbb2k**.

The default mode may be different depending on what version of the B2K adapter you have, and who re-badged it (they all come from the Yealink factory, as far as I can tell). In my case, leaving the PSTN line unplugged will leave the adapter in USB mode by default.

World's First 1U 64GB Server
RS-1564 Open Series

Why you should choose OSS ?

- **Customer Service:** OSS has an industry leading 98% customer retention rate. Our success relies on your satisfaction.

- **Rapid Delivery:** OSS ships most orders from 1-1000 systems in 3-6 days. OSS designs and manufactures all systems in our state of the art multi-million dollar, fully-automated 32,000 ft^2 production line.

Find Out More »

1-866-664-STOR
sales@OSShpc.com

OpenSourceStorage.com

OSS is not just another vendor…
We're your partner in business.

Open Source Storage®

AMD Solution Provider GOLD

diff -u

WHAT'S NEW IN KERNEL DEVELOPMENT

The **Raw Driver**, used to gain direct access to unbuffered I/O on block devices, has been deprecated for a long time, now that the open() system call supports the **O_DIRECT option** to provide the same functionality. **Adrian Bunk** has been orchestrating the deprecation and removal process, only to hit a snag at the final moment. As is usually the case with unwanted user-facing kernel features, simply removing them tends to present a dilemma: either users must find an alternative to that feature, or they must no longer upgrade their kernel. Such situations usually result in the feature remaining in the kernel while a grass-roots effort is made to clean up user space. In the current case with the Raw Driver, it turns out that many users still depend upon it, although a lot of them are making efforts to migrate to O_DIRECT as quickly as possible. But with such widespread use, it's also likely the Raw Driver will have to be kept in the kernel for a long time to come.

Adrian also has been continuing his work to remove all **OSS sound drivers** from the kernel, but it is slow going. There are still approximately 50 OSS drivers to deal with. Some are for hardware that is fully supported by ALSA, and so those can be removed safely. Others have incomplete or broken ALSA equivalents that need to be fixed, and some have no ALSA versions at all. Adrian has been very diligent over a long period of time, tracking down driver authors and bugs, working with users to identify missing ALSA features and making sure that only truly obsolete OSS drivers are removed and not any that actually are still needed.

An old **ATI RADEON framebuffer driver**, not updated since 2002 and long since obviated by a newer driver, has been patched out of the kernel by **Michael Hanselmann**. Although the old driver has been marked as old for a long time, the replacement is not perfect either. In particular, **David S. Miller** has pointed out a bug in the screen blanking routing that can confuse the X Window System under some conditions. But even David favors Michael's patch, as do other big-time kernel hackers like **Benjamin Herrenschmidt**, so it does seem as though the old driver will be removed before too long. However, **Andrew Morton** also has said that if possible, he would "prefer to avoid any userland breakage" when removing the older driver.

Jeff Garzik has published the hardware specifications of two previously closed SATA controller chips, **Silicon Image's 3114** and **3124 chipsets**. Silicon Image graciously gave Jeff permission to publish these docs, presumably after much private discussion. This new documentation also may encourage support for **NCQ** (Native Command Queuing), used in high-performance data transfer. This kind of openness must be appreciated in a hardware company. It's important to remember that a lot of hardware remains completely undocumented to the free software community, requiring much effort in reverse engineering or else the abandonment of support for a given product entirely.

Although **Wim Van Sebroeck** has been maintaining the **Watchdog drivers** for a while now, he has just agreed to add himself to the **MAINTAINERS file**. **Kumar Gala** recently asked on the kernel mailing list about tracking down the Watchdog maintainer, and **Arnd Bergmann** was the one to suggest that Wim add himself to the file.

Kernel configuration is always under scrutiny for ways to simplify and clarify the myriad available options. Recently, **Randy Dunlap** hit on the idea of migrating **SATA configuration** out of the SCSI area entirely. SATA does depend on SCSI to provide a function library, but that library could be implemented anywhere, without being tied to SCSI. As Randy reasoned it, there was no reason for users to have to understand this esoteric relationship between Serial ATA and SCSI. And apparently, although Randy himself is not yet interested in seeing this change accepted into the kernel, the idea seems to have general support among kernel developers, and it probably would be accepted if Randy submitted a version that satisfied him.

—Zack Brown

On the Web

LINUX CONSULTANTS SURVEY

For the past six months or so, Ken Hess has been conducting an on-line Linux Consultants Survey to gather consultants' opinions on Linux, both its current state and its future. Now, he's sharing the results of that survey with linuxjournal.com readers (**www.linuxjournal.com/article/8873**). Based on their customers' experiences, find out what Linux pros are saying about Linux in the data center, as a server and on the desktop.

MAKING VOTING SAFE AGAIN WITH OPEN SOURCE

It seems as though every branch of government spends countless hours and money on its voting system—collecting ballots, counting ballots, recounting, recounting and recounting—and we the people still can't trust the results. Clearly, closed and proprietary systems aren't working, so why not extend democracy to the voting system itself and make it open source? In "The Politics of Honest Voting" (**www.linuxjournal.com/article/8872**), *LJ* Publisher Phil Hughes outlines what an open-source voting system might look like. Share your thoughts on the matter, and get involved with turning the current system on its head.

DOC'S BLOG

Senior Editor Doc Searls is blogging now on linuxjournal.com, bringing breaking news and commentary on Linux business, trends and evolution. Bookmark this page **www.linuxjournal.com/blog/800285** to go straight to his blog, or sign up for the LJ.com RSS feed at **www.linuxjournal.com/xstatic/aboutus/rss_page** to be notified when a new entry is posted.

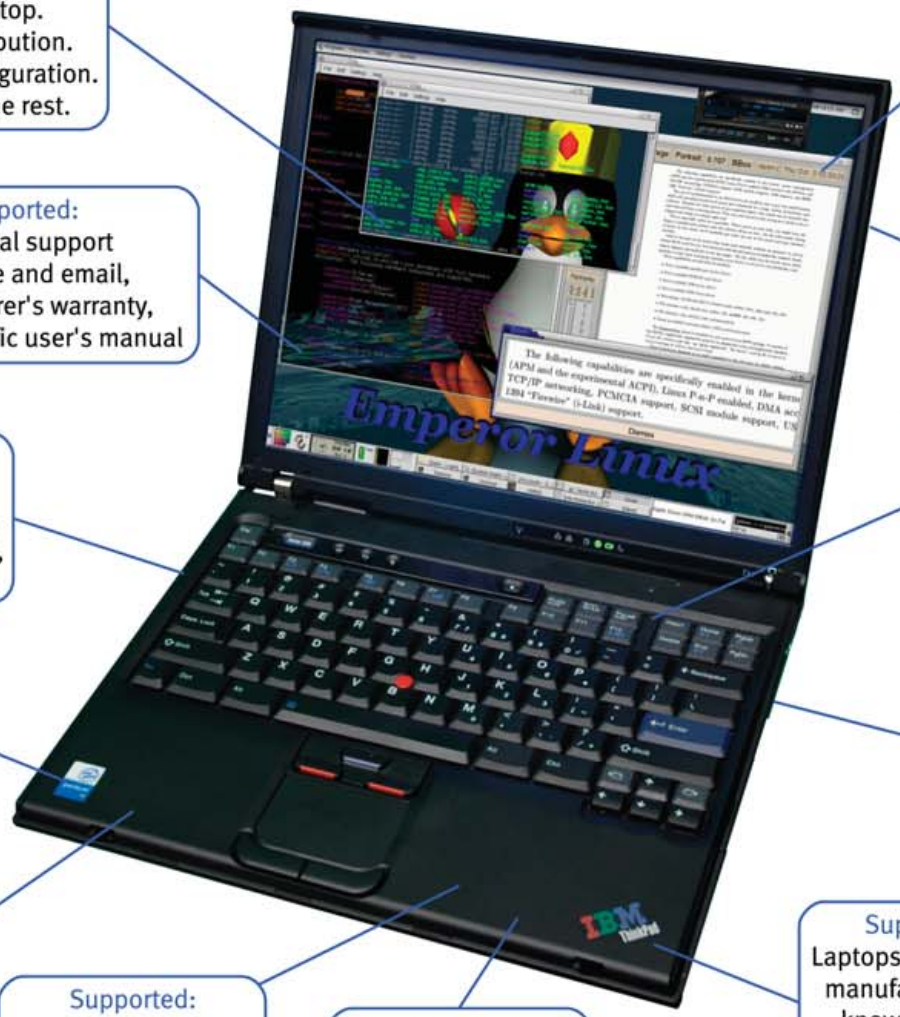
Linux laptops. Supported.

Supported:
Pre-configured Linux installation.
You choose your laptop.
You choose your distribution.
You customize your configuration.
Let EmperorLinux do the rest.

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

Supported:
Technical support by phone and email, manufacturer's warranty, system-specific user's manual

Supported:
Power management: suspend, hibernate, processor control



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

Supported:
One touch to control: suspend, hibernate, brightness, volume, external VGA, wireless

Supported:
True multiprocessing with Intel Core Duo CPU, up to 4 GB RAM

Supported:
Port connectivity: USB, PCMCIA, VGA, Express Card, SVideo, parallel, serial, FireWire

Supported:
Media cards: Compact Flash, Secure Digital

Supported:
Internal optical drive: CDRW, DVD±RW

Supported:
Biometric fingerprint (GDM login with PAM)

Supported:
Laptops from top tier manufacturers you know and trust: Dell, Lenovo, Sharp, Panasonic, Sony

Since 1999, EmperorLinux has provided pre-installed Linux laptop solutions to universities, corporations, and individual Linux enthusiasts. We specialize in the installation of the Linux operating system on a wide range of the finest laptops and notebooks made by IBM, Lenovo, Dell, Sharp, Sony, and Panasonic. We offer a range of the latest Linux distributions, as well as Windows dual boot options. We customize each Linux distribution to the particular machine it will run upon and provide support for: ethernet, wireless, EVDO mobile broadband, PCMCIA, USB, FireWire, X-server, CD/DVD/CDRW, sound, power management, and more. All our systems come with one year of Linux technical support by both phone and email, and full manufacturers' warranties apply.

Visit www.EmperorLinux.com or call 1-888-651-6686 for details.

EmperorLinux

...where Linux & laptops converge

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

Model specifications and availability may vary.

GET THE SERVER THAT TAKES YOUR BUSINESS AS SERIOUSLY AS YOU DO.

Intel® Xeon® Processors powering **Ciara VXPRO™** server provide the quality and dependability to keep up with your growing business.

3 Good Reasons to Buy from Ciara Technology

Built To Order

Customizes Servers, Blade & Storage:
Get the technology that's right for your business and not right for your supplier. With a capability to manufacture over 2500 systems a day, Ciara is suited to accomodate any customer requirement. Our record growth enabled us in February 2003 to inaugurate an all-new, ultra-modern manufacturing plant of 576.000 sqf. Our systems are build under the ISO 9001-2000 certification.

Service...Service...Service

Incorporated in 1984, Ciara Technology is a world-class provider of computer systems including desktop, laptop, servers, storage, supercomputer clusters as well as other software and integration services. All our systems are serviced by Ciara highly trained and certified technicians and system engineers. We are an accountable supplier - One single point of contact for all your technoloy needs.

Intel Premier Provider

Ciara have a strong working relationship with Intel, so we have access to information and support that give us – and you – significant advantages in deploying and managing your systems and applications. The result is a more flexible solution that meets your current needs, while enabling easy expansion to accommodate emerging technologies and new business growth.

For all your computer needs visit us at

ciara-tech.com 1-866-789-7225

-TECH

$1099[1]

VXPRO-R7230NH
Affordable Starter Server

Ultra Addfordable,
Easy to Service
Base Configuration

Intel® Pentium® D Processor 920
(2.8GHz Dual Core Processor)
800MHz Front Side Bus
2x2MB L2 Cache

1GB (2x512MB) ECC DDR2 533 Memory
(Expandable to 8GB)

One 80GB (7,200RPM) SATA150 HDD
(Up to 4 Hot Swap Hard Drives)

Floppy and CD-ROM included

1U Rackmout
3 Years Warranty Return to Ciara

Upgrade Your System

Intel® Pentium® D Processor 930 (3.0GHz Dual Core Processor)	**Add $99**
Intel® Pentium® D Processor 940 (3.2GHz Dual Core Processor)	**Add $140**
Additional 80GB (7,200RPM) SATA150	**Add $72**
Additional 1GB (2 x 512MB) ECC DDR2533	**Add $109**

$2955[1]

VXPRO-R7520BB2
Low Voltage - 4 Cores Server

Extremely Powerful
Ultra Low Power Consumption
Mid Server Configuration

Intel® Xeon® LV Processor
(2.0GHz Dual Core Processor)
667MHz Front Side Bus
2MB of Shared L2 Cache

2GB (2x1GB) ECC/Reg DDR2 400 Memory
(Expandable to 16GB)

One 80GB (7,200RPM) SATA150 HDD
(Up to 4 Hot Swap Hard Drives)

Floppy and CD-ROM included

1U Rackmout
3 Years Warranty Return to Ciara

Upgrade Your System

Additional 80GB (7,200RPM) SATA150	**Add $72**
Additional 2GB (2 x 1GB) ECC/Reg DDR2 400	**Add $299**

intel Xeon® inside™

L J Index, May 2006

1. Smallest number of Weblogs that mention "open source" per day: 500
2. Largest number of Weblogs that mention "open source" per day: 1,050
3. Smallest number of Weblogs that mention "linux" per day: 1,250
4. Largest number of Weblogs that mention "linux" per day: 2,600
5. Percentage of smartphones shipped with Linux in Q1 2004: 3.4
6. Percentage of smartphones shipped with Linux in Q1 2005: 13.7
7. Percentage increase of smartphones shipped with Linux between Q1s 2004 and 2005: 412
8. Current Linux percentage share of advanced mobile OSes: 17
9. Projected Linux percentage share of advanced mobile OSes by 2009: 29
10. Billions of mobile phone subscribers worldwide by late 2005: 2
11. Number of top 500 supercomputers that run on Linux: 360
12. Percentage of top 500 supercomputers that run on Linux: 72
13. Number of top 500 supercomputers that run on Linux distros: 30
14. Percentage of top 500 supercomputers that run on Linux distros: 6
15. Total number of Linux-based supercomputers in the top 500: 390
16. Percentage of Linux-based supercomputers in the top 500: 78
17. Number of Linux-based supercomputers in the top 10: 5
18. Position of the CNK/Linux-based IBM BlueGene/L in the top 500: 1
19. Growth rate in size of the CNK/Linux-based IBM BlueGene/L in the last year: 2
20. Top Linpack performance of the BlueGene/L, in teraflops: 280.6

Sources: 1–4: Technorati (during the month of January 2006; numbers rounded to nearest 50) | 5–7: Gartner, via *Linux Devices* | 8, 9: TDG, via *Linux Devices* | 11–20: Top 500 Supercomputer Sites (t500.org)

—Doc Searls

They Said It

Great things are not done by impulse, but by a series of small things brought together.
—George Eliot or Vincent van Gogh, on Google (many quote sites on the Web are split between the two)

All large systems that work start as small systems that work.
—Stowe Boyd

UNIX is basically a simple operating system, but you have to be a genius to understand the simplicity.
—Dennis M. Ritchie, www.brainyquote.com/quotes/authors/d/dennis_ritchie.html

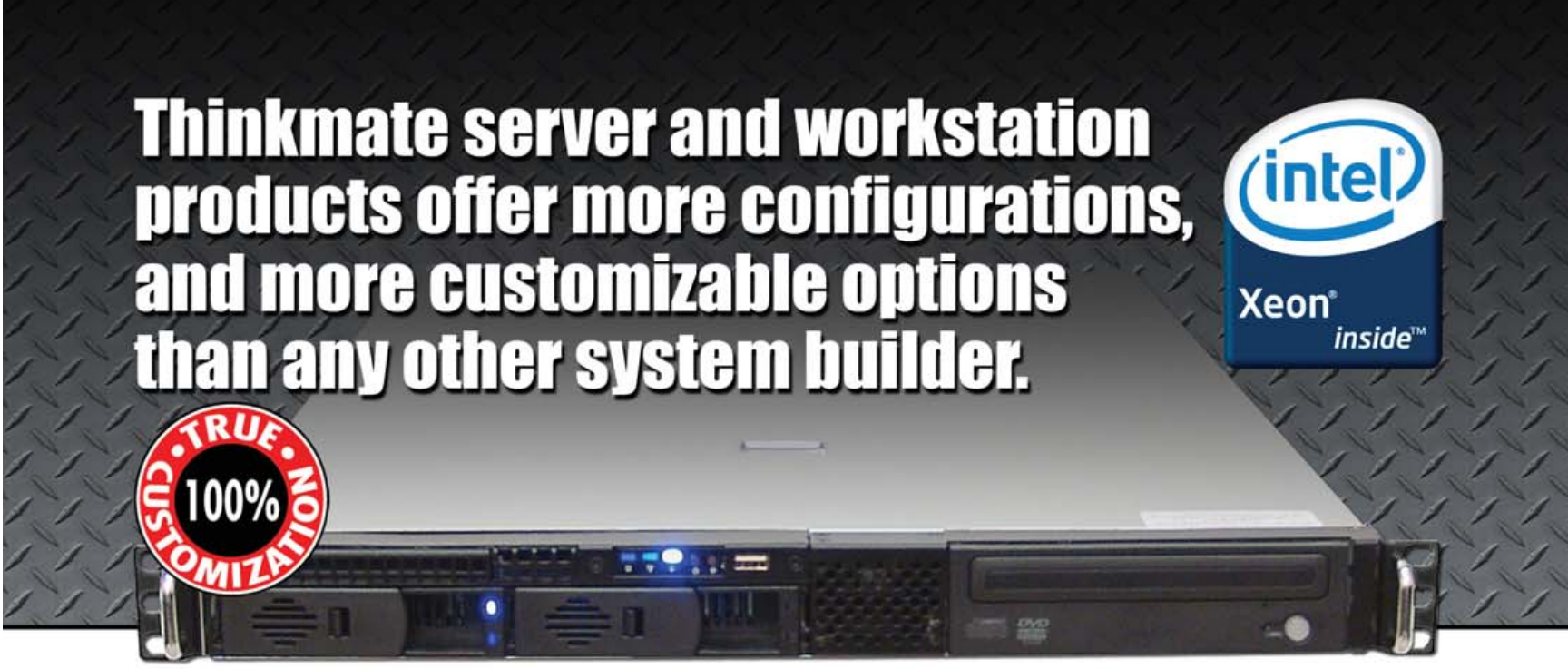
Software wants to become worthless without skilled attention.
—Don Marti (in a conversation)

Skill without imagination is craftsmanship and gives us many useful objects such as wickerwork picnic baskets. Imagination without skill gives us modern art.
—Tom Stoppard, jon.linuxworld.com

When you ask a question about an open-source product, ask the community, not one specific person. When you ask for one person to answer the question, then other people who may know the answer, might not help (in fact they almost never will, assuming you had some reason to want to know the answer from this one specific person). I've been doing this for many years. People almost never want to hear this, so I usually just ignore the questions, even if they have easy answers, because I want a community to develop, one where people help each other. That's the only way it can grow. And I want that kind of growth even more than I want you to get over this particular hurdle.

On the other hand, if you see a newbie ask a question of someone specific, and you know the answer, and you are not the person he or she asked, go ahead and answer it. Assume the person just wants the answer, not really from anyone in particular. If they complain that your name isn't Linus or Brian or Alice, you can tell them that's true, but the answer is still the right one.
—Dave Winer, www.scripting.com/2006/01/14.html#When:8:18:54PM

Thinkmate server and workstation products offer more configurations, and more customizable options than any other system builder.



Thinkmate 1U Server, 2x Serial-ATA or SCSI drives with Intel® Xeon® Processors

Dozens of customizable systems online, unlimited possibilities by phone.

- Rackmount Servers
- High-Performance Servers
- Storage Servers
- Pedestal Servers
- Silent Workstations
- Blade Servers
- Clustering
- Notebooks

Operating System Options:

Thinkmate systems are available with either No OS, or pre-loaded with Linux or Microsoft Windows operating systems. Thinkmate also offers dual-boot and virtualization options.

100% True Hardware Customization:

Thinkmate is an innovative provider of an extensive variety of computer solutions. We completely customize all of our machines to match your individual needs. Our online quoting and ordering system has more customizable options than any other system builder on the web. If you can't find exactly what you need on our site, then give us a call and we would be more than happy to help you find it!

Service:

Thinkmate takes customer service to a new level. All of our systems have a minimum of a 3-year advanced replacement warranty and offer up to a 3-year next business day onsite warranty through IBM Global Services. We understand mission critical situations and provide superior services to keep all of our customers satisfied.

GSA Scheduling:

GSA Schedule Contract GS-35F-0313M

We offer rapid GSA scheduling for custom configurations. If you have a specific hardware requirement, we can have your configuration posted on the GSA schedule within 2-4 weeks.

THINKMATE

(800) 371-1212

WWW.THINKMATE.COM

Have Fon

In early February 2006, the Spain-based company FON (**en.fon.com**) was three-months old and had just 3,000 "Foneros" when founder Martin Varsavsky announced a $21.7 million investment from Skype, Google and Sequoia Capital.

If all those companies have their way, everybody in a position to use or deploy Wi-Fi Net connections will be a breed of "Fonero". Varavsky explains, "To us, the world is divided into Linus, Bills and Aliens. A Linus shares his/her bandwidth for free with other Foneros, Bills share their bandwidth for a small fee, and Aliens don't share their bandwidth at all." Because Aliens are those creatures called customers.

And those needn't be just geeks like the readers of *Linux Journal*. Ethan Zuckerman, in his blog My Heart's in Accra (**www.ethanzuckerman.com/blog/?p=363**), explains why he's both a Fonero and on the company's advisory board:

> There's a philosophical bias to many of these projects—a belief that Internet access is an inalienable right and should be free—that I find charming, but totally impractical for the parts of the world I'm most concerned about.
>
> In Africa, bandwidth isn't cheap. Entire universities run on less bandwidth than I have coming into my house on a DSL line. Being altruistic and leaving your wireless access point open in Africa is pretty much a guarantee that you're going to end up with other users abusing the limited bandwidth you have. It's important that African users have the opportunity to share their bandwidth in a way that allows for "bandwidth shaping"—sharing some bandwidth with other users and retaining the rest for your own needs—and billing, so other users can share the cost with you. FON's current software isn't optimized for this situation yet, but it's close, and FON is engaged with the issues in a serious and sustained way. I predict that FON is something I'll be able to pitch enthusiastically to African friends in the very near future.

To run FON, download software based on Sebastian Gostchall's DD-WRT open-source project (**www.dd-wrt.org**). And, you run it on a FON-compatible router. Right now, that's a Linksys WRT54G/GS/GL (versions 1x to 4x), which are the ones with Linux inside. The first 3,000 are being sold far below cost. Those may be gone by the time you read this, but the company is sure to make it as easy as possible to become a Linus, if not a Bill.

—Doc Searls

Invention Is the Mother of Necessity

Krugle is a new search engine just for source code and other technical stuff. Ken Krugler, company founder and CTO, puts the appeal in simple terms, "Krugle is a search engine for programmers."

I was at a conference in San José when Krugle CEO Steve Larsen showed the beta version of Krugle to Bill Weinberg, an old friend who now works as an Open Software Architecture Specialist at OSDL. "I have to have this", Bill said. Then, when Steve Larsen continued with the demo, Bill added, "No, you don't understand. I need this."

See if it hits you the same way. Check it out at **krugle.com**.

—Doc Searls

The Inevitable, Eventual, Free Linux Desk/Laptop

Will Linux reach mainstream desktops and laptops without a major vendor making the push? Several vendors have recently stepped up to answer that question.

At CES in January 2006, Google cofounder Larry Page made a public show of his company's support (sans details) for MIT's $100 laptop, designed to "revolutionize how we educate the world's children".

At the end of January, Red Hat announced support for the project as well. At the time of this writing, the company is working on adapting Fedora and plans to make the project an open and public one. The company also signed on as a platinum supporter of the Desktop Linux Summit, an event Linspire launched three years ago and still runs.

The *New York Times* also reported that Nicolas Negroponte, who is running the $100 laptop project, is close to lining up $700 million from seven countries—China, India, Brazil, Argentina, Thailand, Egypt and Nigeria—interested in buying 7 million of the units. A Taiwanese manufacturer was also reportedly lined up.

Meanwhile, Nat Friedman showed off Novell's Linux Desktop 10 in Paris. He played videos and MP3 music files (with Banshee, Novell's own player, using licensed patents), downloaded pictures from a digital camera and exchanged photos with an iPod. He also showed off XGL, an open-source graphics subsystem. Right now, it's on track to be available by the time you read this.

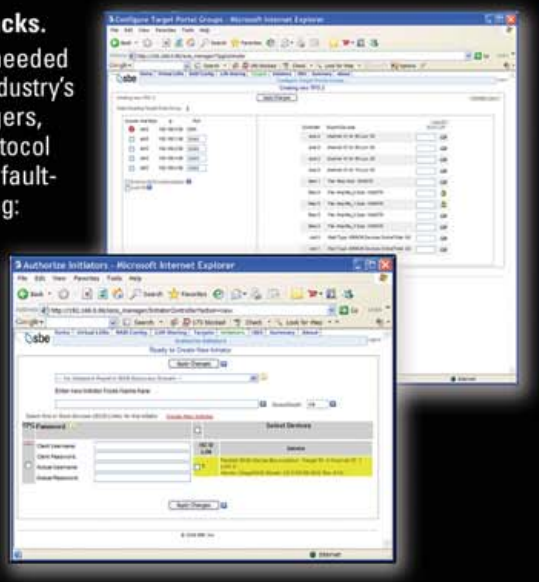
And, of course, the noncommercial open-source projects—GNOME, KDE, freedesktop.org (**freedesktop.org**) and so on—continue to move forward.

—Doc Searls

iSCSI that makes your IP SAN a whole lot richer...

SBE's feature-rich iSCSI software solutions deliver enterprise-level reliability at affordable price points...

Highly scalable & fully standards compliant iSCSI Target and Initiator protocol stacks. Early iSCSI deployments did not offer the high-quality transport and fault tolerance needed by enterprise storage managers. However, SBE's PyX iSCSI solutions were among the industry's first to provide the full error recovery features required by enterprise storage managers, namely Error Recovery Level 2 (ERL2). Providing broad Linux OS support, SBE's iSCSI protocol stack enables robust, cost-effective IP-based storage transport solutions delivering fault-tolerant, high availability networks with a rich mix of advanced feature sets, including:



- Proven performance from Wi-Fi to 10G with unlimited port count
- Aggregation of all available bandwidth across subnets and ports
- Multi-pathing I/O functionality for maximum redundancy and reliability
- Performance as fast as your underlying hardware can go
- Storage media and transport independent architecture
- Full interoperability with all compliant iSCSI initiators

sbe®

flexibility on demand | 925-355-2000 | info@sbei.com | www.sbei.com

REUVEN M. LERNER

Google Web Services

With a little SOAP, cleanliness is next to Googliness.

For the past few months, we've been looking at a number of Web services offered by Amazon, allowing us to search through its catalog with relative ease. Amazon decided several years ago to make its Web services largely free, on the assumption that this would raise the number of people eventually buying from its Web site. And indeed, a large number of developers now use Amazon Web services to create everything from custom bookstores to programs that can help with bookstore management.

Amazon isn't the only commercial Web site that has opened up its catalog to the outside world. Google, another 900-pound Internet gorilla, also released its Web APIs several years ago. These APIs make it possible to search through Google's extensive catalog of Web content. It's impossible to know whether this catalog is the largest in the world, but from my perspective, that's somewhat irrelevant. Google's catalog is large enough, and is updated frequently enough, for me to rely on it as my primary search engine most of the time.

Google has made a number of different APIs available over the last few years. This month, we look at the simplest of them, for performing basic searches of the Web archive. We examine how Google uses WSDL (Web service description language) to advertise its Web services and how we can make SOAP calls to search through Google's extensive library for our own purposes.

Getting Started

If you have worked with Amazon Web services, getting started with Google's APIs will not surprise you a great deal. To begin, both companies require that you register to use their services. Registration is free in both cases and provides you with an identification key that is placed in every request to the server.

To obtain a Google key, you first need to register for a Google account. Now, I've had a "Google account" for some time, for use with services such as Gmail and its personalized news page. However, it seems the APIs are linked to a different set of accounts. The fact that I had to register and log in to the API system, even after initially logging in to my "main" Google account, struck me as a bit odd.

That said, creating an account is simple and straightforward. Go to the main Google API page (**www.google.com/apis**), click on create a Google account, and fill out the form. Soon after submitting the HTML form, you will receive e-mail from Google confirming the creation of your account and containing your Google key, along with a URL to visit in order to confirm the account's creation. After confirming the creation of your account, you're ready to move forward with the use of your Google key, creating programs that take advantage of Google's Web services.

Before we do that though, we should consider the restrictions that Google places on the service and the data we retrieve through it. Amazon allows participants to make only one API call per second, which means a maximum of 86,400 calls in a given 24-hour period. Google, by contrast, allows users to make only 1,000 calls in a given 24-hour period.

Moreover, the way in which these maximums are defined indicates the way in which violations will be handled. Google will return an error message if you have made more than 1,000 queries in the previous 24 hours, whereas Amazon will complain only if a query comes within one second of a previous query. Neither service keeps track of these numbers before returning an error message, but it is obviously easier to recover from violating Amazon's restrictions (by sleeping for one second and retrying) than Google's (as the program might need to sleep for up to 24 hours before retrying).

There are a number of legal differences between the two sites' services. Amazon pioneered the idea of affiliate vendors on the Web, encouraging people to create commercial services around its database. By contrast, Google explicitly states that users are forbidden from creating a commercial service around its search results. (If you are interested in creating a commercial service based around Internet search data, consider looking at Amazon's Alexa Web search platform service, which doesn't have these restrictions. At the same time, it'll cost you 25 cents for every 1,000 requests, which can add up quickly for a popular site.)

Finally, there are some technical differences between the two sites. Amazon's APIs work via both SOAP and REST, allowing developers to choose between these two formats. Google, by contrast, provides only a SOAP interface to its search engine. So, in order to create our search system, we need to install and use a SOAP client library. Fortunately, most languages have high-level libraries that allow for SOAP calls.

SOAP::Lite

SOAP, formerly the Simple Object Access Protocol, but now an acronym that officially doesn't stand for anything, provides a relatively easy method for sending an XML-encapsulated query to a server. The server then responds with an XML-encoded response. Over the years, SOAP has strayed far from its simple roots. Although SOAP is still easier to understand, implement and work with than some more complicated protocols (such as CORBA), it is more difficult than most people would like to admit. If I can get away with it, I personally prefer to use XML-RPC for Web services. Although XML-RPC doesn't offer all of the features of SOAP, it is far easier to work with.

That said, Google requires that we use SOAP, and with many good SOAP client libraries available nowadays, we should not be afraid to work with it. Perl programmers have a particularly strong implementation, known as SOAP::Lite, at their disposal. For the programming examples in this article, we use Perl and SOAP::Lite. Note that the Lite part of the module name describes the ease with which programmers can implement Web services, not a stripped-down version of SOAP. You can install the latest version of SOAP::Lite from CPAN by typing:

```
perl -MCPAN -e 'install SOAP::Lite'
```

The SOAP::Lite installation will ask you to indicate which tests, if any, you want to perform before installing the module. I normally accept the defaults, but you might want to add to or remove from these depending on your needs.

With SOAP::Lite installed, it's time to write a program that

The Industry Leader for Server Appliances

Custom server appliances or off the shelf reference platforms, built with your image and software starting under $1,000.

From design to deployment, we handle it all.

Delivering an appliance requires the right partner. MBX Systems is the right partner. We understand that by putting your name on our hardware, you're putting your reputation in our hands. We take that seriously. We provide the services you need to service your customers. Better than the competition. You never even need to touch the hardware. Engineering. Design. Deployment. We handle it all, so you can focus on what's important to you. Your software. Your sales. Your success.

Visit us at www.mbx.com or call 1-800-681-0016 today.

MBX systems

www.mbx.com | 1-800-681-0016

queries Google. But to do that, we need to know the URL of the service, as well as the method that we will be invoking on Google's computer, along with the names and types of any parameters we want to send. We could specify these by hand, but that would mean a lot of work on our part. Moreover, Google currently expects SOAP requests to be pointed at **api.google.com/search/beta2**. If Google ever decides to change that URL without warning, many people might be surprised and upset.

Luckily, Google has provided a WSDL file, describing the services offered via Google's APIs, as well as the request and response parameters the system accepts. It also describes the endpoint for queries, allowing Google (in theory) to make changes to the service without notifying developers in advance. Of course, this assumes that the WSDL file itself will remain in the same location. It also assumes that the names of the services will not change, and that each of them is documented somewhere, because the choice of which method to invoke still requires human intervention.

WSDL is written in XML, and it is fairly easy to understand, once you realize that it's describing nothing more than the various Web services available on a particular server, including the number, names and types of inputs. Thus, the WSDL entry for doGoogleSearch, which performs the basic Google search of Web content, is defined as follows:

```
<message name="doGoogleSearch">
<part name="key"             type="xsd:string"/>
<part name="q"               type="xsd:string"/>
<part name="start"           type="xsd:int"/>
<part name="maxResults"      type="xsd:int"/>
<part name="filter"          type="xsd:boolean"/>
<part name="restrict"        type="xsd:string"/>
<part name="safeSearch"      type="xsd:boolean"/>
<part name="lr"              type="xsd:string"/>
<part name="ie"              type="xsd:string"/>
<part name="oe"              type="xsd:string"/>
</message>
```

To use WSDL from within a Perl program using SOAP::Lite, we invoke SOAP::Lite->service with the WSDL file's URL. If the file resides on the local filesystem, make sure that the URL begins with file:. For example:

```
my $google_wsdl = "http://api.google.com/GoogleSearch.wsdl";
my $query = SOAP::Lite->service($google_wsdl);
```

SOAP::Lite is then smart enough to look through the WSDL and make all of the advertised methods dynamically available, such that we can do the following:

```
my $results =
$query->doGoogleSearch($google_key,
                $query_string,
                $starting_page,
                $max_results,
                $filter,
                $geographic_restriction,
                $safe_search,
                $language_restriction, 'utf-8', 'utf-8');
```

Do you see what happened here? There is a one-to-one mapping between the inputs described in the WSDL and the parameters that we pass to $query->doGoogleSearch().

Simple Queries with doGoogleSearch

We have now seen the core of our Google search program written in Perl. All that's left is to review the input parameters and the contents of $results, which contains the results returned from Google.

The documentation for the API at **www.google.com/apis/reference.html** describes the input parameters. All of them are mandatory, but some of them are more important than others. In particular, the Google key and the query string typically will be set, and the others will be set with simple default values, as you can see in Listing 1.

Most people, including myself, typically want to query the widest possible number of Web pages with our queries; however, there are times when it is more appropriate to retrieve data only from servers in a particular geography or in a certain language. The fact that Google's API makes this possible and straightforward opens the door for many different interesting applications.

Just as we send a query to Google via SOAP-encoded XML, we receive a result in SOAP-encoded XML. But as SOAP::Lite shielded us from having to write even a tiny bit of XML for the query, we similarly will be insulated when it comes to the response. The $results variable provides a Perl interface to the data that we received in response.

And exactly what data will we receive? To know that, we can look at the WSDL file once again. It indicates (among other things) that we will receive responses as a set of results, each of which looks like this:

```
<xsd:complexType name="ResultElement">
<xsd:all>
    <xsd:element name="summary" type="xsd:string"/>
    <xsd:element name="URL" type="xsd:string"/>
    <xsd:element name="snippet" type="xsd:string"/>
    <xsd:element name="title" type="xsd:string"/>
    <xsd:element name="cachedSize" type="xsd:string"/>
    <xsd:element name="relatedInformationPresent" type="xsd:boolean"/>
    <xsd:element name="hostName" type="xsd:string"/>
    <xsd:element name="directoryCategory" type="typens:DirectoryCategory"/>
    <xsd:element name="directoryTitle" type="xsd:string"/>
</xsd:all>
</xsd:complexType>
```

In other words, each search result we receive back from Google (up to a maximum of ten) will provide all of the information we need to create a results page that looks just like Google's. Moreover, we can pick and choose the elements we want to display, showing (for example) only the title and the dmoz directory category and title. Or we can show a short snippet from the searched page. Or all of these. Or none of these.

doGoogleSearch is not the only method described in the WSDL file. There also are other methods, such as working with Google's cached pages and checking the spelling of individual words. When Web services were first unveiled to the public, a common example was that a word processor would now be able to call a remote Web service for spell-checking, rather than coming with a built-in system. That day is still far off in the future, but you can imagine using Google's API for an experimental version of such a service.

Moreover, we can use these outputs as inputs into another Web service call, either locally or remotely. Combining data from multiple sites is an increasingly popular thing to do, especially when combined with Google's maps API. It's amazing to see what can happen when you combine services in this way—something that we will explore in the coming months.

Listing 1.
google-query.pl

```
#!/usr/bin/perl

use strict;
use diagnostics;
use warnings;

use SOAP::Lite;

# --------------------------------
# Get the Google key from ~/.google_key
my $google_key_file = "/Users/reuven/.google_key";
open GOOGLE_KEY, $google_key_file or die "Cannot read
'$google_key_file': $! ";

my ($google_key) = <GOOGLE_KEY>;
chomp $google_key;

close GOOGLE_KEY;

# --------------------------------
# Get the command-line argument
if ($#ARGV != 0)
{
    print "$0: Invoke with a single argument, your Google search term.\n";
    exit;
}

my $query_string = shift @ARGV;

# --------------------------------
# Get the WSDL file
my $google_wsdl = "http://api.google.com/GoogleSearch.wsdl";
my $query = SOAP::Lite->service($google_wsdl);

# --------------------------------
# Use the WSDL to make the query
my $starting_page = 1;
my $max_results = 10;
my $filter = 'false';
my $geographic_restriction = '';
my $safe_search = 'false';
my $language_restriction = '';

my $results =
    $query->doGoogleSearch($google_key,
                $query_string,
                $starting_page,
                $max_results,
                $filter,
                $geographic_restriction,
                $safe_search,
                $language_restriction, 'utf-8', 'utf-8');

my @results = @{$results->{resultElements}};

if (@results)
{
    # Iterate through each result we got
    my $counter = 1;
    foreach my $result (@results)
    {
    print "Result $counter of ", $#results + 1, ":\n";

    foreach my $key (sort keys %{$result})
    {
        my $value = $result->{$key};

        # Is this a hash value?  If so, display it accordingly
        if (UNIVERSAL::isa($value, 'HASH'))
        {
        print "\t'$key':\n";

        foreach my $subkey (sort keys %{$value})
        {
            print "\t\t'$subkey' => '$value->{$subkey}'\n";
        }
        }

        # Display the value as a simple string

        else
        {
        print "\t'$key' => '$value'\n";
        }
    }

    $counter++;
    }
}
else
{
    print "There were no results for your query of '$query_string'.\n";
}
```

Conclusion

This month, we took a brief look at Google's search API. Using some simple tools, including the SOAP::Lite module for Perl, we were able to build a simple command-line version of Google's search page. In coming months, we'll look at Google's map API and begin to see how we can create mashup services that combine multiple data sources.■

Resources for this article: www.linuxjournal.com/article/8881.

Reuven M. Lerner, a longtime Web/database consultant, is currently a PhD student in Learning Sciences at Northwestern University in Evanston, Illinois. He and his wife recently celebrated the birth of their son Amotz David.

Just because your IT equipment goes dark doesn't mean you have to go blind.

SecureLinx SLC

Reach your IT equipment from anywhere as easily as changing a light bulb.

SecureLinx™ SLC secure console managers from Lantronix provide consolidated access so you can control, manage and repair your IT equipment from anywhere, at anytime.

For your free Console management white paper visit ***lantronix.com/slcwp/***

Network down?
No sweat.
On the road?
No problem.

With SecureLinx you can finally achieve true "lights out," out-of-band data center management. And SecureLinx SLC has your back with the highest level of security available. For more information, check out the specs at **lantronix.com/slc/** or call us at (800) 422-7055.

LANTRONIX®
Network anything. Network everything.™

2005 Network Computing WELL-CONNECTED AWARDS WINNER

Network Computing Editor's Choice

Lantronix's SecureLinx SLC16
Awarded Network Computing's
Editor's Choice Award

Sun microsystems Solaris™ Ready

MARCEL GAGNÉ

The Virtual Streets of $HOME

Use Linux to visit virtual consoles, cities and battle zones.

François, what are you looking for on Freshmeat? *Quoi?* A program to digitize you so you can go inside the computer? Yes, I know what it looks like in the movies, but virtual reality hasn't quite made it there yet. I thought you understood that when we discussed lightcycles months ago. No, François, I don't think people are going to be living inside computers anytime soon. I'm not laughing at you, *mon ami*. I am just amused, that's all. No, I'm sorry to disappoint you, but I don't think there are cities or people in your Linux system either. We will discuss this later. Our guests will be here any moment, and we must be ready for them.

What did you say, *mon ami?* They are already here? Quickly, François, help our guests to their tables. Welcome, everyone, to *Chez Marcel*, where fine wine meets exceptional Linux fare and the most superb clientele. When you have finished seating our guests, François, head down to the wine cellar and bring back the 2002 Côtes du Roussillon Villages.

François and I were just discussing the possibility of virtual worlds inside our computers, a truly amazing prospect but one that is still fantasy. It's true that amazing things have happened in the time I've been working with computers. Your Linux system is one of those things, and its open nature means a freedom to explore that simply doesn't exist elsewhere. Still, I keep thinking that the computing model in general is still in its infancy. Maybe it's because I watched too much science fiction and as a result, my expectations are a bit high. Think back to the movie *Tron*, for instance. In the opening sequence, Flynn the hero of the show, sends a program named CLU into the system to locate some missing files. CLU, the program, looks like Flynn and moves around in a 3-D tank while a companion bit offers yes or no advice. There are towering skyscraper-like structures all around as he navigates his tank down digital streets. That's the virtual computer world I wanted to experience in my younger days.

Ah, François, you have returned with the wine. Please, pour for our guests. May I suggest, *mes amis*, that you enjoy the many hidden flavors in this excellent red.

Although there may be no hidden worlds inside the system, plenty of things are otherwise hidden from view. Virtual consoles, for instance, scroll information that is hidden from view once your graphical desktop starts up. Sure, you could jump out of your graphical session with a Ctrl-Alt-F1 to see what is happening out there, but there is a better way. To view the hidden contents of that virtual console, type the following at a shell prompt (you will need root permissions for this):

```
cat /dev/vcs1
```

You see, what you may not know is that your system keeps track of the contents of those virtual consoles (1–6) in a special device file, /dev/sdaX, where X is the number of your virtual console. For example, here is a sample of the output of the first VT on my Ubuntu test system:

```
* Starting OpenBSD Secure Shell server...
[ ok ]  * Starting Bluetooth services...  hcid sdpd
[ ok ]  * Starting RAID monitoring services...
[ ok ]  * Starting anac(h)ronistic cron: anacron
[ ok ]  * Starting deferred execution scheduler...
[ ok ]  * Starting periodic command scheduler...
[ ok ]  * Checking battery state...
[ ok ]  * Starting TiMidity++ ALSA midi emulation...
[ ok ]
Ubuntu 6.04 "Dapper Drake" Development Branch francois tty1
```

This is interesting stuff, but it hardly qualifies as a hidden world, and it just doesn't have the *Wow!* factor my humble waiter is looking for. Yet, despite what I said to François, there are ways to see cities inside your Linux system. It's a bit of a stretch, but some fascinating visualization programs exist—experimental in nature—that try to create a real-world view of the virtual world of processes, memory and, of course, programs. One of these is Rudolf Hersen's ps3 (see the on-line Resources), and to take full advantage of ps3, you need a 3-D video card with acceleration.

Compiling the program is fairly simple, but it does require that you have the SDL development libraries:

```
tar -xjvf ps3-0.3.0.tar.bz2
cd ps3-0.3.0
make
```

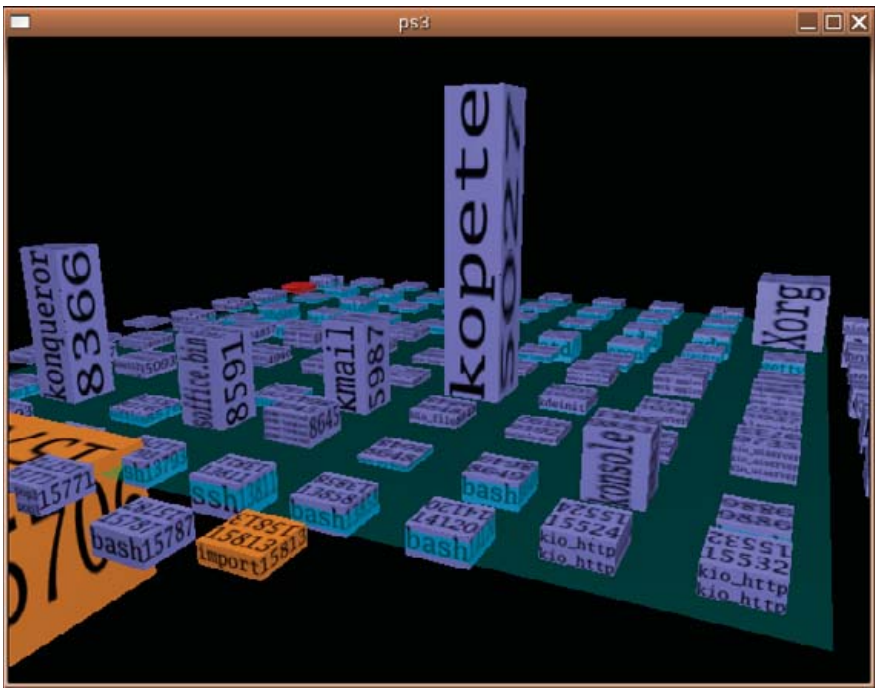


Figure 1. Navigating system processes with ps3 is like flying down into a virtual city.

NOTE:

If you find yourself missing the lightcycles, check out Marcel's "Battles Inside the Computer" on our Web site at **www.linuxjournal.com/article/6638**.

To run the program, type `./ps3` from the same directory, and you should see a 3-D representation of your process table. When it starts, you may get something other than an ideal view, but that's the whole point of ps3. You can rotate the views in all three axes and look at the process table from above or below. If the processes are too high at the beginning, simply scale them down to something more reasonable. Each process is identified by its program name and its process ID.

Navigating the ps3 display is done entirely with the mouse. Click the left-mouse button and drag to rotate and adjust the height and speed of horizontal rotation. Click and and drag using the right-mouse button to rotate the view horizontally and vertically. The wheel on your mouse lets you zoom in and out. To quit the ps3 viewer, press the letter Q on the keyboard.

ps3 is in no way a scientifically accurate means of viewing system processes, but it is enlightening and entertaining. So now we have virtual buildings and the makings of a virtual city somewhere inside your system. All we're missing now are tanks. Well, I may have an answer to that one as well. It's called *BZFlag*, and this certainly calls for François to refill our glasses. *Mon ami*, if you please.

BZFlag is a multiplayer 3-D tank battle game you can play with others across the Internet (Tim Riker is the current maintainer of *BZFlag*, but the original author is Chris Schoeneman). The name, *BZFlag*, actually stands for *Battle Zone capture Flag*. It is, in essence, a capture-the-flag game. To get in on the action, look no further than your distribution's CDs for starters. *BZFlag*'s popularity means it is often included with distros. Should you want to run the latest and greatest version, however, visit the *BZFlag* site (see Resources). You'll find binaries, source and even packages for other operating systems. That way, you can get everyone in on the action.

Figure 2. At any given time, dozens of *BZFlag* servers are running worldwide and hundreds of people are playing.

Unless you specify otherwise, *BZFlag* starts in full-screen mode, but you can override this by starting the program with the -window option. The game begins at the Join Game screen. Before finding a server (the first option on the screen), you may want to change your Callsign (or nickname). We'll look at some of these other options after we've selected a server. For now, move your cursor to the Find Server label and press Enter.

You won't have any trouble finding people to play with—you'll get a list of dozens of servers currently hosting games (Figure 2). Scroll down the list of names to find one that suits you. Your criteria might be the number of players, how busy a server is or how many teams are involved. When you look at the server list, make sure you pay attention to the type of game being hosted on the server. Some have team-oriented capture-the-flag play, and others host free-style action. You also may be limited by the number of shots at your disposal, so aim carefully.

When you have made your choice, press Enter, and you'll find yourself back at the Join screen (Figure 3) with a server selected. You could simply start the game, but you may want to fine-tune a few more things before you start up your tank. Cursor down to the Team label, and press your left or right arrow keys. By default, you will be assigned to a team automatically, but you can change that here if you prefer. One of the roles you can play instead of joining a team is that of Observer. This is not a bad idea if you are new to the game, because it lets you watch how others are handling themselves.

Figure 3. The Join screen lets you define your callsign as well as your team.

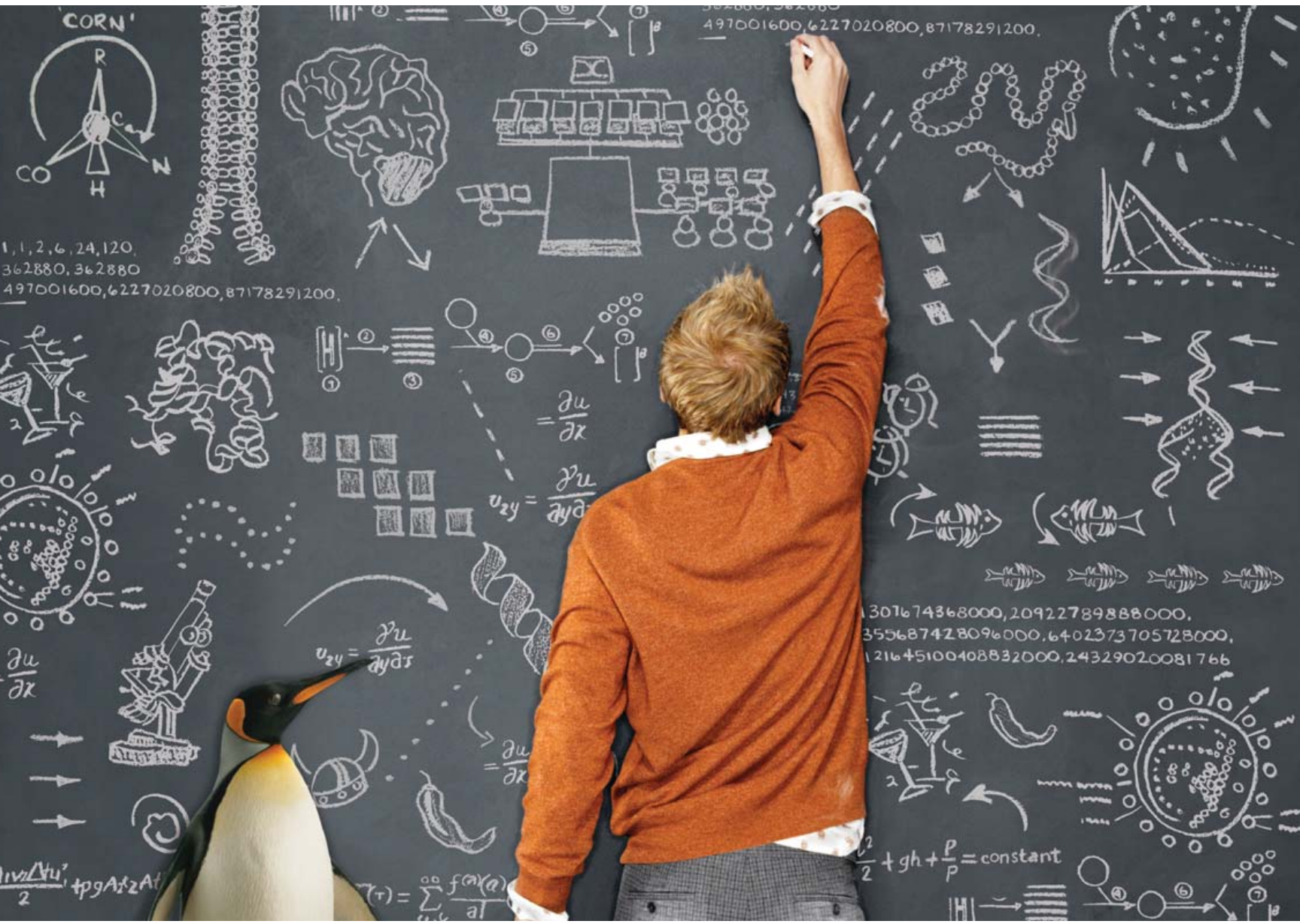
The Join screen also lets you enter the name of a server manually, rather than search for it. This is useful for hosting private games on a local LAN. Speaking of hosting games, I'm sure you noticed the Start Server option at the bottom of that list. Let's go ahead and join the game. Scroll back up to Connect and press Enter.

I hope you are ready, *mes amis*, because the action starts immediately, and some of these players are, well, *seasoned*. Move your tank using your mouse, and fire by clicking with the left-mouse button. These tanks are highly maneuverable and even can jump in some games (you do this by pressing the Tab key). To learn all the keystrokes, by the way, press Esc at any time, and select Help. During play, *BZFlag* provides an extensive heads-up display

THE PENGUIN CURES

Millions of compounds to go. Database analyses keep one computer clogged; while a microarray analysis chokes the other. The computer hopping begins; so does the throbbing in your brain. Exhale. Penguin Computing® Clusters combine the economy of Linux with the ease of Scyld. Unique, centrally-managed Scyld ClusterWare™ HPC makes large pools of Linux servers act like a single virtual system. So you get supercomputer power, manageability and scalability, without the supercomputer price. Penguin Computing. So many drugs. So little time.

HEADACHES



Highly SCYLD

AMD 64 Opteron

REALLIFELINUX

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

PENGUIN COMPUTING®

www.penguincomputing.com

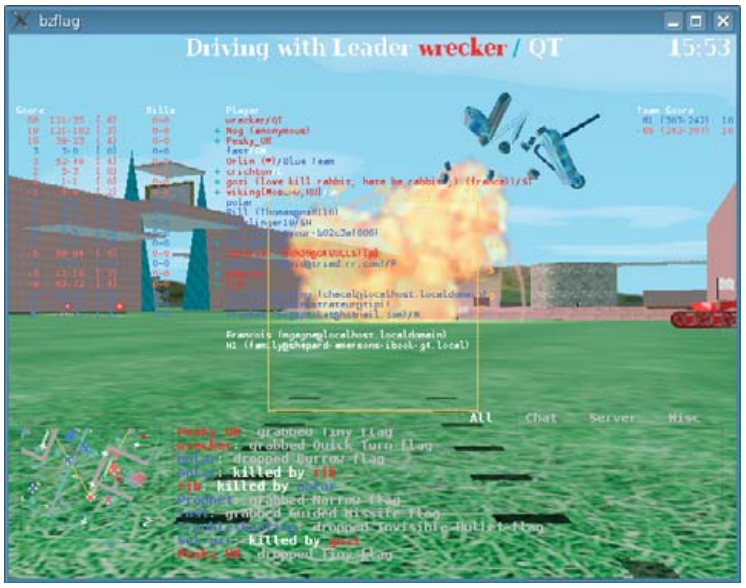


Figure 4. The action is fast and tense, with tanks blowing up everywhere you turn. Be careful not to be one of them.

Figure 5. *Lincity* is a computerized city simulation that makes you wonder why creating a Utopia is so darn difficult.

with stats on players, kills, personal scores, team scores and more (Figure 4).

Keep an eye on the map to your lower left, as it can alert you to enemy tanks. If you can drive, fire and type at the same time, press N to send a chat message to the group, or M to send one only to your teammates. If you see the boss coming, press F12 to exit the game in a hurry. Just a little joke, *mes amis*. I would never suggest that you play this at work.

The hour is getting late, *mes amis*, but I don't want to leave you with the impression that all the virtual worlds that may exist in our systems are built entirely on destruction and mayhem. You can, in fact, build an entire civilization, including a city, its farms, factories, markets and every other trapping of modern (or premodern) civilization. Download *Lincity* (or check your distribution CDs) and start building. The idea of this highly addictive and time-consuming game is for you to build a city, and in the process, feed and clothe your people, and create jobs so you can build and sustain an economy. Invest in renewable energy as you strive to build a civic Utopia (Figure 5).

As things get better and better, you can save your game and get back to creating this ideal world of yours. Okay, you're right, it's not as easy at it sounds. The clock is ticking, and the months go by fast. Without careful attention, your world may wither away in its own poisons. I should warn you that starting from scratch may be a bit of a confidence destroyer. Why not start when things are good? When the game begins, click the Menu button in the upper left. The main window then provides you with some choices, including one to Load a saved game. The game comes with two: one is called Good Times and the other (you guessed it), Bad Times. I recommend Good Times to get your virtual flippers wet. When you get so good at this that you feel you can fix anything, go for the Bad Times, and see if you can pull your city back from being $25 million in debt.

The clock, *mes amis*, it is telling us that closing time is upon us. With all these sounds of artillery and explosions coming from your workstations, it seems obvious that we will have to stay open just a little longer. François will happily refill your glasses one final time before we say, "*Au revoir*". The games may be all virtual, but the wine is real. It's a good thing too, but I'd hate to have it spilled every time someone fired a shot. On that note, please raise your glasses, *mes amis*, and let us all drink to one another's health. *A votre santé! Bon appétit!*■

Resources for this article: www.linuxjournal.com/article/8882.

Marcel Gagné is an award-winning writer living in Mississauga, Ontario. He is the author of the all-new *Moving to Ubuntu Linux*, his fifth book from Addison-Wesley. He also makes regular television appearances as Call for Help's Linux guy. Marcel is also a pilot, a past Top-40 disc jockey, writes science fiction and fantasy, and folds a mean Origami T-Rex. He can be reached via e-mail at mggagne@salmar.com. You can discover lots of other things (including great Wine links) from his Web site at www.marcelgagne.com.

Big business power, small business price tag.

Run your business on the Processor many of the world's largest companies choose. Robust ZT X9000 series systems powered by the Intel® Xeon® Processor meet your needs today and grow with you into the future.

ZT 2U Server X9557

Intel® Xeon® Processor 3 GHz
(2MB L2 Cache, 800MHz FSB)

- Intel® E7320 Chipset Server Board
- 1GB DDR2 400MHz ECC/ Reg. SDRAM
- 4 x 250GB SATA 16MB Cache Enterprise Hard Drive (RAID 5)
- 6 x 1" Hot-swap SATA Drive Bays
- Slim DVD-ROM and Floppy Drive
- 4 Channel SATA RAID Controller
- Dual Intel® 82541GI Gigabit Ethernet Controllers
- 2U Rackmount Chassis w/ 550W Power Supply
- SuperDoctor III Server Management Software
- 3-Year Limited Warranty

$1,999

ZT 4U/Tower Server X9558

Dual Intel® Xeon® Processors 3 GHz
(2MB L2 Cache, 800MHz FSB)

- Intel® E7320 Chipset Server Board
- 1GB DDR2 400MHz ECC/ Reg. SDRAM (Up to 12GB)
- 2 x 80GB SATA 8MB Cache Hard Drive (RAID 1 for O/S)
- 6 x 400GB SATA 16MB Cache Enterprise Hard Drive (RAID 5 for DATA)
- 8 x 1" Hot-swap SATA Drive Bays
- DVD-RW and Floppy Drive
- 64bit High Performance SATA RAID Controller
- Dual Intel® 82541GI Gigabit Ethernet Controllers
- 4U Rackmount/Tower Chassis w/650W Redundant-Cooling Power Supply
- SuperDoctor III Server Management Software
- 3-Year Limited Warranty

$3,899

ZT Tower Server DC X6011

Intel® Pentium® D Processor 950
(2 X 2MB L2 Cache, 3.40 GHz, 800MHz)

- Intel® D945GPM Chipset Mainboard
- 1GB DDR2 667MHz Dual Channel
- 4 x 250GB SATA 16MB Cache Enterprise Hard Drive (Extendable to 8 HDD)
- 4 Channel SATA RAID Controller (RAID 0,1,5,10 Supports)
- 16x DVD-RW and Floppy Drive
- Intel® Gigabit Ethernet / IEEE1394
- SuperSilent Tower Chassis w/ 500watt Power Supply
- Logitech® Keyboard & Optical Mouse
- 1 Year Limited Warranty

$1,499

Go to ztgroup.com/go/linuxjournal

Call 866- ZTGROUP (866-984-7687)

Promote code : Lj0506

ZT GROUP

DAVE TAYLOR

Counting Cards

Here are some *Blackjack* card-handling routines you can count on.

In my last few columns, we've had a good stab at starting to build a *Blackjack* game within the confines and capabilities of the shell. The last column wrapped up the discussion of how to shuffle an array of 52 integer values and how to unwrap a given card to identify suit and rank so it can be displayed attractively.

This column goes further into the mathematics of *Blackjack*, with a routine that can be given an array of cards and return the numeric value of the hand. If you're a *Blackjack* player though, you'll instantly catch something we're skipping for now. In *Blackjack*, an Ace can be scored as having one point or 11 points, which is how the hand of Ace + King can be a blackjack (that is, worth 21 points).

We'll just count the Ace as being worth 11 points for this first pass through the game, and perhaps later we'll come back and add the nuance of having the Ace be worth one or 11. Note, by the way, that this adds significant complexity, because there are then four ways to score the hand of Ace + Ace (as 2, 12, 12 or 22), so theoretically, the routine that returns the numeric value of a given hand actually should return an array of values.

But, let's start with the straightforward case. Last month, I showed how to extract the rank of a given card with the equation:

```
rank=$(( $card % 13 ))
```

In a typically UNIX way, rank actually ranges from 0–12, rather than the expected 1–13, so because we'd like to leave cards #2–10 in each suit to be the corresponding value, that means we have the rather odd situation where rank 0 = King, rank 1 = Ace, rank 11 = Jack and rank 12 = Queen. No matter, really, because we're going to have to map card rank into numeric values anyway for one or more of the cards—however we slice it.

With that in mind, here's a function that can turn a set of card values into a point value, remembering that all face cards are worth ten points and that, for now, the Ace is worth only 11 points:

```
function handValue
{
   # feed this as many cards as are in the hand
   handvalue=0  # initialize
   for cardvalue
   do
     rankvalue=$(( $cardvalue % 13 ))
     case $rankvalue in
       0|11|12 ) rankvalue=10   ;;
       1       ) rankvalue=11   ;;
     esac

     handvalue=$(( $handvalue + $rankvalue ))
    done
}
```

Let's examine some nuances to this before we go much further. First, notice that the conditional case statements can be pretty sophisticated, so we catch the three situations of rankvalue = 0 (King), rankvalue = 11 (Jack) and rankvalue = 12 (Queen) with a succinct 0|11|12 notation.

What I like even better with this function is that by using the for loop without specifying a looping constraint, the shell automatically steps through all values given to the function and then terminates, meaning we have a nice, flexible function that will work just as well with four or five cards as it would with only two. (It turns out that you can't have more than five cards in a *Blackjack* hand, because if you get five cards and haven't gone over a point value of 21, you have a "five card monty", and it's rather a good hand!)

Invoking this is typically awkward, as are all functions in the shell, because you can't actually return a value and assign it to a variable or include it in an echo statement or something similar. Here's how we can easily calculate the initial point values of the player's hand and the dealer's:

```
handValue ${player[1]} ${player[2]}
echo "Player's hand is worth $handvalue points"

handValue ${dealer[1]} ${dealer[2]}
echo "Dealer's hand is worth $handvalue points"
```

Blackjack is a game that's very much in the dealer's favor, because the player has to take cards and play through the hand before the dealer has to take a single card. There's a significant house advantage for this reason, but in this case, we now can have a loop where we ask players if they want to receive another card (a "hit") or stick with the hand they have (a "stand") by simply keeping track of their cards and invoking handValue after each hit to ensure they haven't exceeded 21 points (a "bust").

To get this working though, we have to restructure some of the code (not an uncommon occurrence as a program evolves). Instead of simply referencing the deck itself, we now have a pair of arrays, one for the player and one for the dealer. To initialize them, we drop the value –1 into each slot (in the initialization function). Then, we deal the hands with:

```
player[1]=${newdeck[1]}
player[2]=${newdeck[3]}
nextplayercard=3            # player starts with two cards

dealer[1]=${newdeck[2]}
dealer[2]=${newdeck[4]}
nextdealercard=3            # dealer also has two cards

nextcard=5                   # we've dealt the first four cards already
```

You can see the tracking variables we need to use to remember how far down the deck we've moved. We don't want to give two players the same card!

With that loop in mind, here's the main player loop:

```
while [ $handvalue -lt 22 ]
do

  echo -n "H)it or S)tand? "
```

```
  read answer
  if [ $answer = "stand" -o $answer = "s" ] ; then
     break
  fi

  player[$nextplayercard]=${newdeck[$nextcard]}

  showCard ${player[$nextplayercard]}

  echo "** You've been dealt: $cardname"

  handValue ${player[1]} ${player[2]} ${player[3]} ${player[4]} ${player[5]}

  nextcard=$(( $nextcard + 1 ))
  nextplayercard=$(( $nextplayercard + 1 ))
done
```

That's the simplified version of this loop. The more sophisticated version can be found on the *Linux Journal* FTP site (**ftp.ssc.com/pub/lj/listings/issue145/8860.tgz**). Notice that it's pretty straightforward. As long as the hand value is less than 22 points, the player can add cards or opt to stand. In the latter case, the break statement pulls you out of the while loop, ready to proceed with the program.

Because nextcard is the pointer into the deck that keeps track of how many cards have been dealt, it needs to be incremented each time a card is dealt, but as we're using nextplayercard to keep track of the individual player array, we also need to increment that each time through the loop too.

Let's look at one simple tweak before we wrap up, however. Instead of merely asking whether the player wants to hit or stand, we can recommend a move by calculating whether the hand value is less than 16:

```
if [ $handvalue -lt 16 ] ; then
   echo -n "H)it or S)tand? (recommended: hit) "
else
           echo -n "H)it or S)tand? (recommended: stand) "
fi
```

Generally, we'll have a quick demo, but notice that we do have some bugs in this script that need to be dealt with first, though:

```
$ blackjack.sh
** You've been dealt: 3 of Clubs, Queen of Clubs
H)it or S)tand? (recommended: hit) h
** You've been dealt: 8 of Hearts
H)it or S)tand? (recommended: stand) s
You stand with a hand value of 21
```

Perfect. And here's another run:

```
$ blackjack.sh
** You've been dealt: 4 of Clubs, Jack of Hearts
H)it or S)tand? (recommended: hit) h
** You've been dealt: 10 of Diamonds

*** Busted!  Your hand is worth 24 **
```

Ah, tough luck on that last one!

Rather than point out specific problems, let me note here that being dealt either of the following two sequences is quite a problem: A A or 2 2 2 2 3 4. Can you see why?

Next month, we'll look at solving these problems!■

Dave Taylor is a 26-year veteran of UNIX, creator of The Elm Mail System, and most recently author of both the best-selling *Wicked Cool Shell Scripts* and *Teach Yourself Unix in 24 Hours*, among his 16 technical books. His main Web site is at www.intuitive.com.

MICK BAUER

Security Features in Debian 3.1

Debian gives you every security feature you need and more, but using these tools can be a daunting task.

Last month, I began a three-part series on distribution-specific security features, beginning with SUSE Linux 10.0. This month, I continue with Debian 3.1, and next month I will conclude with Red Hat Enterprise Linux.

As you may recall, unless you missed last month's column or have been enjoying yourself in memory-impairing ways since then, several things about SUSE 10.0 really struck me: its wide variety of security-enhancing software packages and security-scanning tools; its inclusion of several different virtual machine platforms; and Novell AppArmor, which adds Mandatory Access Controls (MACs) to individual applications and processes.

When I began exploring security features in Debian 3.1 (Sarge) GNU/Linux, I was therefore particularly interested to determine how does Debian 3.1 compare with SUSE 10.0 in those areas? And, what unique security features does Debian bring to the table?

Like SUSE, Debian GNU/Linux is a general-purpose Linux distribution designed to be useful in a wide variety of desktop and server roles. Also like SUSE, Debian includes a long and varied bundle of binary software packages.

Unlike SUSE, Debian is a 100% not-for-profit undertaking. There is no expensive Enterprise version of Debian 3.1 with more features than the freeware version. There's only one version of Debian GNU/Linux 3.1, and it's 100% free—unless you purchase Debian CD-ROMs from a Debian re-packager such as LinuxCentral (see the on-line Resources), in which case you're paying primarily for the cost of CD-ROM production, not for Debian itself.

Arguably, there are security ramifications associated with any purely free software product. Business-oriented IT managers love to ask, "Who's accountable when things go wrong?" But others point to Debian's impressive record of releasing timely security patches as evidence that the Debian Security Team is at least as dependable and responsive as any equivalent commercial entity. My own opinion is that its freeness isn't a major factor one way or the other. Debian doesn't have a reputation for being any more or less secure than commercial general-purpose Linux distributions.

Installing Debian GNU/Linux 3.1

So, what is the Debian installation experience like, and how does it encourage good security?

Compared to other major general-purpose Linux distributions, Debian's installer is decidedly old-school. It uses a bare-bones, text-based GUI that does little more than install software packages. Although this may be off-putting to many users, especially those new to Linux, it minimizes the system resources required to install Debian and the amount of time you'll spend waiting for the installer to load itself into RAM.

Software package installation, as with any Linux distribution, is the heart of the Debian installation process, and in Debian 3.1 it's handled by aptitude. aptitude is similar to its predecessor, dselect, but with a couple of important differences. The first is that although it's text-based like dselect, aptitude sports drop-down menus you can access by pressing the F10 key. The second difference is that, for me at least, aptitude organizes packages in a much less confusing way than dselect. It's still primitive compared to the graphical package installers in SUSE, Red Hat Enterprise Linux and so on (and arguably clunkier than the text-based Slackware installer); however, aptitude is a significant improvement over dselect.

With aptitude, it's also easy to update your local package list and get the latest security patches from the Debian.org site (see Resources). In fact, anytime you install software using the Advanced Packaging Tool (apt) system (for example, when you run aptitude or apt-get), Debian automatically checks for security updates for the packages you're attempting to install.

The bad news about the Debian installer is that it doesn't seem to do very much to harden your system, even in a preliminary way. It doesn't give you an opportunity to create even a basic local firewall policy or choose a preconfigured or default policy. It doesn't even check your root and first nonprivileged-user account passwords for complexity (although it does warn you that passwords *need* to be complex).

Rather, it appears as though in Debian the emphasis is on providing users with as wide a variety of security-related software packages as possible, rather than actually helping users set up any of those packages. Considering that Debian consists of more than 15,000 software packages in all, you've got many choices indeed. Table 1 lists some Debian packages that directly enhance system security.

In addition to the local security-enhancing packages in Table 1, Debian includes many tools for analyzing the security of other systems and networks. Table 2 lists some notable ones.

Sifting through all these packages at installation time can be daunting. One thing that helps is aptitude's ability to search for packages by name. Another is the "Securing Debian Manual" (see Resources).

Once you've selected and installed your initial set of software packages, aptitude runs a few post-installation scripts (depending on what you installed). On my test system, I was disappointed to see very little in these scripts germane to security—these deal primarily with basic system setup, such as network settings. If you need to reconfigure these basic settings later (without editing files in /etc directly), you can re-invoke that part of the installer with the base-config command.

In summary, Debian's installation-time security features are disappointing and sparse. It may not be fair to compare the purely volunteer-driven Debian effort to a commercial product, but in my opinion, Debian sorely needs a centralized, security feature-rich installation and administration utility akin to SUSE's YaST.

servers DIRECT™

GO STRAIGHT TO THE SOURCE!

Cost-efficient performance for high performance computing

Intel® Xeon™ Processor powering ServersDirect Systems provides outstanding price/performance solutions for your HPS needs

$999

SDR-1108T 1U Xeon™ Simple and Affordable Server

Entry-level server solutions for networking, shared Internet access, file and print sharing

- 1U rack designed with 2 x 1" Hot-swap SATA Drive Bays
- Intel® Xeon™ 2.8 Ghz with 800MHz 2MB Cache (Dual Processor Option)
- Intel® E7320 (Lindenhurst VS) Chipset
- Kingston 1024MB DDRII 400 ECC Reg.memory
- 1pcx120GB SATA 7200RPM hard drive
- ATI Rage XL SVGA PCI video controller with 8 MB of video memory
- 6300ESB (Hance Rapids) SATA Controller (2x Drive support)
- RAID 0, and 1 support
- 2x Intel® 82541GI Gigabit Ethernet Controllers

2U Xeon™ Applications Server
SDR-2116T

Optimal solution for mail and messaging, databases, and applications

- Intel Xeon™ 2.8 Ghz with 800MHz 2MB Cache (Dual Processor Option)
- 2U Chassis with 460W Power Supply
- Kingston 1024MB DDRII 400 ECC Reg.memory
- Western Digital 250GB SATA RAID Drive
- 16 x 1" Hot-swap SATA drive bays
- Integrated ATI Rage XL SVGA PCI video controller
- One Intel® 82541PI Gigabit Ethernet controller
- Dual-channel SATA controller supporting RAID 0 and 1

$1199

3U Xeon™ Storage Server
SDR-3309T

Provides optimal solution for Databses and applications

- Intel® Xeon™ 2.8Ghz with 800FSB 2MB Cache (Dual Processor Option)
- 3U Chassis with 650W Redundant Power Supply
- Intel® server board with Intel® E7520 (Lindenhurst) Chipset
- Kingston 1024MB DDR333 ECC Reg. (2x512MB)
- 1pc x Western Digital 250GB SATA RAID Drive
- 16 x 1" Hot-swap SATA drive bays
- Integrated ATI Rage XL SVGA PCI video controller
- Dual Intel® Gigabit Ethernet Controllers
- 2 SATA Channels with RAID 0 and 1

$2,699

4U Enterprise Server
SDR-4305T

Easily scalable storage solution with Hot-swap functionality for growing business

- Intel® Xeon™ Processor 3.0Ghz with 800FSB 2MB Cache (Dual Processor Option)
- 4U Chassis with 16 hot-swap bays & 650W redundant power supply
- Intel® server board with Intel® E7520 (Lindenhurst) Chipset
- Kingston 1024MB DDR333 ECC Reg. (2x512MB)
- 1pc x Western Digital 250GB SATA RAID Drive
- 16 x 1" Hot-swap SATA drive bays
- Integrated ATI Rage XL SVGA PCI video controller
- Dual Intel® Gigabit Ethernet Controllers
- 2 SATA Channels with RAID 0 and 1

$3,499

5U Advanced Storage Server
SDR-5302T

The absolute best storage capacity available in a 5U format, at a incomparable cost/TB ratio

- Intel® Xeon™ Processor 3.0Ghz with 800FSB 2MB Cache (Dual Processor Option)
- 5U Chassis with 24 hot-swap bays & 950W redundant power supply
- Supermicro® server board with Intel® E7520 (Lindenhurst) Chipset
- Kingston 1024MB DDR2 400 ECC Reg. (2x512MB)
- 12pcs x Western Digital 250GB SATA RAID Drive
- 1pc x 3Ware 9550SX-12 port RAID Controller Card
- Integrated ATI Rage XL SVGA PCI video controller
- Dual Intel® 82541PI Single-Port Gigabit Ethernet Controllers

$5,999

SERVERS DIRECT CAN HELP YOU CONFIGURE YOUR NEXT HIGH PERFORMANCE SERVER SYSTEM - CONTACT 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 assemble systems that require a little extra. Servers Direct - your direct source for scalable, cost effective server solutions.

1.877.727.7887 | WWW.SERVERSDIRECT.COM

intel Xeon inside™

Table 1. Some Security-Enhancing Packages in Debian 3.1

Package Name	Description
aide, fam, tripwire, osiris	File/system integrity checkers.
bastille	Excellent, comprehensive and interactive (yet scriptable) hardening utility.
bochs	Bochs virtual x86 PC.
bozohttpd, dhttpd, thttpd	Minimally featured, secure Web server dæmons.
chrootuid, jailer, jailtool, makejail	Utilities for using and creating chroot jails.
clamav	General-purpose virus scanner.
cracklib2, cracklib-runtime	Library and utilities to prevent users from choosing easily guessed passwords.
filtergen, fireflier, firestarter, ferm, fwbuilder, guarddog, mason, shorewall	Tools for generating and managing local firewall policies.
flawfinder, pscan, rats	Scripts that parse source code for security vulnerabilities.
freeradius, freeradius-ldap, etc.	Free radius server, useful for WLANs running WPA.
frox, ftp-proxy	FTP proxies.
gnupg, gnupg2, gpa, gnupg-agent	GNU Privacy Guard (gpg), a versatile and ubiquitous e-mail- and file-encryption utility.
harden, harden-clients, harden-servers, etc.	Actually an empty package containing only scripts that install and un-install *other* packages so as to improve system security.
ipsec-tools, pipsecd, openswan, openswan-modules-source	Tools for building IPSec-based virtual private networks.
libapache-mod-chroot, libapache2-mod-chroot	Apache module to run httpd chrooted without requiring a populated chroot jail.
libapache-mod-security, libapache2-mod-security	Proxies user input and server output for Apache.
oftpd, twoftpd, vsftpd	Minimally featured, secure FTP server dæmons.
privoxy	Privacy-enhancing Web proxy.
psad	Port-scan attack detector.
pyca, tinyca	Certificate authority managers.
selinux-utils, libselinux1	Utilities and shared libraries for SELinux.
slat	Analyzes information flow in SELinux policies.
slapd	OpenLDAP server dæmon.
squidguard	Adds access controls and other security functions to the popular Squid Web proxy.
squidview, srg	Log analyzers for Squid.
syslog-ng	Next-generation syslog dæmon with many more features than standard syslogd.
trustees	Extends file/directory permissions to allow different permissions for different (multiple) groups on a single object.
uml-utilities	User-mode Linux virtual machine engine for Linux guests.

Like other major Linux distributions, Debian increases in size and complexity with each new release. The paradox here is that Debian's ever-growing, almost unparalleled selection of software packages makes it more complex, even to the point of confusion—confusion causes sloppiness; sloppiness introduces avoidable security holes. A central administration utility would go a long way to reduce this confusion and enhance security for Debian neophytes and power users alike. It would go even further if it included modules for creating local firewall policies, managing virtual machines, managing SELinux or Trustees policies and so on.

All ranting aside, I like Debian, and as of this writing, I'm in the process of migrating my Web server from SUSE to Debian (though my laptop will remain a SUSE box). It's also worth mentioning that there are many unofficial Debian installers available, including other Linux distributions based on Debian and able to run Debian packages (see Resources).

So, moving on, let's talk about some particularly interesting and useful groups of security-related packages in Debian GNU/Linux 3.1.

Virtual Machines in Debian

If you want a hypervisor-based virtual machine environment, such as Xen for Debian, you need to obtain and compile source code, though that's not too huge of a barrier. Debian has no Xen packages. Debian does include, however, binary packages for two other general-purpose virtual machine envi-

Table 2. Security Audit Tools in Debian 3.1

Package Name	Description
dsniff, ettercap	Packet sniffers for switched environments.
ethereal, tcpdump	Excellent packet sniffers.
fping	Flood ping (multiple-target ping).
idswakeup	Attack simulator for testing intrusion detection systems (IDSes).
john	John the Ripper, a password-cracking tool (legitimately used for identifying weak passwords).
kismet	Wireless LAN sniffer that supports many wireless cards.
nessus, nessusd, nessus-plugins	Nessus general-purpose security scanner.
nmap	Undisputed king of port scanners.
snort	Outstanding packet sniffer, packet logger and intrusion detection system.

ronments: user-mode Linux (UML) and Bochs. (It also includes Wine, but this is more of a shim for running specific Windows applications than a virtual machine per se.)

Of Debian's two officially supported virtual machines, user-mode Linux is probably the most viable option for using virtual hosts to segregate different application environments, for example, Apache on one virtual machine and BIND9 on another. This is because of performance limitations in Bochs: Bochs emulates every single x86 CPU instruction and all PC devices. Bochs therefore would appear to be more suited to single guest-system applications, such as running Windows applications on your Linux desktop system. The Bochs Project home page (see Resources) includes official documentation and links to mailing lists, discussion boards and so forth. Debian's bochs-doc package also contains Bochs documentation.

User-mode Linux doesn't support Windows guest systems, but it is much faster than Bochs and has the added advantage of running all guest systems' kernels as nonprivileged users (that is, not as root, like the underlying "host" kernel). See Debian's user-mode-linux-doc package for more information. If you run a Debian guest on an underlying Debian host, you may need to install the user-mode-linux package (on the guest) from Debian's unstable release—the stable version is unavailable for some reason.

I must add a disclaimer at this point: I've never used UML myself, being a VMware user of long standing (see my review of VMware Desktop 5.5 on page 56). Therefore, I can't tell you firsthand how to use UML or even how well it works in Debian.

Enhanced Access Controls in Debian

Several packages in Debian GNU/Linux 3.1 enhance local access controls. The trustees package lets you define multiple sets of permissions on a single file/directory/device object by associating a trustee object with it. For example, you can give members of the users group read-only access to the file foo.txt, and give members of the foomasters group write privileges to the same file.

A much more comprehensive set of controls is provided by SELinux, the US National Security Agency's type-enforcement and role-based access control system for the Linux kernel. SELinux makes it possible to manage users, groups and system resources with a very high level of granularity, even to the extent of making it possible to restrict root's own privileges.

The trade-off is complexity. Creating and managing SELinux policies that don't impair needed functionality can be involved. Luckily, besides its standard selinux-utils package, Debian includes checkpolicy, an SELinux policy compiler, and setools, a group of utilities for analyzing SELinux policies and managing users.

If SELinux is more than you're willing to tackle, Debian provides several other tools for delegating root's authority. sudo, of course, is the classic in this category, but there's also osh, the Operator's Shell.

Limited-Feature SSH Packages in Debian

Another interesting category of tools that are well represented in Debian are limited-feature Secure Shell (SSH) tools. SSH, of course, is an encrypted, strongly authenticated means of running remote shells, executing remote commands and even for tunneling other TCP-based network applications including the X Window System. But what if you want to offer users only a subset of SSH functionality—for example, encrypted file transfers, without giving them shell access?

Two Debian packages that address this problem are rssh, which allows users to use scp, rdist, rsync, cvs or sftp over SSH without actual shell access, and scponly, which allows scp without allowing remote shells.

Filesystem Encryption in Debian

The last category of security tools I highlight here is filesystem encryption. These are different from more general-purpose encryption tools, such as gnupg and bcrypt, which are used to encrypt individual files. Filesystem encryption tools let you encrypt entire volumes (directory structures), for example, on USB drives and other removable media.

Three Debian packages that provide filesystem encryption are cryptsetup, which manages loopback-device encryption via the Linux 2.6 kernel's dm-crypt functionality; encfs, which doesn't require use of loopback devices; and lufs-cryptofs, an encryption module for the Linux Userland Filesystem (lufs). Of the three, cryptsetup offers the best performance, because it operates at the kernel level. The user-space filesystems, encfs and lufs, work at a higher layer of abstraction than the kernel—that is, they're less efficient. They're also, however, more useful for networked filesystems.

Debian's Stability

I'd be remiss if I didn't at least briefly discuss one of my favorite characteristics of Debian, and the main reason I'm running it on my new Web server—Debian's relatively glacial release schedule. On the one hand, the delay in releasing Debian 3.1 (three years, or 21 dog/computer years after 3.0) was a bit extreme, and the Debian team has pledged a more predictable release cycle, probably one year from now on. But it's also true that stability enhances security.

Put another way, if you use Debian to run the latest desktop applications, or other things that depend on the very latest hardware drivers, you may be happier with the Debian variant Ubuntu, which has a predictable and short (six-month) release cycle. If, however, you want to build an appliance system that chugs along in a corner, requiring little ongoing maintenance other than regular security patches, Debian's longer release cycle is positively luxurious. In many situations, it's preferable to run somewhat-outdated but fully security-patched applications than it is to have to upgrade the entire operating system every six months (or sooner). I admit, however, that I am among the world's laziest system administrators!

Conclusion

Like UNIX itself, Debian provides the security-minded user with maximal power, flexibility and variety of tools, at the cost of complexity. Debian GNU/Linux 3.1 is probably not for you if you have an aversion to man pages or Google. But it's very flexible indeed. This article scratches only the surface of Debian's potential as a platform for secure server operations or for security scanning and auditing.

Next month, I'll conclude my "Security Features" trilogy with Red Hat Enterprise Linux. Until then, take care!■

Resources for this article: www.linuxjournal.com/article/8885.

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".

Are you

shocked

by the high cost of iSCSI & Fibre Channel storage?

AoE is your answer!

ATA-over-Ethernet = simple, low cost, expandable storage.

www.coraid.com

EtherDrive® SR1520

- RAID enabled 3U appliance with 15 slots for hot swap SATA disks
- Check out our other Storage Appliances and NAS Gateway

Winner
Product Excellence Award
LinuxWorld Conference & Expo
Best Storage Solution

1. Ethernet Storage – without the TCP/IP overhead!
2. Unlimited expandability, at the lowest possible price point!!
3. You want more storage…you just buy more disks – it's that simple!!!

Visit us at www.coraid.com for more information.

CORAID

1.706.548.7200 The Linux Storage People **www.coraid.com**

DEE-ANN LEBLANC

Playing PlayStation Games in Linux

Run your favorite PlayStation games on Linux with PCSX.

This article focuses on Sony PlayStation games and the PCSX PlayStation Emulator. I chose this particular system because you can find PlayStation games both on-line and in game stores, primarily in the Used section.

Getting and Installing PCSX

To get PCSX, point a browser to the Web site (see the on-line Resources), scroll down to the Linux port section, and download the latest build. Once you have the file, change to your download directory. Next, uncompress and then unpackage the file. For example, on the command line inside your download directory, you might type `tar xzvf Lpcsx-1.5.tgz`.

This action creates a directory called Pcsx in your current location (for example, ~/Downloads/Pcsx). Now that you have the main tool unpacked, it's time to download and add plugins.

Getting Plugins and BIOS

PCSX is just a program shell. Plugins provide the functionality you need in order to play your games. To find a good selection, go to the Next-Gen Emulation site (see Resources) and click PlayStation. Along the left-hand side of the PLUGINS section of links, click Linux Plugins to find your options.

The plugins I selected were Pete's XGL2 Linux GPU (video), P.E.O.P.s Linux OSS SPU (sound), CDR Mooby Linux (to use ISO files of my games instead of the CDs) and padJoy. If you want to learn more about any of the plugins, click the home icon next to the entry in the listing. Otherwise, click the disk next to it in order to download the file. Either save them directly into the Plugin subdirectory (for example, ~/Downloads/Pcsx/Plugin), or copy them there once you have them downloaded.

In addition to plugins, you need a PlayStation BIOS. "Need" is a strong word—PCSX comes with a rudimentary BIOS, but many recommend downloading a real PlayStation version for the best game compatibility. It's legally questionable to offer the BIOS content so I won't give you a link. However, reading TheGing's Guide to PlayStation BIOS Images (see Resources) will not only educate you more about PlayStation BIOSes, it will give you a list of versions to try. Enter the name of the version you want to use in a search engine, and you'll find the files soon enough. Save the file into the Bios subdirectory (for example, ~/Downloads/Pcsx/Bios), or move it there once you have it.

Installing the "Easy Stuff"

Some parts are simple to install, and some parts are more difficult. Let's start with the easy ones, beginning with the BIOS. It probably came in a file ending in .zip, so use either your graphical file manager to uncompress it, or type `unzip` *`filename`* to do it by hand (for example, `unzip scph1001.zip`). That's it. It's installed.

Next, we install Pete's XGL2 Linux GPU plugin. As you might guess from the name, if you know much about sound in Linux, this plugin uses the Open Sound System (OSS). If your system doesn't use OSS, you need to install and set it up before your sound will work. Your distribution already may have it in place; see the documentation for details or search your package management system.

The tarball you downloaded for this plugin is in a file similar to gpupetexgl208.tar.gz. Using your preferred method, unpack the file. There is no configuration directory by default, so create Pcsx/cfg (for example, ~/Downloads/Pcsx/cfg). Now, copy the files gpuPeteXGL2.cfg and cfgPeteXGL2 into the cfg directory.

Getting the P.E.O.P.s Linux OSS SPU plugin, whose filename is similar to spupeopsoss108.tar.gz, is a nearly identical process. Unpackage it in Plugin, and then copy spuPeopsOSS.cfg and cfgPeopsOSS into the cfg directory.

Installing CDR Mooby Linux

This plugin can be a bit tougher. The installation can appear to go well and then not work, but there's a quick fix available, so don't worry. CDR Mooby comes in a file similar to cdrmooby2.8.tgz. Unpack this tarball in the Plugin directory. This should be all you need to do. However, if you find later when you start PCSX, you see the error (the program will start anyway, look on the command line):

```
libbz2.so.1.0: cannot open shared object file: No such
file or directory
```

then PCSX is looking in the wrong place for this library. Type one of the following two commands (try locate first, and if it doesn't work, try find):

```
locate libbz2.so.1.0
```

or:

```
find / -name libbz2.so.1.0* 2> /dev/null
```

As an example, your result might include:

```
/usr/lib/libbz2.so.1.0.2
```

If so, notice the difference in the filenames. To make a symbolic link so PCSX can find the library, using the example above, type (as root):

```
ln -s /usr/lib/libbz2.so.1.0.2 /usr/lib/libbz2.so.1.0
```

Adjust what you type accordingly.

Installing padJoy

I've saved the "worst" for last. You don't have to use a game

controller to use PCSX (the keyboard works too), but you may want to use a game controller to get a genuine PlayStation experience.

I say this is the worst because there's more to padJoy than simply installing the plugin. You also have to get your game controller working, but one thing at a time. First, make sure you installed the tools necessary to compile C programming code (such as GCC). You also need the GNOME development tools. In addition, make sure that you have gtk-devel—though it may be called something like gtk+-devel in your package management system.

Once you have everything you need in place, compile the padJoy plugin. The padJoy file you downloaded looks similar to padJoy082.tgz. Unpackage it in the Plugin folder, and it creates its own subdirectory called, not surprisingly, padJoy (for example, ~/Downloads/Pcsx/Plugin/padJoy). Enter padJoy/src (so, for example, ~/Downloads/Pcsx/Plugin/padJoy/src), and type `make`. This command should compile the plugin. If the compilation fails, you may be missing a dependency—hopefully, there are hints available in the output displayed.

You now find the files cfgPadJoy and libpadJoy-0.8.so in the src directory. Copy cfgPadJoy into Pcsx/cfg (so, ~/Downloads/Pcsx/cfg) and libpadJoy-0.8.so into Pcsx/Plugin (so, ~/Downloads/Pcsx/Plugin).

Before you proceed, consider the game controller you intend to use with padJoy. Do you already own one? Is it digital or analog? Does it have a connector that can attach to your computer, such as USB? Does it require a game port, and do you have one? (Check your sound card if you aren't sure.) Does it have its own funky connector? If you own an Xbox controller already (not the Xbox 360, which uses USB, but the original Xbox), you can go to Dan Gray's site (see Resources) and read how to use a bit of soldering to convert the controller's connector to use USB—use these instructions at your own risk, of course. If you own another type of controller with a proprietary connector, you can usually purchase a third-party converter on-line.

I tried two different controllers with PCSX. First, I dug around and found a joystick that connects to a computer's game port. Then I discovered that my SoundBlaster Live! card has a game port. The first thing I noticed is that the joystick devices didn't exist by default on my system (look for /dev/js0 and/or /dev/input/js0, these are often symlinked together); however, that's because my distribution uses devfs and creates only the devices it needs at the time. All I had to do was become the root user and type the following two commands:

```
modprobe analog
modprobe joydev
```

Then, when I typed `ls /dev/j* /dev/input/j*`, I found that the device /dev/input/js0 had been created, showing that the system found my joystick. If you think that you have everything set up properly and are just missing the device file, type `mknod /dev/input/js0 c 13 0` to create it. To test your joystick (or gamepad, or whatever you're using), you need the joystick tools installed if they aren't already. Then, type `jstest /dev/input/js0` (adjusting the path for your driver file). You should see output such as:

```
Joystick (Analog 3-axis 4-button joystick) has 3 axes and 4 buttons. Driver version is 2.1.0.
Testing ... (interrupt to exit)
Axes:  0:     0  1:     0  2:-22892 Buttons:  0:off  1:off  2:off  3:off
```

If you see this, it's a good sign. Move the joystick controller around and press some buttons. The numbers should change and the button positions should change. If this happens, you're ready to move on. Press Ctrl-C to get out of the tool. If you see an error message or nothing, the joystick isn't being recognized. You can find a list of all supported input hardware on SourceForge (see Resources). It is often possible to get third-party converters that allow you to hook up game console controllers such as

PlayStation 2 gamepads. Typically, if you can attach a gamepad through USB, you can use it.

If you have an original Xbox controller, you can modify it to connect to regular USB (again, see Dan Gray's site for details); however, it will no longer be usable with your Xbox after that. Xbox 360 controllers, on the other hand, have USB connectors. Gentoo users can turn to the Gentoo Wiki (see Resources) for more information on using the Xbox 360 controller. Users of other distributions can as well, but will have to adjust their instructions for their versions of Linux. For example, they will have to learn how to build a kernel from scratch if their kernel's xpad driver isn't as new as the one linked to from the Gentoo site (the driver for Fedora Core 4's kernel 2.6.14-1.1656_FC4-i686 was far older at version .5 compared to the 1.6 of the version that supports the 360 controller, so you likely will need to update). Those using Xbox controllers will need the xpad driver. Because they are USB controllers, your system will load the driver for you when you plug in the gamepad—if the pad is properly recognized. The same jstest program works here as well.

Once you're (relatively) sure you have your hardware working and all of your plugins properly installed, you can finally move on to configuring your emulation software.

Configuring PCSX and Plugins

PCSX is just a core program. It requires plugins to do anything, and you already have these installed. To configure the plugins, change to the directory you created when you unpacked the files—for example, ~/Downloads/Pcsx. From there, run the program from the command line by typing `./pcsx`.

A Pcsx Msg dialog appears, telling you to configure the program. Click OK to open the PCSX Configuration dialog (Figure 1). Many of these dialog boxes don't need you to do anything unless you have a specialty in sound or graphics and like to tweak things, so I will skip to those that are essential.

The main dialog box to configure is the Pad section, so click Configure under Pad 1 (Figure 2).

Next to Emulation, click the PCSX radio button. If your controller is analog, check the analog check box as well. From here, you can click the various buttons to change what they map to. With a joystick, for example, you might click the up arrow in the left cluster of four and then press the joystick handle forward. You can set some of these buttons to map to your keyboard as well, but your keyboard options are limited, so try to keep most on the controller. If you're into creating macros, use the M buttons on the bottom.

Click OK when you are finished mapping the keys. If you have two controllers, click the Configure button under Pad 2 and repeat the process for the second one—make sure to change its device listing; you can use jstest to confirm for yourself which pad is js0 and which pad is js1. After setting up the controllers, you need to tell PCSX which BIOS to use. Otherwise, under the BIOS section, select the BIOS that you downloaded. To do so, find the Bios section and use the drop-down list box to choose the BIOS in the listing.

Your configuration is now complete. Click OK, if all went well, and PCSX starts (Figure 3).

Figure 1. The Main PCSX Configuration Dialog

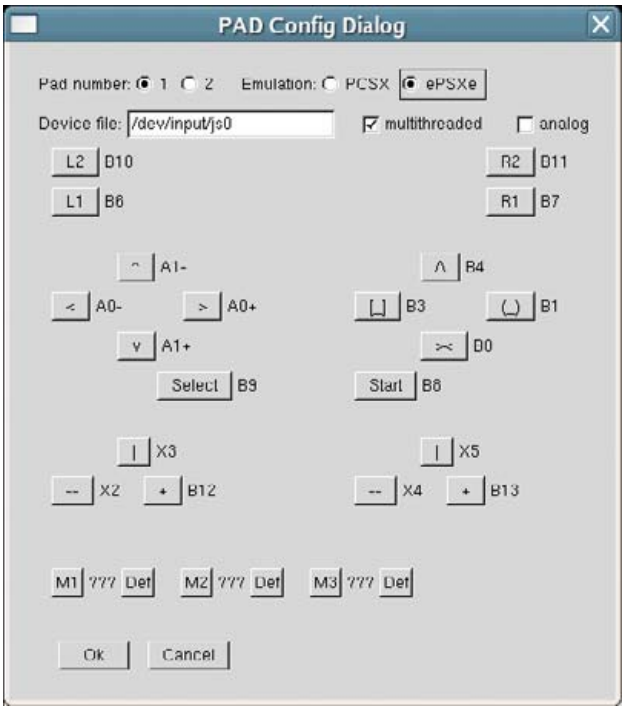


Figure 2. The PAD Config Dialog

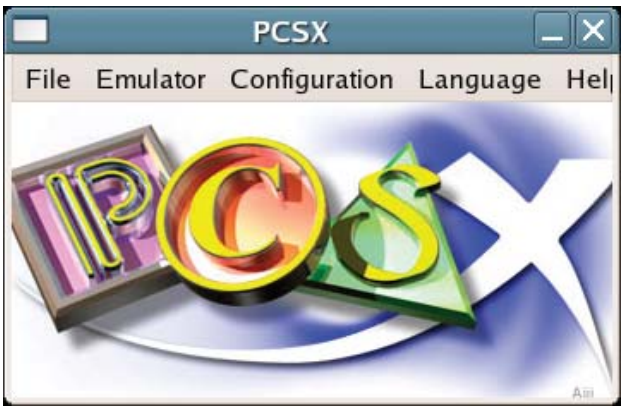


Figure 3. The Main PCSX Window

Preparing to Play

PCSX can't read a PlayStation CD-ROM directly unless you use a different plug-in than the one I chose. Don't despair. I chose the different plugin for a reason. It is, in fact, much faster for game play if you create an image of the game CD-ROM(s) and store them on your hard drive. You can't use most standard tools to do this, however, because there are many little issues in the way (see the Mega Games site if you're interested). Instead, use `cdrdao` to build an ISO file from the CD's raw content. For many, the command will look like this:

```
cdrdao read-cd –read-raw –datafile frogger.bin
➥–device ATAPI:0,0,0 –driver generic-mmc-raw frogger.toc
```

where frogger.bin is the data file to create (the CD you will select when it comes time to play), and frogger.toc is the table of contents file to create. Both of these files are named after the game, so I easily can tell which one I want to choose. The ATAPI:0,0,0 entry will work for most CD-ROM drives.

Finally, Playing a Game

Yes, it's been a long haul, but you finally can attempt to play a game. I'll warn you right now that not all games will work. *Frogger* worked immediately, but I'm still fussing with *Final Fantasy VII*, which is, of course, a more complex game.

Start PCSX just as you did earlier: enter the Pcsx directory and type `./pcsx`. This time, only the PCSX dialog appears. If you need to, use the Configuration menu to adjust your settings. When you're ready to play, select File→Run CD and then navigate to where you stored your .bin and .toc files. Select the .bin file for the game, and click OK (Figure 4). It might take a bit of practice to figure out your control setup, but it gets easier.

Figure 4. *Frogger* for the Original PlayStation Running in PCSX

Make sure you're not running something that hogs processor time or RAM in the background. You can watch for this by opening a terminal window and typing `top` to open the process monitor. You may find that trying to make the game window larger doesn't work and, in fact, even crashes your machine. If you want to run a game through the specified BIOS, choose File→Run CD Through BIOS. This action might convince some touchy games to play.■

Resources for this article: www.linuxjournal.com/article/8888.

Dee-Ann LeBlanc (dee-ann.blog-city.com) is an award-winning technical writer and journalist specializing in Linux and miniature huskies. She welcomes comments sent to dee@renaissoft.com.

JON "MADDOG" HALL

Tales from the Beach

Maddog introduces his new column with liquid poetry in motion.

I have always loved the ocean, and by very definition, the beach. Jimmy Buffet was always my favorite singer, not just because of the music, which mostly told of fun and carefree days, but because of his music's relation with the sea and the beach. Of course, I like organ music too, and cathedrals, but anyone who is not swayed by Jimmy's magic in my book is a little bizarre.

When I was living in California, I would often go to the Santa Cruz Boardwalk to people watch. Santa Cruz was the first place I took Linus Torvalds and his young family when they moved to California. It is the stuff of the Beach Boys and surfer living, but families go there too to relax and watch the sea lions. Few people get angry at the beach. I also enjoy going to Florianopolis, Brazil, every year during our winter to attend OpenBeach—a bunch of geeks and their loved ones sitting around, sunning and talking about Software Livre!

Sailors also like beaches, and I like sailing. Not "racing sailing" (sorry, Don Becker of Beowulf fame) but "cruising sailing", with a cooler of cold drinks in the cockpit and one hand on the helm, your friends sunning themselves on the foredeck. Sailing your boat into a quiet cove, dropping anchor in the shallow water and diving overboard for a swim to a small island restaurant on the beach. "No shoes, no shirt...no problem!" Bathing suits are overdress.

Many late nights (and early mornings) in Bermuda, Tortola, St. Johns, Veracruz and "Floropa" with friends at various "Pirate Bars" (and you know who you are!). Reggae or Latino music rules in those spots (although good ole rock and roll also works), as young bodies hop to the beat and old men and women hold hands and watch, remembering when they were young. Beach restaurants, music everywhere...Gilberto Gil seems to Creatively weave a Commons ground (a tip of the hat to Lawrence Lessig). I wish the United States had a minister of culture like Mr. Gil.

Blue penguins (also known as fairy penguins) are so small they come ashore only at dusk and in waves, so the predators that typically swoop from the sky will be confused at the numbers. They waddle madly for the cover of the tall beachhead grass and their nests, where they are safe.

There is, of course, a darker side to the beach. It is where the full force of the ocean meets the earth. Victims of hurricanes and of the Tsunami in Asia understand that all too well. It is also the spot that a lot of the fiercest battles were waged, as armies of men tried to come ashore in times of war, with little cover from enemy weaponry.

And in these days, development of the beachhead leaves many without access. People with money and power buy the land and close it off so others cannot access it. To be fair, some people who want to use the beach do not treat it well, leaving garbage and glass where they should not. The people who close off the beach say it is "to protect their property" or to "protect their privacy", but it still limits the resource.

The beachhead should be available for everyone, for there is only a limited amount of beaches on the planet, and everyone should be able to enjoy them.

So this column is named Beachhead, and it describes me (think "Parrot Head" and you will understand), the beach itself and a frame of mind. I hope that sometimes it brings you joy and fun, like walking barefoot on a hot day down the beach with your best friend, waves splashing over your feet and a cold drink in your hand, watching for the penguins.

Some days it will not be so nice, casting a storm warning. Some may stay, not believing the warning. Others may evacuate, fearing the tides will be too high, and others will batten down, knowing that the seawall may go, but unless they are there with the sandbags, disaster will certainly happen. Free and Open Source people that I have met always have sandbags.

And some days there will be news of the Tsunami, and it will hurt, but we know that life will go on and renew. We have friends to help us.

Welcome to the Beachhead.■

Jon "maddog" Hall is the Executive Director of Linux International (www.li.org), a non-profit association of end users who wish to support and promote the Linux operating system. During his career in commercial computing, which started in 1969, Mr Hall has been a programmer, systems designer, systems administrator, product manager, technical marketing manager and educator. He has worked for such companies as Western Electric Corporation, Aetna Life and Casualty, Bell Laboratories, Digital Equipment Corporation, VA Linux Systems and SGI. He is now an independent consultant in Free and Open Source Software (FOSS) Business and Technical issues.

DOC SEARLS

Linus Takes a Pass on the New GPL Draft

The "Readers Digest" version of the Linux Torvalds vs. GPLv3 controversy.

In January 2006, the Free Software Foundation (FSF) released in draft form (**gplv3.fsf.org/draft**) a revised and updated version of the GNU General Public License (GPL). Linux has grown for 14 years under version 2 of the GPL. Naturally, there was immediate interest in how the new draft would go down with Linus Torvalds, who created Linux and chose the original license.

Linus chose to keep Linux licensed under GPLv2.

How and why Linus made that choice is made clear by exchanges on the Linux-Kernel Mailing List (LKML) between January 25, 2006 and February 2, 2006, mostly on the thread "GPL V3 and Linux—Dead Copyright Holders". It's such an interesting read, I decided to excerpt and quote the most relevant sections here.

On Wednesday, January 25, 2006, Chase Venters wrote:

> This means that when the code went GPLv1→GPLv2, the transition was permissible. Linux v1.0 shipped with the GPLv2. It did not ship with a separate clause specifying that "You may only use *this* version of the GPL" as it now does. (I haven't done any research to find out when this clause was added, but it was after the transition to v2.)

Here's how Linus first weighed in:

> Bzzt. Look closer.
>
> The Linux kernel has *always* been under the GPLv2. Nothing else has ever been valid.
>
> The "version 2 of the License, or (at your option) any later version" language in the GPL copying file is not—and has never been—part of the actual License itself. It's part of the *explanatory* text that talks about how to apply the license to your program, and it says that *if* you want to accept any later versions of the GPL, you can state so in your source code.
>
> The Linux kernel has never stated that in general. Some authors have chosen to use the suggested FSF boilerplate (including the "any later version" language), but the kernel in general never has.
>
> In other words: the *default* license strategy is always just the particular version of the GPL that accompanies a project. If you want to license a program under *any* later version of the GPL, you have to state so explicitly. Linux never did.
>
> So, the extra blurb at the top of the COPYING file in the kernel source tree was added not to *change* the license, but to *clarify* these points so that there wouldn't be any confusion.
>
> The Linux kernel is under the GPL version 2. Not anything else. Some individual files are licensable under v3, but not the kernel in general.
>
> And quite frankly, I don't see that changing. I think it's insane to require people to make their private signing keys available, for example. I wouldn't do it. So I don't think the GPLv3 conversion is going to happen for the kernel, since I personally don't want to convert any of my code.

The thread then continued:

> If a migration to v3 were to occur, the only potential hairball I see is if someone objected on the grounds that they contributed code to a version of the kernel Linus had marked as "GPLv2 Only". IANAL.

And, Linus responded:

> No. You think "v2 or later" is the default. It's not. The *default* is to not allow conversion.
>
> Conversion isn't going to happen.

Here's how Linus put it in a summary posting on January 27, 2006:

> Linux has been v2-only for a *long* time, long before there was even any choice of licenses. That explicit "v2 only" thing was there at least for 2.4.0, which is more than five years ago. So this is not some sudden reaction to the current release of GPLv3. This has been there quite *independently* of the current GPLv3 discussion.
>
> If you disagree with code you write, you can (and always have been able to) say so, and dual-license in many different ways, including using the "or later version" language. But that doesn't change the fact that others (a *lot* of others) have been very much aware of the "v2 only" rule for the kernel, and that most of the Linux kernel sources are under that rule.
>
> People argue that Linux hasn't specified a version, and that by virtue of paragraph 9, you'd be able to choose any version you like. I disagree. Linux has always specified the version: I don't put the license in the source code, the source code just says: "Copyright (C) 1991–2002 Linus Torvalds", and the license is in the COPYING file, which has ALWAYS been v2. Even before (for clarification reasons) it explicitly said so.
>
> In other words, that "if no version is mentioned" simply isn't

even an argument. That's like arguing that "if no license is mentioned, it's under any license you want", which is crap. If no license is mentioned, you don't have any license at all to use it. The license AND VERSION has always been very much explicit: linux/COPYING has been there since 1992, and it's been the *version_2* of the license since day 1.

People can argue against that any way they like. In the end, the only way you can *really* argue against it is in court. Last I saw, intentions mattered more than any legalistic sophistry. The fact that Linux has been distributed with a specific version of the GPL is a big damn clue, and the fact that I have made my intentions very clear over several years is another HUGE clue.

I don't see any real upsides to GPLv3, and I do see potential downsides. Things that have been valid under v2 are no longer valid under v3, so changing the license has real downsides.

Quite frankly, *if* we ever change to GPLv3, it's going to be because somebody convinces me and other copyright holders to add the "or any later license" to all files, just because v3 really is so much better. It doesn't seem likely, but hey, if somebody shows that the GPLv2 is unconstitutional (hah!), maybe something like that happens.

So I'm not *entirely* dismissing an upgrade, but quite frankly,

> to upgrade would be a huge issue. Not just I, but others that have worked on Linux during the last five to ten years would have to agree on it. In contrast, staying with GPLv2 is a no-brainer: we've used it for almost 15 years, and it's worked fine, and nobody needs any convincing.
>
> And that really is a big issue: GPLv2 is a perfectly fine license. It has worked well for us for 14 years; nothing really changed with the introduction of GPLv3. The fact that there is a newer license to choose from doesn't detract from the older ones.

A major sticking point for Linus is fresh language in GPLv3 that, to quote the license draft, "intrinsically disfavors technical attempts to restrict users' freedom to copy, modify, and share copyrighted works".

Returning to the thread, on Wednesday, February 1, 2006, Karim Yaghmour wrote:

> DRM is something worth fighting, but we need something that attacks the root problem, not its symptoms. In comparison, GPLv2 was indeed successful in that it attacked the root problem of software distribution freedom. How it may leverage that by introducing restrictions on symptoms of another problem still evades me.

Linus responded:

> Side note: the reason GPLv2 is so successful at fighting the root problem of using copyright to fight restrictive copyrights is that it makes "interesting material" available under a license that forbids further restricting it.
>
> I would suggest that anybody who wants to fight DRM practices seriously look at the equivalent angle. If you create interesting content, you can forbid that *content* ever be encrypted or limited.
>
> In other words, I personally think that the anti-DRM clause is much more sensible in the context of the Creative Commons licenses, than it is in software licenses. If you create valuable and useful content that other people want to be able to use (catchy tunes, funny animation, good icons), I would suggest you protect that *content* by saying that it cannot be used in any content-protection schemes.
>
> Afaik, all the Creative Commons licenses already require that you can't use technological measures to restrict the rights you give with the CC licenses. The "Share Alike" license in particular requires all work based on it also to be shared alike—that is, it has the "GPL feel" to it.
>
> If enough interesting content is licensed that way, DRM eventually becomes marginalized. Yes, it takes decades, but that's really no different at all from how the GPL works. The GPL has taken decades, and it hasn't "marginalized" commercial proprietary software yet, but it's gotten to the point where fewer people at least *worry* about it.
>
> As long as you expect Disney to feed your brain and just sit there on your couch, Disney & Co. will always be able to control the content you see. DRM is the smallest part of it—the crap we see and hear every day (regardless of any protection) is a much bigger issue.
>
> The GPL already requires source code (that is, non-protected content). So the GPL already *does* have an anti-DRM clause as far as the *software* is concerned. If you want to fight DRM on non-software fronts, you need to create non-software content, and fight it *there*.
>
> I realize that programmers are bad at content creation. So many programmers feel that they can't fight DRM that way. Tough. Spread the word instead. Don't try to fight DRM the wrong way.

In a February 2, 2006 post, Linus replied to a suggestion that GPLv2 is itself deficient at fighting DRM. Here's the suggestion:

> The point is not only getting access to the source code, but also being able to change it. Being able to freely study the code is only half of the beauty of the GPL. The other half, being able to change it, can be very effectively stopped using DRM.

And, Linus' reply:

> No it cannot.
>
> Sure, DRM may mean that you cannot *install* or *run* your changes on other people's hardware. But, it in no way changes the fact that you got all the source code, and you can make changes (and use their changes) to it. That requirement has always been there, even with plain GPLv2. You have the source.
>
> The difference? The hardware may run only signed kernels. The fact that the hardware is closed is a *hardware* license issue. Not a software license issue. I'd suggest you take it up with your hardware vendor, and quite possibly just decide not to buy the hardware. Vote with your feet. Join the OpenCores groups. Make your own FPGAs.
>
> And it's important to realize that signed kernels that you can't run in modified form under certain circumstances is not at all a bad idea in many cases.
>
> For example, distributions signing the kernel modules (that are distributed under the GPL) that *they* have compiled, and having their kernels either refuse to load them entirely (under a "secure policy") or marking the resulting kernel as "tainted" (under a "less secure" policy) is a GOOD THING.
>
> Notice how the current GPLv3 draft pretty clearly says that Red Hat would have to distribute its private keys so that people sign their own versions of the modules they recompile, in order to re-create their own versions of the signed binaries that Red Hat creates. That's INSANE.
>
> Btw, what about signed RPM archives? How well do you think a secure auto-updater would work if it cannot trust digital signatures?
>
> I think a lot of people may find that the GPLv3 "anti-DRM" measures aren't all that wonderful after all.
>
> Because digital signatures and cryptography aren't just "bad DRM". They very much are "good security" too.
>
> Babies and bathwater....

Then, also on February 2, 2006, Pierre Ossman wrote:

> So taking open software and closed hardware and combining it into something that I cannot modify is okay by you?

Linus responded:

> But you CAN modify the software part of it. You can run it on other hardware. It boils down to this: we wrote the software. That's the only part *I* care about, and perhaps (at least to me) more importantly, because it's the only part we created, it's the only part I feel we have a moral right to control.
>
> I *literally* feel that we do not—as software developers—have the moral right to enforce our rules on hardware manufacturers. We are not crusaders, trying to force people to bow to our superior God. We are trying to show others that cooperation and openness works better.
>
> That's my standpoint, at least. Always has been. It's the reason I chose the GPL in the first place (and it's the exact same reason that I wrote the original Linux copyright license). I do *software*, and I license *software*.
>
> And I realize that others don't always agree with me.

Reclaim lost time

Levanta Intrepid ™

The world's first Linux management appliance

Plug the Levanta Intrepid ™ into your network and perform the most important Linux management tasks in a fraction of the time you spend now. And gain power and flexibility that you've never had before:

- **Fast & Portable:** Provision servers or workstations practically anywhere, anytime – in minutes. Swap them around, mix it up.
- **Flexible:** Supports commodity hardware, blades, virtual machines, and even mainframes.
- **Out of the Box:** Includes pre-defined templates for servers, workstations, & software stacks. Or create your own.
- **Total Control:** Track any file changes, by any means, at any time. And undo them at will.
- **Disaster Recovery:** Bring dead machines quickly back to life, even if they're unbootable.

Based upon technology that's already been proven in Fortune 500 enterprise data centers. Now available in a box, priced for smaller environments. **Just plug it in and go.**

Check Out the Flash Demo at:

www.levanta.com/demo-lj

LEVANTA®
www.levanta.com
1.877.LEVANTA

WINNER
LINUXWORLD CONFERENCE & EXPO
Most Innovative Hardware Solution

> That's fine. You don't have to. But I licensed my project under a license *I* agreed with, which is the GPLv2. Others who feel differently can license under their own licenses. Including, very much, the GPLv3.
>
> I'm not arguing against the GPLv3.
>
> I'm arguing that the GPLv3 is wrong for *me*, and it's not the license I ever chose.

And, also on February 2, 2006, Pierre Ossman continued:

> Then I have to ask, why GPL and not a BSD license? GPL is after all, forcing our beliefs onto anyone who wishes to benefit from our work.

In response to this post, his last post on this thread, Linus unpacked his differences with the Free Software Foundation:

> Yes, a lot of people see the GPL as a "crusading" license, and I think that's partly because the FSF really has been acting like a crusader.
>
> But I think that one of the main reasons Linux has been successful is that I don't think the Linux community really is into crusading (some small parts of it are, but it's not the main reason). I think Linux has made the GPL more "socially acceptable", by being a hell of a lot less religious about it than the FSF was.
>
> So to me, the point of the GPL is not the "convert the infidels" logic, but something totally different—"quid pro quo".
>
> This is where I started out. My initial reason for my original license (which was also "you must make changes available under the same license") was not crusading, but simple reciprocity. I give out source code—you can use it if you reciprocate.
>
> In other words, to me, the GPL "give back source" is an issue of fairness. I don't ask for anything more than I give. I ask for source code and the ability to incorporate your changes back into *my* use, but I don't want to limit *your* use in any way.
>
> So in my world view—not as a crusader—the GPLv2 is *fair*. It asks others to give back exactly what I myself offer: the source code to play with. I don't ask for control over their other projects (be they hardware or software), and I don't ask for control over copyrights (in the kernel, people are *encouraged* to keep their copyrights, rather than sign them over to me).
>
> I only ask for exact reciprocity of what I give: the license for me to use freely the changes to source code that I initiated.
>
> The GPLv3 fundamentally changes that balance, in my opinion. It asks for more than it gives. It no longer asks for just source back, it asks for *control* over whatever system you used the source in.
>
> See? I think the GPLv3 makes *perfect* sense as a conversion tool. But as a "please reciprocate in kind" tool, the GPLv2 is better.
>
> Now, my very earliest original license (and the GPLv2) fit my notion of reciprocity, and as mentioned, that was the reason I "originally" selected that over the BSD license. However, over time, having seen how things evolve, I've come to appreciate another aspect of the GPLv2, which is why I would never put a project I personally really cared about under the BSD license—encouraging merging.
>
> I've come to believe that the BSD license is not a "sustainable" license, because while it encourages (and allows) forking even more than the GPL does, it does not encourage merging the forks back.
>
> And I've come to the private conclusion that the real value of a fork is lost if you don't have the ability to merge back the end result. Not that all forks should be merged back—most forks are dead ends—but the firm ability to merge back *if* it turns out to be something other than a dead end.
>
> The GPL guarantees you the right to both fork *and* merge the result back—equally, and on both sides. That makes it sustainable. In contrast, the BSD license encourages forking, but also allows for not merging back, and that means that if the project ever gets to the point where there are economic or political reasons to diverge, it *will* eventually diverge, and there is no counteracting force at all.
>
> Now, not all projects have any economic incentives to diverge. There are good reasons to stay on one base, and the costs of forking are bigger than the advantages. So projects like Apache and Sendmail have worked fine—the pain of being different (when you're "just" a network service) is generally much higher than the gain of differentiation.
>
> But just about anywhere else, the "cohesion" of a BSD-licensed project is just lower. You'll have people make a commercial release of it, and they'll spend a bit more effort on it, and eventually the original freely licensed project will become immaterial.
>
> So long-term, I believe that a GPL'd project is stabler. I believe, for example, the fact that Wine switched over to the LGPL (which shares a lot of the cohesion argument) was a very important decision for the project, and that it would eventually have otherwise become irrelevant, and the commercial users of the BSD-licensed code would have taken over. But note that my second reason is not why I *began* using the GPLv2, and that it's also equally true of the GPLv3 and LGPL. Anyway, there are other reasons I like the GPLv2. It's a "known entity" and it's been around for a long time.
>
> In fact, even in -92, when I switched to the GPL, that "known factor" part was a major secondary reason for switching. I could have tried to just change my own license—but I felt it was an advantage to be something that people knew about, and not have to explain it and check it with lawyers. The fact that the GPLv2 was still "young" back then was nothing compared to how wet behind the ears my *own* license was.

What we have here is an excellent look at just how conservative and practical Linus, Linux and open-source development all are.■

Doc Searls is Senior Editor of *Linux Journal*.

Expand Your HPC Flexibility.

Enjoy great dual-core performance plus the flexibility to run 32-bit and 64-bit applications with Dual-Core Intel® Xeon® Processors in your AMAX HPC Servers.

Amax Systems, with leading edge Dual Core and Low Voltage Processing Technologies, are widely deployed by many data centers throughout the nation. Our ISO9001-Certified Burn-in Testing procedures, highly skilled engineers and best of breed platforms provide customers with reliable and flexible rack solutions, ranging from 16U to 42U, maximizing performance and cost efficiency!

To learn more about AMAX HPC Servers, visit www.amax.com.

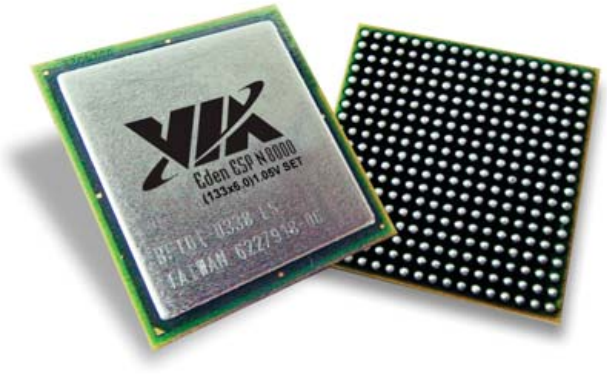
See us at Las Vegas Interop May 2 - May 4 Booth #2339

Toll Free: 1.866.626.2948 ext.0506 | WWW.AMAX.COM

AMAX Information Technologies
An ISO-9001 Certified Company

OpenVZ Project's Virtuozzo

Just as winter turns to spring each year, and Chicago Cubs fans prepare for another year of cruel disappointment, so too does this month bring another virtualization product announcement. The OpenVZ Project has a new beta, based on the Linux 2.6.15 kernel. New in this release is better hardware support (notably for AMD dual-core processors), resizing of ext3 filesystems and improved memory management. OpenVZ is the community face of the Virtuozzo commercial virtualization product, and those wanting to download or contribute to the project should visit the OpenVZ Web site.
openvz.org

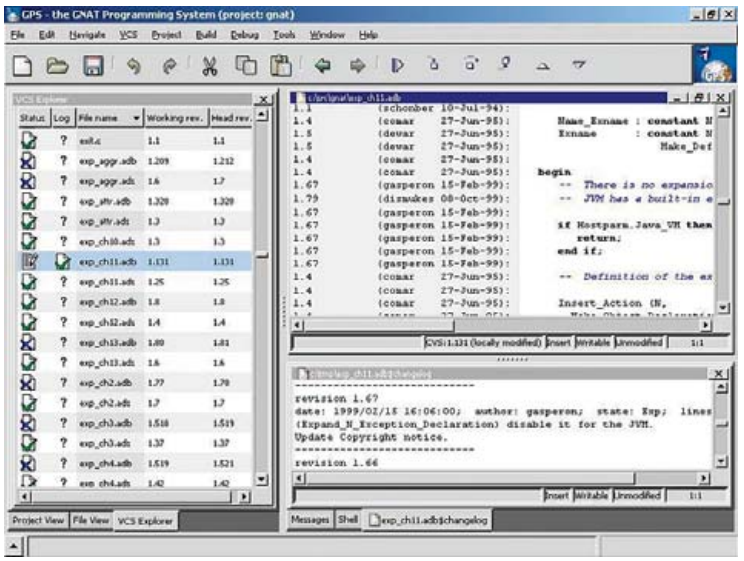


VIA Technologies' Eden and Eden ULV Processors

You're watching *The Princess Bride* on your DVR, and Wesley is about to kiss Buttercup. There's a blissful moment of silence, but it's pierced by the whirr of those pesky cooling fans. VIA Technologies wishes to save you such agony, and to that end has released the fanless Eden and Eden ULV processors. Boasting 90nm manufacturing technology and sipping a meager 3.5 watts for their 1GHz ULV processor, it's just the ticket for an application requiring either low power or low heat specs. For something with a little more umph, the 1.5GHz version is still a power miser, consuming only 7 watts. And Linux is fully supported, of course.
www.via.com.tw

AdaCore's GNAT Programming Studio

Your editor has fond memories of developing an Ada compiler in Pascal to generate 6502 machine code in college. My therapist says the trauma will eventually fade. For those looking for a more modern take on Ada, AdaCore has a new version of its GNAT Programming Studio, now available for x86-64-bit versions of Linux. In addition to ADA, that old DOD favorite, GPS also supports C and C++ for cross-language development. GPS is part of the GNAT Pro Ada Development Environment, and subscriptions start at $14,000 US.
www.adacore.com



BakBone's NetVault

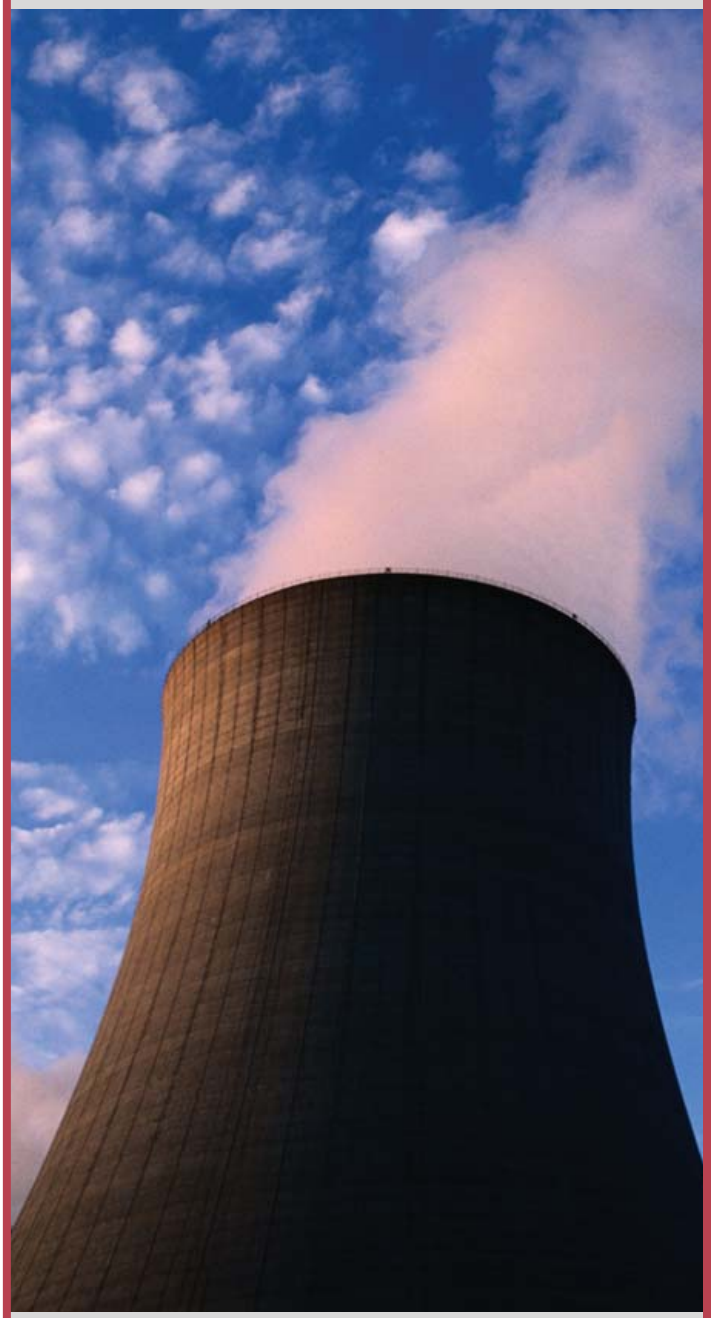
If you truly love your data, let it run away. If it doesn't come back, restore it from backups using BakBone's NetVault. The recently released Enterprise Edition 7.4 can manage all your troublesome backup needs, even in a heterogeneous platform environment (that is, you're stuck backing up your NT servers as well as your Linux and Solaris boxes). The latest release adds support for backing up VMware ESX server environments, allowing guest systems to access tape drives installed on the host. Pricing begins at $1,195 US for Intel-based systems.
www.bakbone.com

Please send information about releases of Linux-related products to newproducts@ssc.com or New Products c/o *Linux Journal*, 1752 NW Market Street, #200, Seattle, WA 98107. Submissions are edited for length and content.

Data center heading for a meltdown?

Time to try DC powered servers from Rackable Systems

Our award-winning servers and storage products are designed to reduce power consumption and increase efficiency in any data center. Customers deploying Rackable Systems servers with DC power and the AMD Opteron™ Processor HE reduce heat output and lower power costs by as much as 67%. And with density levels as high as 88 dual-core servers per cabinet, Rackable Systems solutions can help transform your data center into a highly efficient, power-friendly environment.



To learn more about Rackable Systems' broad range of server and storage offerings, visit www.rackable.com/savepower.

AMD 64 Opteron

Rackable® systems

408.240.8300 www.rackable.com

SOFTWARE

VMware Workstation 5.5 for Linux Hosts

Is VMware a compelling purchase in the face of free virtualization competition? MICK BAUER

Few virtual computer environments are as stable, popular and rich in features as VMware. I've been a fan and user of VMware Workstation since version 2.0. I use it for testing network applications, illicitly running Linux in Windows-only environments and, most recently, for testing the sample code in my book *Linux Server Security*, 2nd ed., across different Linux distributions. (I also wrote most of that edition using MS Word running on a virtual Windows XP machine!) [Do you really want to admit that?—Ed.]

VMware has some serious competition nowadays in the Open Source community. Xen, FAUmachine and user-mode Linux are promising and 100%-free PC virtualization environments. Nevertheless, VMware Workstation 5.5 remains a compelling purchase in the face of all this competition.

Overview and Specifications

VMware Workstation is a user-space application (aided by a couple of proprietary kernel modules) that creates virtual x86-based computers on top of your physical 32-bit or 64-bit x86-based "host" computer.

VMware Workstation 5.5 runs on the following host operating systems:

- Mandrake Linux 10 and 9.0.
- Red Hat Enterprise Linux AS/ES/WS 4.0, 3.0 and 2.1, 32- and 64-bit.
- Red Hat Linux 9.0, 8.0, 7.3 and 7.2.

In VMware parlance, host refers to the system running VMware software. Guest systems are virtual machines running on the VMware host. For the remainder of this review, I use the terms guest system and virtual machine interchangeably.

- SUSE Linux 10.0 and 9.3, 32- and 64-bit.
- SUSE Linux 9.2, 9.1, 9.0, 8.2, 8.1, 8.0 and 7.3.
- SUSE Linux Enterprise Server 9 SP3 (beta, experimental support).
- SUSE Linux Enterprise Server 9.0, 32-bit and 64-bit.
- SUSE Linux Enterprise Server 8.
- Novell Linux Desktop 9 SP2 (beta).
- Ubuntu Linux 5.10 and 5.04, 32-bit and 64-bit (experimental support).
- Windows XP Professional and XP Home Edition.
- Windows XP Professional x64 Edition.
- Windows 2000 Professional.
- Windows 2000 Server, Windows 2000 Advanced Server.
- Windows Server 2003.
- Windows Server 2003 x64 Edition.

Practically any reasonably modern x-86-compatible or x-86-64-compatible PC works as a host platform. VMware supports most Intel processors since Pentium II, and AMD processors (Athlon or better), provided they run at least 400MHz (500MHz or faster is recommended). VMware also supports multiprocessor systems. VMware Workstation 5.5 lets you create virtual machines that use Two-Way Virtual Symmetric Multiprocessing, an experimental feature.

If you need a virtual machine with more than two virtual processors, this is supported in VMware ESX Server, but if you create one and copy it to a VMware Workstation 5.5 host, it won't run unless you change its Number of CPUs setting to 2. You also can create virtual machines with the Two-Way Virtual Symmetric Multiprocessing feature on a uniprocessor host system, if it has either a dual-core CPU or hyperthreading enabled. However, according to the VMware Workstation User Manual, virtual machine performance will be subpar. And, while I'm still on the subject of CPUs, although you can't have more than two CPUs in a virtual (guest) machine, the underlying host can have as many as you like.

Besides a fast CPU (or CPUs), you need plenty of RAM. This is a simple enough equation. You need enough RAM for your host OS, for VMware itself and enough RAM for as many host OSes you intend to run concurrently. For example, my laptop has 1GB of RAM, of which SUSE 9.3 running KDE, a few Konsole shells, the usual assortment of panel applets and VMware itself use a total of about 200MB. That leaves me 800MB for virtual machines. I can comfortably (that is, without hitting swap too much) run three virtual machines that each has 256MB of RAM and so forth.

Officially, VMware requires your host system to have a minimum of 128MB of RAM (256MB is recommended), with no maximum per se, but only a total of 4GB can be used between all guest VMs.

You also need enough hard disk space both for VMware itself and for as many virtual machines as you anticipate maintaining. Both IDE and SCSI disks are supported, both on the underlying host OS and on virtual hosts.

As with RAM, the more disk space on your host system, the better. As a general rule of thumb, you need 172MB for VMware and at least 2GB per virtual machine. By using VMware shared volumes (actually Samba shares), you can share data volumes between virtual machines. This allows you to use the minimum necessary disk space for virtual machines' guest OS software and one big shared volume for application data. This is also a handy means of sharing data between virtual machines and the underlying host OS.

VMware Workstation 5.5 supports a long list of operating systems for guest/virtual machines. These include:

- Most versions of MS Windows (fully supported), including Vista (experimental support).
- Mandrake Linux, versions since 8.2.
- Red Hat Linux, versions since 7.0.
- SUSE Linux, versions since 7.3.
- Solaris x86 (experimental support), versions 9 and 10.

In practice, non-officially supported x86 operating systems often work fine as guest OSes. For example, in researching my article "Security Features in Debian GNU/Linux 3.1" (see page 36), I successfully installed Debian 3.1 on a virtual machine, despite the fact that it's not officially supported (the X Window System didn't work, but everything else I tried did, including networking).

Installing VMware Workstation

Installing VMware Workstation on a supported Linux system is a breeze. You install the RPM version either by executing a single command or by unpacking the .tgz version and manually running the installer script vmware-install.pl (which is executed automatically if you install the RPM). Then, you run the configuration script vmware-config.pl. That's it! The installer scripts and the configuration script do all the work for you.

For example, to install the RPM version of VMware, I executed the commands:

```
rpm -Uvh ./VMware-workstation-5.5.1-19175.i386.rpm
```

and:

```
vmware-config.pl
```

The configuration script asks you a number of questions, regarding things like how to set up networking. For most users, the default values are fine; otherwise, the Workstation 5 User Manual provides clear and comprehensive descriptions of the various options presented by the installer script.

Speaking of which, the user manual is, in my opinion, a model of effective technical writing—everything you need to know about VMware is included, explained in plain English and organized in a logical manner. It's accessible from within VMware's Help menu in HTML format, and it also can be downloaded as one big (490-page) PDF file from **vmware.com**.

Creating Guest Systems

Once VMware Workstation is installed and configured, you can run the vmware executable in the X Window System to start creating and using virtual machines. Figure 1 shows the New Virtual Machine Wizard.

To create a Typical virtual machine with this wizard (as opposed to a Custom virtual system), you need to make only four decisions:

1. In which OS will the guest machine run?
2. Where on your host machine's filesystem will the virtual machine's files go?

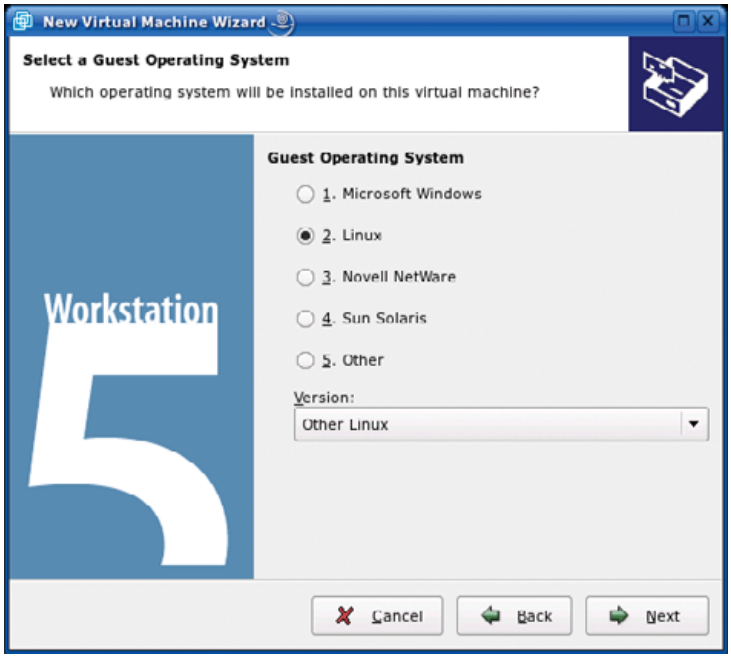


Figure 1. New Virtual Machine Wizard

3. Which flavor of VMware networking will the guest machine use?

4. What type and size of virtual disk to use?

Because this article is a review and not a how-to, I forego explaining all the different options available at this point. A few words about virtual machine hard disks and networking options, however, might help illustrate VMware's flexibility and power.

A virtual machine's hard drive is usually a virtual disk—that is, a regular file that essentially is mounted by VMware as a loopback filesystem. The beauty of this approach is that the virtual disk file doesn't need to reflect its capacity; if you install only 300MB worth of system software, applications and files on your virtual machine, its disk file will be only 300MB or so. The size you specify when setting up the virtual machine therefore will be its maximum size, not its actual size (unless you check the Allocate all disk space now option).

If you run the New Virtual Machine Wizard in Custom rather than Typical mode, you additionally can choose whether to create a virtual SCSI disk (the default) or a virtual IDE disk. You also can choose whether to use a virtual disk at all. If you prefer, and if your intended guest OS is supported in this mode, you alternatively can designate a physical disk partition on your host system as the virtual machine's root. This is handy if you have a dual-boot system on which you'd like to run both (or more) local OSes simultaneously, but support for this feature is limited and comes with some caveats. Tread lightly with this feature, and be sure to read the user manual carefully before you attempt to use it.

You can network your virtual machines using one of three methods. In bridged networking, the default, your virtual machine is given a virtual Ethernet interface connected to the same LAN as your host machine's physical network card (or wireless card—in VMware 5 you can now bridge WLAN interfaces on Linux hosts). In other words, your virtual machine appears on your local LAN as though it were sitting side by side with your host machine.

With Network Address Translation (NAT), your host system acts like a NAT firewall. Your virtual machine is given a fake IP address, and when it connects to other resources on your LAN or beyond, VMware translates the source IP address on all its packets to that of the host system's physical network interface. In other words, your virtual machine is hidden from the rest of your LAN by your host system. This is handled strictly by VMware; you don't need to configure iptables on your host OS to achieve this.

The third option is host-only networking. This is similar to the NAT mode in that your virtual machine is assigned a fake IP address on a virtual LAN separated from your physical/actual LAN by your VMware host system. The difference is that none of the virtual machines on your host-only (virtual) LAN will be able to interact with the real LAN unless you explicitly configure the underlying host OS to forward and route those packets. In other words, with host-only networking, you *will* need to configure your host OS to route or bridge your virtual machines' packets. This mode, therefore, is most useful when you *don't* want to connect your virtual and physical LANs—for example, if you're testing potentially dangerous network applications on your virtual LAN.

Other Virtual Devices

Besides virtual CPU, RAM, hard disks and network interfaces, virtual machines also can have virtual floppy disks, CD-ROM/DVD-ROM drives (data only, not movies), USB controllers, SCSI controllers, parallel ports, serial ports, sound cards and mice. Both floppy and CD/DVD drives can use either your host system's actual hardware, or disk-image or ISO files, respectively. In all cases, VMware mounts the real or virtual media for you; you don't need to run the mount command separately.

VMware's SCSI and USB support is similarly transparent. By default, if you plug in a SCSI or USB device to your host system while a virtual machine is running in the foreground (has focus), the virtual machine responds as though you plugged the device in to it. Whether this will actually *work* in a given situation depends both on VMware—the virtual USB controller supports only USB 1.1—and on the capabilities of the guest system. (Does it support USB? Have you installed the correct drivers for your device onto your virtual machine?)

Running Host Systems

Once you've created a virtual machine and installed its operating system, actually using the virtual machine is very, very similar to the real thing. Figure 2 shows the Debian 3.1 installer running on a virtual machine.

You can even, if you like, run the virtual machine in full-screen mode rather than within the VMware window. Installing the VMware Tools package on the guest system adds additional features, such as enhanced virtual-display-adapter support for your guest system and the ability to move your mouse pointer in and out of the VM window without having to click in and escape out.

A number of VMware features make the virtual machine experience *better* than using a real machine, especially for research/test scenarios. One is the ability to take snapshots of virtual machines. A snapshot captures a virtual machine's memory state, disk state and virtual machine settings at a given point, allowing you to roll back to that point later—for example, after losing control of a virus you were examining on the guest system.

Another feature is the ability to create teams of virtual machines. A team is a group of virtual machines with shared networking and startup characteristics. This lets you create, for example, a farm of database servers all connected to the same virtual LANs that all can be started simultaneously with a single-mouse click or command (VMware now has a command-line utility, vmrun, for operating virtual machines and teams).

As you'd probably expect, given that a virtual machine is nothing more than files in a directory, VMware also makes it easy to clone virtual machines. A full clone is simply a copy of the parent VM, identical to it except for having a new MAC address and UUID. A full clone, therefore, is highly portable, and it easily can be copied to other host systems.

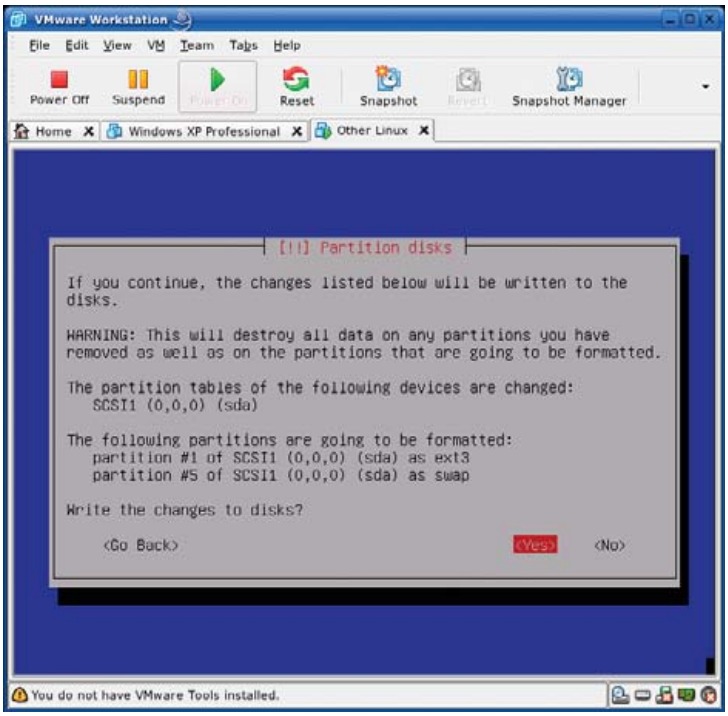


Figure 2. A Virtual Debian Machine

Another option is to create a linked clone, which actually is made from a snapshot of the parent. Changes to the parent don't affect the clone, and vice versa, but the clone must have access to the parent's files at all times.

Conclusions

So, what are the downsides to VMware? Honestly, I've been a very happy user of this product over the years. I have no laundry list of gripes or bugs to share with you, other than one hardware-specific problem with VMware 4.0 on a ThinkPad T42 running Windows XP (which I solved by switching to the Linux version). VMware Workstation 5.5 is a stable, well-documented and easy-to-use product with a rich set of features that is particularly useful to information systems professionals and researchers.

None of that comes for free, of course. The downloadable version of VMware Workstation 5.5 for Linux costs $189 US, and the boxed version is $199 US. I think you'd be hard pressed though to assemble a very good physical computer for that little money, let alone an entire LAN's worth. If in doubt, you can download the full version for a 30-day evaluation (after which you must purchase and install a license to continue using VMware).

Or, you can opt for VMware Server, which is now completely free. Formerly known as VMware GSX Server, the current version of VMware Server was still in beta at the time of this writing, but it will remain a free product even when it reaches production status. Presumably, VMware Server lacks many of VMware Workstation's developer/researcher-oriented features—the server versions of VMware are targeted more for production server applications. Compare and decide for yourself. More information about all VMware products is available at **www.vmware.com**.■

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".

RUNNING SOUND APPLICATIONS UNDER WINE

You still need Wine to get a buzz (and other audio applications) on.

DAVE PHILLIPS

Open any of the popular music trade magazines such as *Keyboard* or *Sound On Sound*, and you can't miss the plethora of colorful advertisements for sound and music software, all of it for Windows and Mac. Much of this software is of truly outstanding quality; some of it has set industry standards for features and performance, and not a bit of it is available for any platform other than Windows and Mac.

The open-source audio development community has made great strides toward providing musicians with a freely available alternative to the Win/Mac hegemony, and they deserve great praise. Nevertheless, it also must be admitted that our community is still relatively small. Potential converts to Linux often ask whether they can run their familiar programs successfully under Linux, and that criterion alone can determine whether they make the change to Linux. For all of Linux's vaunted technical superiority, it's a no-show if you need an application that simply doesn't exist for it.

This article describes how to set up and use the Wine Windows emulation environment for sound and music applications. I test a few programs, and I indicate the quality of performance you can reasonably expect from running Windows music and sound software under Wine.

Some Details

System emulators come in two basic flavors, machine architecture emulators and operating system emulators. Wine is a complete package that emulates the Windows operating system. Windows itself is not required. Wine includes its own versions of the Windows system DLLs, but you can use the native Windows versions if you prefer. Depending on the intended use, the system may require other native support software expected by your applications.

Wine's sound capabilities have developed largely in response to users who want to play their favorite games without leaving their favorite operating system. As a result, Wine has become a good candidate for running Windows audio and MIDI applications. However, before installing the emulator, you should check its documentation for the most current sound system status reports. If you intend to run a particular sound or MIDI application under emulation, your success will depend on a variety of factors, including support for the original file formats, audio sampling rates and required drivers.

The tests in this article were performed on an 800MHz machine with an M-Audio Delta 66 digital audio I/O system and an SBLive Value sound card. The software base included ALSA 1.0.4, JACK 0.99 and a rock-solid 2.4.26 Linux kernel patched for low latency. As always, your mileage may vary.

Wine

Wine is an acronym for either WINdows Emulator or Wine Is Not an Emulator. Curiously, both interpretations are correct. Wine is the wine executable, a

Linux program that runs Windows programs, and it is equally libwine, a library designed to assist Windows/Linux cross-platform development.

After 12 years at the alpha-release level, Wine is now officially a beta-stage project. Hopefully, this event signals a more consistently stable environment, but some programs still may behave erratically. The Wine documentation gives detailed instructions for submitting useful bug reports, so if you find that your favorite Windows program doesn't work well (or at all) under Wine, you can help yourself and the project by submitting a report.

Wine's support for basic sound and MIDI is good, and support for audio extensions such as Microsoft's DirectX is improving, but you won't be able to use Wine to run large, integrated multimedia applications, such as Cubase or SONAR. However, Wine can run a variety of sound and music programs, even some fairly big packages. Check the Wine Web site (see the on-line Resources) for links to lists that rate the compatibility of various Windows applications.

Getting It, Building It

The WineHQ Web site provides Wine in a variety of package formats, including the common RPM and DEB formats and full source tarball. Use your package manager of choice to install the latest version. If you decide to build Wine from the source package, simply open an xterm, enter your new wine-x.x.x directory and run `./tools/wineinstall` (as a normal user). Answer the prompts, then relax and let the Wine installer do its stuff.

After installation, run the notepad.exe file included with the distribution:

```
wine $HOME/c/windows/notepad.exe
```

If the familiar editor appears, Wine is ready for use. Now you can try to run some Windows music and sound applications.

System requirements and build procedures may change from version to version, so if you decide to build Wine yourself, be sure to read the README and follow the recommended installation instructions included with the package. The version used in this article is Wine 0.9.6, released on January 20, 2006.

Audio and MIDI Support in Wine

Linux-based musicians have two good reasons to take an interest in Wine's sound support. The first reason is applications. Some Windows sound programs have no equivalent in native Linux versions, and the possibility of running those programs under Wine is very attractive. The second reason involves libwine. That library is a key component in projects that provide support for running Windows VST/VSTi audio synthesis/processing plugins under Linux. In this article, I focus only on running applications under Wine, but readers interested in learning more about the Linux + VST connection should check out the Web page (see Resources) for details regarding the FST (FreeST) Project.

At the user level, the heart of Wine's audio support can be found in the ~/.wine/config file. Here's the relevant section of that file as it appears in my Wine configuration:

```
[WinMM]
; Uncomment the "Drivers" line matching your sound setting.

"Drivers" = "winealsa.drv"     ; for ALSA users
;"Drivers" = "wineoss.drv"     ; default for most common configurations
;"Drivers" = "winearts.drv"    ; for KDE
;"Drivers" = "winejack.drv"    ; for the JACK sound server
;"Drivers" = "winenas.drv"     ; for the NAS sound system
;"Drivers" = "wineaudioio.drv" ; for Solaris machines
;"Drivers" = ""                ; disables sound
"WaveMapper" = "msacm.drv"           ; do not change !
"MidiMapper" = "midimap.drv"         ; do not change !
```

The WaveMapper and MidiMapper are required; they emulate the native Windows MCI (Media Control Interface) drivers that provide the standard commands for controlling multimedia devices and playing and recording multimedia data files.

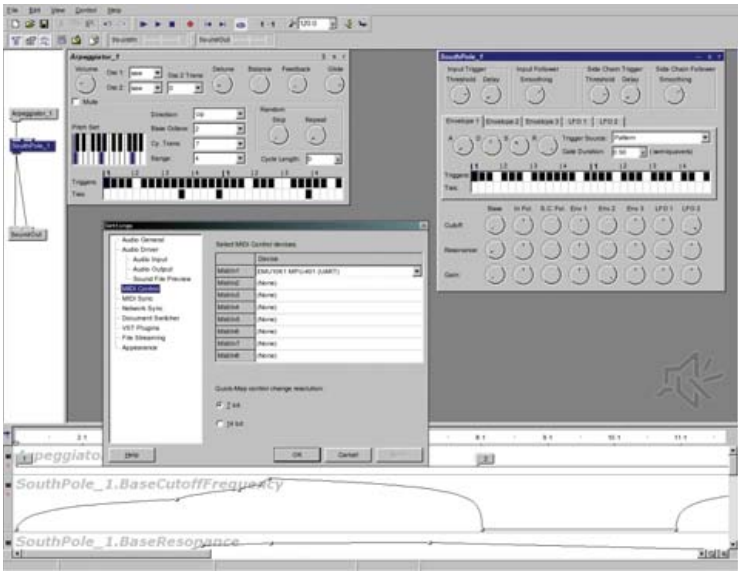


Figure 1. AudioMulch

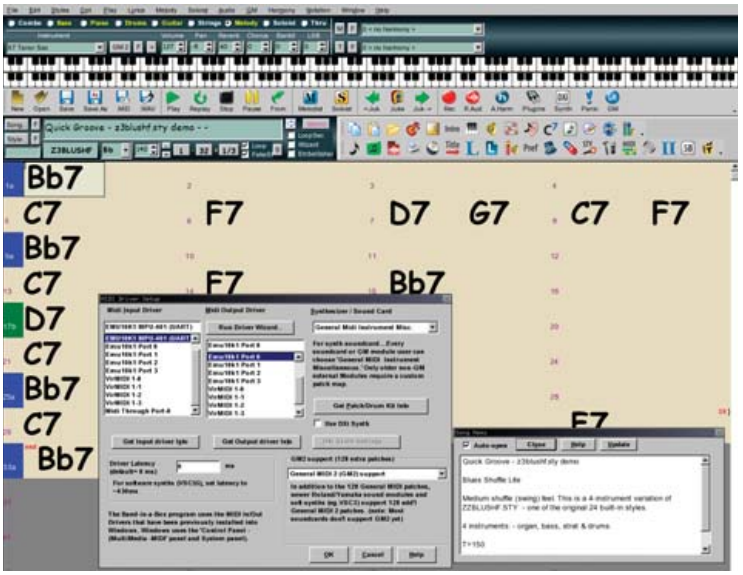


Figure 2. Band-in-a-Box

Wine provides audio interface drivers for OSS/Free (the default), ALSA, aRts, JACK and NAS (a network audio system). You can select a new driver at any time, but you will need to restart Wine. Your choice of sound driver may be determined by the application. In my experiences, some programs worked only with the OSS/Free driver, others worked only with ALSA, and some worked well with either one. I was especially excited to see a JACK driver listed, but as far as I could tell, the JACK driver is broken in this release—a reminder that Wine is still beta-stage software.

Running Sound Apps under Wine

Due to space considerations, it is not possible to describe the installation and configuration details fully for the programs I've reviewed here. I wanted to test Wine's audio performance without going to heroic measures, employing only its default settings as far as possible and doing little more than selecting an appropriate sound driver as described above. I provide a brief description of each tested program, and then I relate my experience

with running the program under Wine. Note that these tests were made mainly with the demos and examples packaged with the programs, and my conclusions are necessarily provisional and incomplete.

AudioMulch

Ross Bencina's AudioMulch is a sound synthesis and music composition environment with a unique interface and a strong emphasis on real-time performance capabilities.

AudioMulch divides itself into three main panels (Figure 1). The leftmost panel is a graphic instrument design and connections center—a canvas upon which you place and connect AudioMulch's various synthesis and processing modules. Next to this panel, we see the controls for the parameters of your selected modules. Underneath it all are the automation controls—a stack of breakpoint displays that control module parameter changes in real time.

Everything in AudioMulch is designed for real-time updates. I verified this assertion by loading an example file and randomly altering its controls and breakpoint displays at random. AudioMulch easily kept up with my changes, and Wine's audio never broke or stuttered. Very impressive!

I tested AudioMulch version 1.0rc2. It installed easily and was ready for immediate use. I loaded and ran every example included with the package, and each one performed perfectly with Wine's OSS/Free and ALSA drivers. Potential users should note that AudioMulch is shareware, not freeware, and the registration fee is $50 US. If you want to test-drive the release candidate, be aware that it will expire on the date indicated at the AudioMulch Web site.

Band-in-a-Box

Band-in-a-Box is an automatic accompaniment generator. The program creates a virtual backing band that interprets a series of user-defined chord changes according to a selected "style". A Band-in-a-Box style is a set of rules governing quantifiable aspects of a particular music performance style, such as country swing, rhumba, waltz time, blues shuffle and so forth. When the user clicks the Play control, the program processes the chord changes by the style rules, generates a real-time performance stream and plays it with your preferred MIDI synthesizer. Voilà, you have your dream rehearsal band.

Band-in-a-Box is the reigning king of the auto-accompaniment software domain. Need to play those changes more slowly? No problem, Band-in-a-Box is a MIDI-based program, so you can adjust the tempo to whatever speed is most comfortable. Want those chords played in a different meter or rhythm? Still no problem, Band-in-a-Box supplies hundreds of styles to choose from, and if you don't like what's included with the base package, you can design your own or access literally thousands of styles and arrangements created and freely distributed by the program's vast base of users and style developers. Don't like the instrumentation for a particular style? Change it on the fly, add or subtract players from the band, or mute parts at will.

I downloaded the most recent Band-in-a-Box demo from the program's Web site and installed it with `wine bbw2004demo.exe` at an xterm prompt. I entered my new ~/c/bbdemo directory and ran `wine bbwdemo` to start Band-in-a-Box. I loaded an example style from the File/BB Song dialog, pressed the Play control and watched as the program apparently played the loaded style. Alas, there was no sound. I reconfigured the default MIDI output to go to the Emu10k1 synthesizer on my SBLive Value sound card, pressed Play, and behold, I had sound. I tested other built-in styles, all perfectly happy to perform as though they were playing under Windows itself.

I discovered only one potentially serious difficulty with the demo version. I configured the MIDI input device to the hardware port on the SBLive, but Band-in-a-Box would not record what I played on my MIDI keyboard. The program's virtual keyboard display worked perfectly, but I prefer to record directly from the hardware interface, so perhaps it's time to fire up the Wine debugging tools.

Band-in-a-Box is strictly commercial software, with a list price of $88 US for the Pro Edition. The demo is, of course, free.

Band-in-a-Box has the honor of a place in the Wine AppDB Gold 10, a selection of Windows applications that has demonstrated consistently excellent performance under Wine. By the measure of my simple tests, I must concur with that rating. Band-in-a-Box is an excellent music application that runs beautifully under emulation. Consider it double-plus recommended.

Buzz

Buzz combines tracker-style pattern and sequence editors with a powerful audio synthesis/processing environment to form an all-in-one package for sound design and music composition. No other music software is quite like Buzz.

I had tried installing Buzz unsuccessfully by following the normal instructions for Windows users, but reading over the comments on the Wine AppDB, I discovered that I needed an installation package different from the one available on the official Buzz Web site. Here's what I did to install and run Buzz successfully under Wine:

- Downloaded the package found at **buzzdistro.cjb.net**.
- Ran `wine buzz_base.exe` to install the program.
- Changed directory to ~/c/Program Files/Buzz.
- Ran `wine buzz`. (Honest!)

And indeed, as shown in Figure 3, Buzz runs under Linux.

Buzz synthesis and processing modules are known as machines in Buzz-

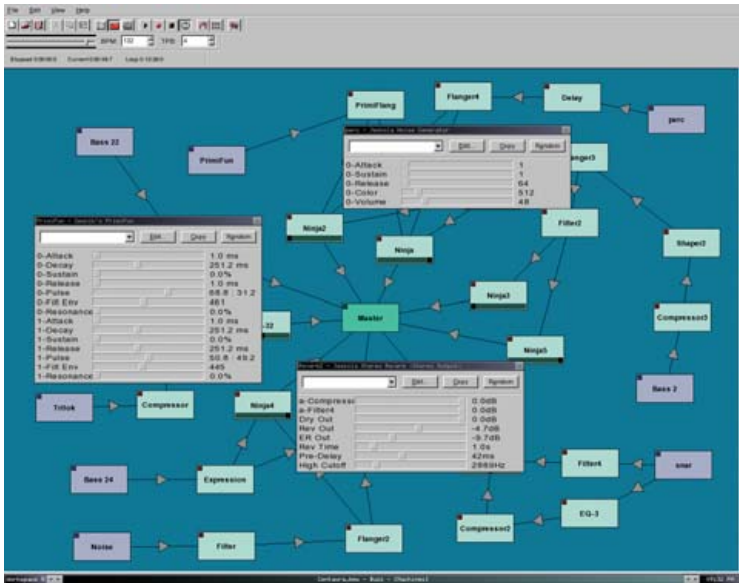


Figure 3. The Wine Buzz

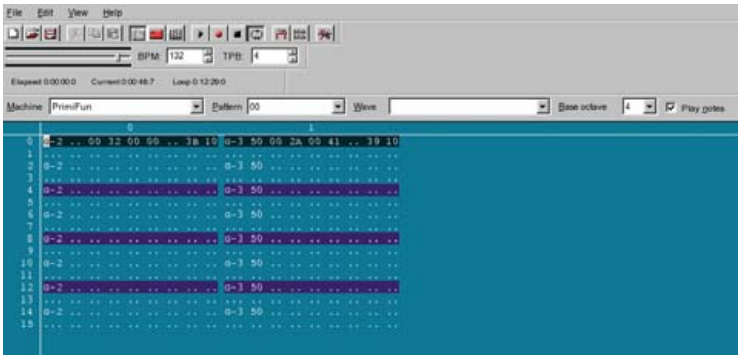


Figure 4. Buzz Tracker Interface

speak. The default package includes dozens of immediately useful machines, and hundreds more are available from the Buzz community. Like many other synthesis applications, Buzz uses a "patching" metaphor to roll your own audio processing network—that is, you link machines together with virtual patching cables to create a data-flow diagram representing your network.

Figure 3 displays some opened machines. Whenever you want to manipulate a machine's parameters, simply double-click on the machine box, and its control panel appears. You can control all parameters in real time with the mouse or with MIDI controllers.

Buzz's composition interface closely resembles a typical tracker interface (Figure 4). A scrolling display represents beats within a selected pattern length. Audio events (typically sampled sounds) are entered on the desired beat lines anywhere within the pattern. Completed patterns are then linked together to form a song sequence.

By the way, the package available from the link above is not the only Buzz-for-Linux package available. If that one doesn't work for you, try the bundle available from Flavor8 (see Resources). Peruse the hints and tips while you're there, and be sure to check out the demos made with Buzz on Linux.

Buzz is much too rich an application to treat in any depth here, so I simply recommend playing and studying some of the demo files included with the distribution. The package includes extensive documentation, and a very active community of users can be reached through the main Buzz site. Buzz is freeware, and though it's a shame that no native Linux version of Buzz exists (or ever will—the source code has been lost), in lieu of a native version, you can still enjoy a pleasant Buzz with Wine. Sorry, I just had to say it.

Some Conclusions

In the course of writing this article, I also tried to run many sound and music programs that failed in various ways. Native Instruments' very cool FM7 loaded and appeared to work (it received MIDI input from my keyboard), but no sound came from it. NI's Tracktion installed and ran, but its audio output was terribly distorted. The latest Finale demo wouldn't install at all, and the Reaktor 5 demo installed but crashed when started. Of course, all these programs run perfectly well in their native Windows environment, which is simply to say that Wine is still in development.

I also solicited the Linux Audio Users mail list regarding opinions of and experiences with the use of Wine with Windows audio applications. As might be expected, input varied. Reports included whole or partial success with applications such as Native Instruments' Battery and Kontakt, the Renoise tracker and the demo for Guitar Pro 3. I plan to put up a Web page that will list Windows audio/MIDI applications that have been tested with Wine, so if you have any notable successes or failures to report, please contact me at dlphilp@linux-sound.org.

Hopefully, Wine's JACK driver will work again in a stable version of Wine by the time this article is printed. JACK is the present and future of Linux audio, and it would be a definite Good Thing for the Wine Project. A virtual ASIO driver might be a helpful addition too.

Ideally, native Linux applications would replace their Windows counterparts, but until that happy time, Wine may prove to be a viable alternative to dual-booting or setting up secondary machines. It may lend a new lease on life to your software investment, and hopefully, it will work well enough to let you run those needed music and sound applications that still have no Linux equivalents.■

Resources for this article: www.linuxjournal.com/article/8886.

Dave Phillips is a musician, teacher and writer living in Findlay, Ohio. He has been an active member of the Linux audio community since his first contact with Linux in 1995. He is the author of *The Book of Linux Music & Sound*, as well as numerous articles in *Linux Journal*. He can be reached at dlphilp@linux-sound.org.

User-Mode Linux

Matthew E. Hoskins

User-mode Linux lets you do tricks like run a safely isolated Debian 3.1 on Fedora 4.

Computers run programs. From the earliest behemoth computers with their hard-wired programs to modern-day disposable calculators and desktop PCs, all our computers run some kind of program. John von Neumann cooked up the concept of storing a computer's program just like any other piece of data, making way for computers to become multipurpose tools no longer locked in to one hard-wired function. Soon the concept of an operating system, or a program to abstract common system-level details like device management and program execution, was born. It didn't take long for some crafty system programmers to realize that a single CPU could be made to perform multiple tasks seemingly at the same time. This gave rise to the first time-sharing and multitasking/multiuser operating systems. All modern computers still operate on this same stored program concept. In the case of a modern personal computer, after switching it on, it runs the stored program in the BIOS, or firmware, which eventually hands off to a multistage bootloader, which in turn loads the OS kernel. The kernel executes and sets up an operating environment in which system resources like CPU time, memory and devices can be used by programs executed beneath the kernel. It's all a long chain of stored programs.

The kernel is a program just like any other (albeit a rather complex one). So, what stops you from executing the kernel just like any other program? Actually, not much at all. This is what user-mode Linux (UML) is all about.

The Linux kernel normally runs with special privileges, because it needs direct access to your hardware. User-mode Linux provides a way to compile the normal Linux kernel sources so that it can be invoked as a regular binary program on top of the base Linux kernel. When you run a kernel on top of the base Linux kernel, you are really running one or more "guest" Linux systems without any special privileges. (There are some exceptions. Some software must be installed as root for user-mode Linux to work.) These guest Linux systems are complete systems that run in a (mostly) safe environment.

How to Do It

In the remainder of this article, I provide a recipe for getting a UML system up and running on your host Linux box. Then, we explore some features and have some fun. The host system I am using for this demonstration is Fedora Core 4 on an Intel P4 with 1GB of memory, but almost any system and distribution will work, provided it is running a recent 2.6 kernel and has a minimum of 256MB of memory.

A guest UML system is just like any other Linux system. It is a combination of a Linux kernel and a collection of small programs, libraries and files that make up the operating system. These are provided in two parts, the kernel and a filesystem image. A filesystem image is a virtual disk partition. This is what will be mounted and used as the root filesystem of our UML system. You have the choice to create these two parts yourself or download them off the Net, ready made from popular distributions. In the interest of instant gratification, we take the ready-made route; take a look at the UML Wiki for more information on building your own filesystems (see the on-line Resources).

Kernels and root filesystem images are available in a number of versions and distributions. Images of Red Hat, Fedora Core, Debian and a number of special-purpose distributions are available. I use Debian 3.1 for this demonstration.

COWs Are Your Friends

User-mode Linux has one very special feature called a Copy-On-Write file or COW. Copy on write is a common computer science concept that defines a mechanism for a chunk of data to remain read-only yet allows modification by writing changed data blocks to an alternate location. The filesystem image you download always remains read-only. Changes made to the filesystem in our running UML system are written to the COW file. This allows us to boot up multiple UML systems from the same read-only

root filesystem image, provided they all have separate COW files. Also, if our UML system becomes corrupted, we simply clear the COW file to start over. The COW files are what is called sparse files; even though they may appear to be big when viewing the file size, only non-null data is actually allocated space on the disk.

Collecting the Pieces

Let's start by collecting the components in a freshly created empty directory. Make sure sufficient disk space is available; after all, we need to house an entire Debian installation. Three gigabytes should be sufficient for the basic system. Download the Debian-3.1-x86-root_fs.bz2 file from **uml.nagafix.co.uk**. Then, grab the 2.6.14.3-bs3 UML kernel from **www.user-mode-linux.org/~blaisorblade/binaries**. Finally, grab the UML utilities sources from **user-mode-linux.sourceforge.net/dl-sf.html**. If any of these files are missing, you can find alternate download locations in the Resources for this article.

Below is a script of the commands for collecting all the parts and compiling the UML utilities that are available only in source form. If you are not interested in setting up networking, you can omit the uml_utilities tarball and skip the compile. All the steps below can be performed as a normal user except the installation of the UML utilities, which requires an su to root:

```
mkdir /tmp/UML-Demo
cd /tmp/UML-Demo
wget http://uml.nagafix.co.uk/Debian-3.1/
➥Debian-3.1-x86-root_fs.bz2
bunzip2 Debian-3.1-x86-root_fs.bz2
wget http://www.user-mode-linux.org/~blaisorblade/binaries/
➥2.6.14.3-bs3/uml-release-2.6.14.3-bs3.tar.bz2
tar -xvjf uml-release-2.6.14.3-bs3.tar.bz2
cp um32-2.6.14-release-mod/vmlinux-2.6.14.3-bs3  .
wget http://mirror.usermodelinux.org/uml/
➥uml_utilities_20040406.tar.bz2
tar -xvjf uml_utilities_20040406.tar.bz2
cd tools
make all
su root
make install DESTDIR=/
exit
cd ..
```

Now we have all the parts collected and are ready to rock 'n' roll. All Linux systems have a kernel command line. In most systems, this command line is invoked by the bootloader (GRUB, LILO and so on). In our case, we compose the command line ourselves to instruct the kernel to use the Debian root filesystem image and a COW file named Debian1.cow as its root (/) filesystem. Your current terminal becomes the console of the guest UML system:

```
cd /tmp/UML-Demo
./vmlinux-2.6.14.3-bs3
➥ubd0=Debian1.cow,Debian-3.1-x86-root_fs root=/dev/ubda
```

After that command is executed, we see the familiar Linux kernel boot messages ending with a Debian system waiting for someone to log in. We can log in as root (there is no password) and poke around as shown in Listing 1.

Pretty cool, eh? It's your very own Debian 3.1 sandbox to make or break as you like. You can ignore warnings about hwclock and tty0, as these are normal for most UML systems because some hardware features are not supported by UML kernels. Feel free at this point to change the root password to anything you like.

Next, let's set up networking. You need two free static IP addresses, one for each side of a tunnel that will be created by the UML utilities we compiled earlier. I use 192.168.1.100 and 192.168.1.101 here. Use anything appropriate for your local network. To get started, boot up your Debian UML again, and use the following command:

```
cd /tmp/UML-Demo
./vmlinux-2.6.14.3-bs3  ubd0=Debian1.cow,Debian-3.1-x86-root_fs
➥root=/dev/ubda eth0=tuntap,,,192.168.1.100
```

After our Debian guest system is booted, log in as root again and modify the network configuration as follows.

Edit the /etc/network/interfaces file to contain only the following lines:

```
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet static
 address 192.168.1.101
 netmask 255.255.255.0
 gateway 192.168.1.1
```

Enter a hostname of your choice in the /etc/hostname file and, finally, copy your resolver settings from /etc/resolv.conf on the host system to the

Listing 1.

Debian 3.1 UML Guest Boot Demonstration

```
Debian GNU/Linux testing/unstable (none) tty0

(none) login: root  <ENTER>
Linux (none) 2.6.14.3-bs3 #7 Fri Dec 16 17:47:00 CET 2005 i686 GNU/Linux

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.

(none):~# ps -ef <ENTER>
UID         PID   PPID  C STIME TTY          TIME CMD
root          1      0  0 22:56 ?        00:00:00 init [2]
root          2      1  0 22:56 ?        00:00:00 [ksoftirqd/0]
root          3      1  0 22:56 ?        00:00:00 [watchdog/0]
root          4      1  0 22:56 ?        00:00:00 [events/0]
root          5      1  0 22:56 ?        00:00:00 [khelper]
root          6      1  0 22:56 ?        00:00:00 [kthread]
root          7      6  0 22:56 ?        00:00:00 [kblockd/0]
root          8      6  0 22:56 ?        00:00:00 [pdflush]
root          9      6  0 22:56 ?        00:00:00 [pdflush]
root         11      6  0 22:56 ?        00:00:00 [aio/0]
root         10      1  0 22:56 ?        00:00:00 [kswapd0]
root         12      1  0 22:56 ?        00:00:00 [kjournald]
root        299      1  0 22:56 ?        00:00:00 /sbin/syslogd
root        305      1  0 22:56 ?        00:00:00 /sbin/klogd
root        343      1  0 22:56 ?        00:00:00 /usr/sbin/exim4 -bd -q30m
root        348      1  0 22:56 ?        00:00:00 /usr/sbin/inetd
daemon      361      1  0 22:56 ?        00:00:00 /usr/sbin/atd
root        364      1  0 22:56 ?        00:00:00 /usr/sbin/cron
root        379      1  0 22:56 tty0     00:00:00 /bin/login —
root        380    379  0 22:56 tty0     00:00:00 -bash
root        384    380  0 22:57 tty0     00:00:00 ps -ef

(none):~# df -h <ENTER>
Filesystem            Size  Used Avail Use% Mounted on
/dev/ubda             1008M  264M  694M  28% /
tmpfs                  768M     0  768M   0% /tmp
tmpfs                   14M     0   14M   0% /dev/shm

(none):~# halt <ENTER>

Broadcast message from root (tty0) (Sun Jan 15 22:57:17 2006):

The system is going down for system halt NOW!
```

guest Debian system. Halt the guest system and reboot.

After the guest system is booted, you will be able to ping it from anywhere on your network. I would suggest doing a couple things to your newly networked Debian system. First, install OpenSSH, and then update all installed packages to current versions. To do so, execute the following commands and answer the simple questions when asked:

```
apt-get install openssh-server
apt-get upgrade
```

The possibilities at this point are wide open. Any network service or application can be run under this guest Debian install. You can use UML to test applications across many kernel versions and Linux distributions all on one box. You can place the filesystem image and COW file on a USB thumbdrive, giving you a stable development environment across all the computers you use. User-mode Linux makes it easy and painless to test system changes that otherwise might make a system unbootable.

Listing 2.

Destruction can be fun if you are just testing.

```
Debian GNU/Linux testing/unstable (none) tty0

(none) login: root
Linux (none) 2.6.14.3-bs3 #7 Fri Dec 16 17:47:00 CET 2005 i686 GNU/Linux

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.

(none):~#  rm -rf /
rm: cannot remove `//proc/meminfo': Operation not permitted
rm: cannot remove `//proc/uptime': Operation not permitted
(... Many warnings about read-only filesystems omitted ...)
rm: cannot remove `//proc/loadavg': Operation not permitted
rm: `//proc/self' changed dev/ino: Operation not permitted
(none):~# df -k
-bash: df: command not found
(none):~# ps -ef
-bash: ps: command not found
(none):~# halt
-bash: /sbin/halt: No such file or directory
(none):~#
```

Fun—As in Destruction!

Okay, you know you've always wanted to do it. Now, here is your chance. Bring up a new standalone guest Debian UML system, and do an `rm -rf /`. If you are like me, your fingers start to curl under as you even consider typing that command. To begin, boot up the new Debian guest using the following command (notice we are using a different COW file, because we do not want to disturb the nice networked setup we created previously):

```
cd /tmp/UML-Demo
./vmlinux-2.6.14.3-bs3  ubd0=DangerDanger.cow,Debian-3.1-x86-root_fs
➥root=/dev/ubda
```

After our doomed friend boots up, let 'em have it. Make sure you double- (perhaps even triple-) check that you are still typing in the guest Debian system (Listing 2)!

It's hosed up pretty good at this point. In fact, you can't even run `halt`, because the halt program itself is gone. From another command window, kill the system with:

```
killall -9 vmlinux-2.6.14.3-bs3
```

Then, see what happens when you try to boot it up again using the same command (Listing 3).

That's gotta hurt. So, as a lesson, *do not* do that on a real system. But because this is a UML guest with a COW file, you simply can delete the DangerDanger.cow file, and this guest system will boot up back to its initial state.

Listing 3.

A Very Ill Debian UML Guest

```
(... boot messages omitted ...)
EXT3-fs: INFO: recovery required on readonly filesystem.
EXT3-fs: write access will be enabled during recovery.
kjournald starting.  Commit interval 5 seconds
EXT3-fs: ubda: orphan cleanup on readonly fs
EXT3-fs: ubda: 66 orphan inodes deleted
EXT3-fs: recovery complete.
EXT3-fs: mounted filesystem with ordered data mode.
VFS: Mounted root (ext3 filesystem) readonly.
Warning: unable to open an initial console.
Kernel panic - not syncing: No init found. Try passing init= option to kernel.
EIP: 0073:[<a01c6691>] CPU: 0 Not tainted ESP: 007b:b7f3afac EFLAGS: 00000282
Not tainted
EAX: 00000000 EBX: 000012eb ECX: 00000013 EDX: 000012eb
ESI: 000012e8 EDI: 00000000 EBP: b7f3afb8 DS: 007b ES: 007b
a10afb80:  [<a0032d2a>] show_regs+0x21a/0x230
a10afbb0:  [<a0016c8c>] panic_exit+0x2c/0x50
a10afbc0:  [<a004a275>] notifier_call_chain+0x25/0x40
a10afbe0:  [<a0037501>] panic+0x71/0x100
a10afc00:  [<a000e2c0>] init+0x100/0x170
a10afc20:  [<a002bf59>] run_kernel_thread+0x39/0x50
a10afce0:  [<a001c3d4>] new_thread_handler+0xc4/0x120
a10afd20:  [<b7f3b420>] 0xb7f3b420
```

More on COW Files

The utility uml_moo included in the UML utilities will read a filesystem image and an associated COW file and create a new merged filesystem image. This allows you to merge changes stored in the COW file into a new master filesystem image. This makes it easy to clone working guest filesystem images when you have them set up the way you want.

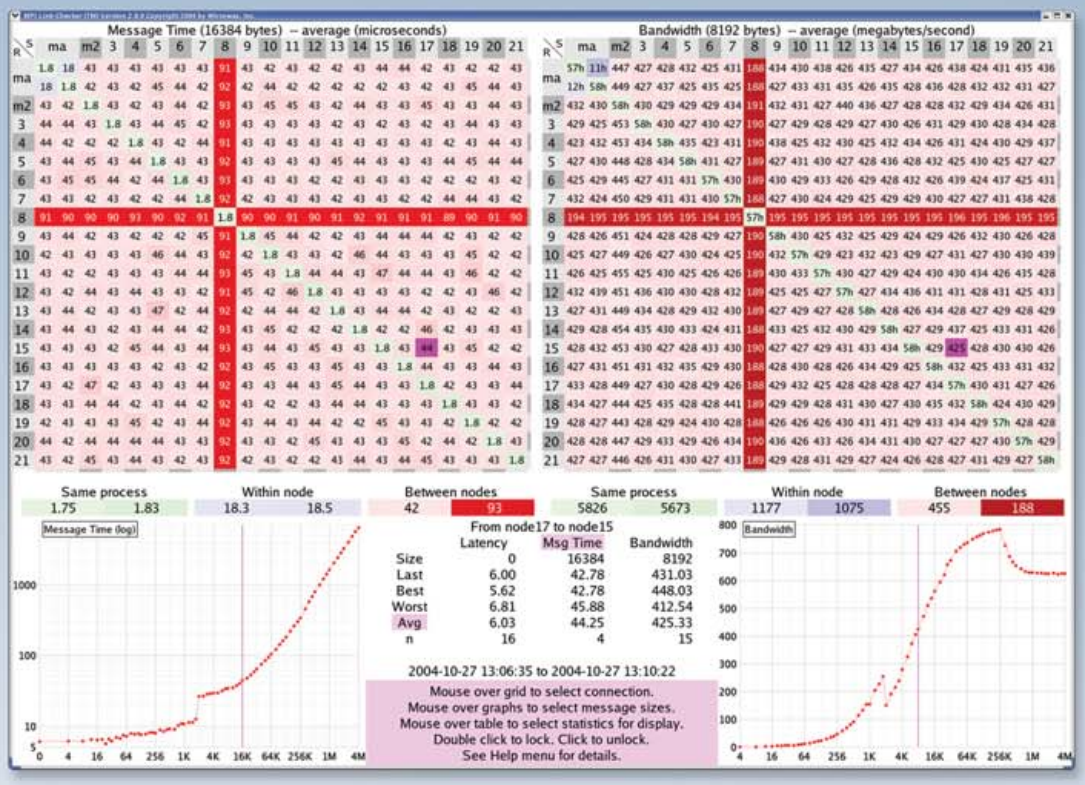
Conclusion

User-mode Linux is fun to play with, but it also has some real-world uses. You can use it to test unknown or untrusted applications while limiting possible damage to the running host system. You can create virtual networks of UMLs by starting up multiple guests at once. This allows you to create a test-lab-in-a-box environment with very little time and effort, so you can try all those "Stupid Linux Tricks" you were afraid to try on a real system!■

Resources for this article: www.linuxjournal.com/article/8883.

Matthew E. Hoskins is a Senior UNIX System Administrator for The New Jersey Institute of Technology where he maintains many of the corporate administrative systems. He enjoys trying to get wildly different systems and software working together, usually with a thin layer of Perl (locally known as "MattGlue"). When not hacking systems, he often can be found hacking in the kitchen. Matt is a member of the Society of Professional Journalists. He is eager to hear your feedback and can be reached at matt@njit.edu.

X Marks the Slow Node!



MPI Link-Checker™ to the Rescue!

A single slow node or intermittent link can cut the speed of MPI applications by half. Whether you use GigE, Myrinet, Quadrics, InfiniBand or InfiniPath HTX, there is only one choice for monitoring and debugging your cluster of SMP nodes: Microway's MPI Link-Checker™.

This unique diagnostic tool uses an end-to-end stress test to find problems with cables, processors, BIOS's, PCI buses, NIC's, switches, and even MPI itself! It provides instant details on how latency and bandwidth vary with packet size. It also provides ancillary data on inter-process and intra-CPU latency, and includes FastCheck!, which runs in CLI mode and checks up to 100 nodes per second.
A complimentary one year license for MPI Link-Checker™ is installed on every Opteron based Microway cluster purchased in 2006.

Wondering what's wrong with your cluster's performance, or need help designing your next one? Microway designs award-winning single and dual core AMD Opteron based clusters. Dual core enables users to increase computing capacity without increasing power requirements, thereby providing the best performance per watt. Configurations include 1U, 2U, and our 4U QuadPuter™ RuggedRack™–available with four or eight dual core Opterons, offering the perfect balance between performance and density.

Microway has been an innovator in HPC since 1982. We have thousands of happy customers in HPC, Energy, Enterprise and Life Science markets. Isn't it time you became one?

Call us first at 508-746-7341 for quotes and benchmarking services. Find technical information, testimonials, and newsletter at www.microway.com.

Excellence in Cluster Technology
64-bit Turnkey Solution
WINNER
ClusterWorld

▲ Microway® Quad Opteron™ Cluster with 36 Opteron 880s, redundant power, 45 hard drives and Myrinet™ in our CoolRak™ cabinet.

Microway®
23 Years of Expertise Built In

AMD 64 Opteron™

QEMU

a Multihost Multitarget Emulator

QEMU comes to the rescue for those times when VMware is overkill.

Daniel Bartholomew

A few months ago, I suddenly found myself with an I-need-to-run-just-one-Windows-application problem. When I had started at my current job, I was determined finally to be Windows-free at work, just like I have been at home for several years. To that end, after I had unpacked my shiny new work computer, I erased Windows, installed my current favorite Linux distribution, and set up Ximian Evolution to connect to the Microsoft Exchange server. I thought that I had finally arrived at Linux nirvana.

It was not to be.

Microsoft's Exchange mail server has this feature where a team, or group of people, can access a shared mailbox. My manager thought it would be a good idea to set up one for our team and have clients send e-mail to that address instead of to each of us individually. With the shared inbox in place, I found myself needing to check it several times a day.

Evolution can connect to shared mailboxes, but not in the way I have to connect to mine. My department, being Linux-friendly and security-conscious, is not on the corporate network, so those who are on Windows-based systems in my department need to configure their Outlook e-mail client to connect to the Exchange server over HTTP. Evolution seems to support connecting to shared mailboxes only when you are on the same network as the Exchange server, not via the HTTP method.

There was no way around it. I had to run Outlook. And not just any version. I had to run Outlook 2003, which is the version that can use the HTTP-connection method. The problem with Outlook 2003 that older versions of Outlook do not have is that it is not compatible with Wine or CrossOver Office from CodeWeavers, which ruled out what I considered to be the obvious first-choice solution for running it on Linux.

My options were therefore:

1. Go back to using Windows.

2. Find another way to run Outlook 2003.

I did not want to go back to using Windows, not for only one application, so I started looking around for answers to the other option. My requirements were simple: it had to be able to run Outlook 2003, it had to be cheap, it had to be usable and it had to be reliable—no crashing. VMware is an obvious choice for this sort of thing, but as I was footing the bill myself, VMware was not an option. After a little bit of searching, I found an excellent VMware alternative: QEMU.

What Is QEMU?

According to its home page: "QEMU is a FAST! processor emulator using dynamic translation to achieve good emulation speed."

QEMU is a multihost, multitarget emulator. QEMU will run on x86, x86-64 and PowerPC systems, and it can emulate x86, x86-64, ARM, SPARC, PowerPC and MIPS architectures. For most of these, it can be run in two ways: full-system emulation and user-mode emulation. For details on which modes are supported for which architectures, check out the link in the online Resources.

User-mode emulation allows you to run Linux binaries compiled for other architectures on your machine. This is great for application development and testing, but I was more interested in full-system emulation.

Full-system emulation emulates a complete computer system from the BIOS on up to things like video and sound cards. For x86 system emulation, QEMU simulates a machine with the following peripherals:

- i440FX host PCI bridge and PIIX3 PCI to ISA bridge.
- Cirrus CLGD 5446 PCI VGA card or dummy VGA card with Bochs VESA extensions (hardware level, including all nonstandard modes).
- PS/2 mouse and keyboard.
- Two PCI IDE interfaces with hard disk and CD-ROM support.
- Floppy disk.
- NE2000 PCI network adapters.
- Serial ports.
- SoundBlaster 16 card.
- PC BIOS from the Bochs Project.
- Plex86/Bochs LGPL VGA BIOS.

From the above list, you probably can tell that QEMU is not in contention as the Ultimate Linux Box. However, each of the emulated devices is well supported by Linux and Windows, which leads to easy Virtual Machine (VM) installs and no driver hunting, which is a "Very Good Thing".

Being on an x86-based machine myself and not needing to run an OS that requires or even uses an x86-64, ARM, SPARC, PowerPC or MIPS processor, I can't vouch for QEMU's performance in that regard. I have tested some disk images of DebianPPC, Gentoo for SPARC and MenuetOS_64, which is written in x86-64 assembly language. They all booted and ran without trouble, but I was not able to compare their performance to real hardware. These, and many other QEMU-ready disk images, are available from the FreeOS Zoo (see Resources).

My purpose in using QEMU was to run an x86-based OS—Microsoft's Windows XP—inside my x86-based OS of choice, which is currently Ubuntu Linux 5.10. The good thing about this particular setup is that QEMU can employ a virtualization layer, called the KQEMU accelerator, on top of its standard emulation engine that speeds things up to what the QEMU Web site claims are "near native speeds". Near native or not, I can say this, with the KQEMU accelerator installed, things are definitely faster.

The accelerator hands off as much processing as it can to the real processor and emulates only the necessary bits. This makes perfect sense. Why emulate x86 on x86? If there are good reasons to do so, I cannot think of any.

Installing QEMU

To install QEMU, download the source package from the main QEMU Web site (see Resources) and the binary kqemu package. There is also a binary QEMU package available. If you download and install the binary, you will not be able to use KQEMU, because it needs to be compiled into QEMU to work. KQEMU, unlike QEMU, is available only as a binary package. It is not open source. See the KQEMU sidebar for more information. At the time of this writing, QEMU is at version 0.8.0, and KQEMU is at version 0.7.2. Because they are under active development, there may be updated versions available by the time you read this.

Once I had downloaded the two packages, I first untarred QEMU with:

```
tar -zxvf qemu-0.8.0.tar.gz
```

Next, I changed directories into the qemu-0.8.0 directory I had just created and did:

```
tar -zxvf ../path/to/kqemu-0.7.2.tar.gz
```

This created a kqemu directory inside of my qemu-0.8.0 directory.

When I compile applications from source, a `./configure`, `make`, `make install` at this point is usually all that I need to do to get a piece of software installed. QEMU needed a bit more hand holding.

In order to have QEMU compile successfully on my machine, I had to make a few changes to the configure script. The changes themselves were quite simple. First, QEMU does not get along with 4.x versions of gcc, so I had to change the cc= and host_cc= lines to use gcc-3.4 specifically. Then, I had to change `kqemu="no"` to `kqemu="yes"`. Finally, it was necessary to

About KQEMU

Unlike QEMU, which is open source, KQEMU is a closed-source, proprietary product. The reason for this is money. QEMU developer Fabrice Bellard has stated that he would be willing to open-source KQEMU on one condition: if a corporate sponsor picked up the tab for its continued development. Until then, although you can download it without cost, KQEMU will remain a proprietary component in an otherwise open-source product.

There is a project to create an open-source drop-in replacement to KQEMU called qvm86 (see Resources). I have not used it, but I have read statements that say it works as well as or better than KQEMU.

Figure 1. QEMU Running the KNOPPIX Live CD

enter the path to my kernel source tree in `kernel_path=""`. One note: QEMU uses SDL for output, so although I did not need to install anything extra for my particular setup, others may have to install some SDL libraries before the configure script will be happy.

Once I was able to run `./configure` without it complaining, I ran `make` and then `make install` to install QEMU to my /usr/local/ directory. To install the KQEMU accelerator kernel module, I typed the following into an open terminal:

```
modprobe kqemu
```

Using an Existing Image

Once I had installed QEMU, I wanted to see it in action. The easiest way to try it out was to boot a live CD ISO image like KNOPPIX, Ubuntu, SimplyMepis, DSL, Puppy or one of the scores of others. To boot QEMU off a bootable CD image, I simply entered the following at the command line:

```
qemu -boot d -cdrom path/to/distro.iso
```

The -boot d parameter tells QEMU to boot from the CD drive, and -cdrom path/to/distro.iso tells QEMU where the CD-ROM "drive" is, which in this example, is simply an ISO image. I also could have pointed QEMU at my actual CD-ROM drive—/dev/cdrom—and when I installed Windows, that is what I did.

Creating an Image and Installing an OS

Before I could install an operating system, I first needed to prepare a virtual hard disk in which to install. QEMU understands various disk image formats, including VMware's vmdk, which would have come in handy if I had some of them lying around. As it turned out, the default, "raw" format worked well. A raw format disk image acts like an unformatted hard drive, which was perfect for my needs.

I used the following command to create an image named winxp.img, 5GB in size, which I figured was big enough to install Windows XP and Outlook and give me plenty of e-mail storage:

```
qemu-img create winxp.img 5120M
```

Looking back, a better size would have been 4GB, because that would have made it easier to create DVD backups.

Now that I had a virtual hard drive, I put my Windows XP Pro installation CD into my CD-ROM drive and launched QEMU with the appropriate arguments:

```
qemu -boot d -hda path/to/winxp.img -cdrom /dev/cdrom -m 256 -localtime
```

The -m 256 option set the memory allocated to the VM to 256MB—the default is 128MB, which is a bit small for Windows XP Pro. The -localtime option set the virtual BIOS clock to the local time on the host machine—the default is to set the BIOS clock to Universal Coordinated Time.

I found that installing Windows onto a virtual machine was very similar to installing it on a "real" computer. The installer comes up and has you choose where to install Windows. It asks you if you want to format your hard drive, prompts you to enter in your license key and so on. Once the base install was done, I shut down the VM, replaced my Windows XP Pro installation CD with my Microsoft Outlook installation CD and launched QEMU like so:

```
qemu -boot c -hda path/to/winxp.img -cdrom /dev/cdrom -m 256 -localtime
```

Figure 2. Booting Windows for the First Time

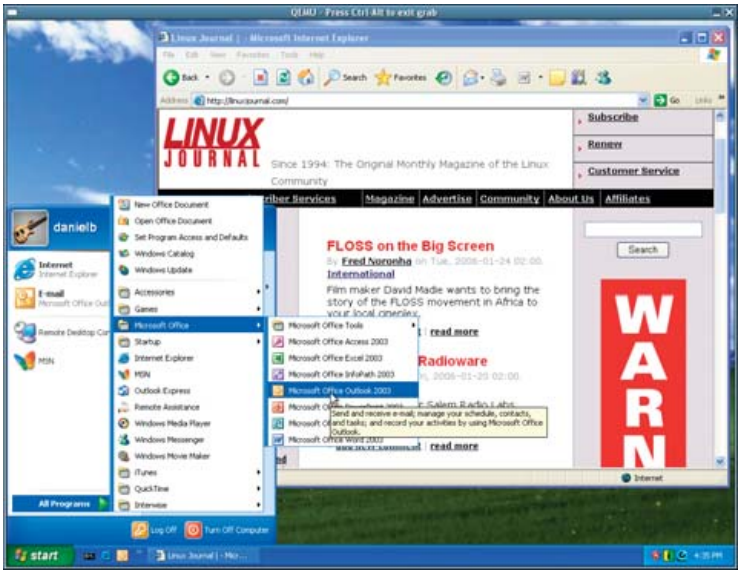


Figure 3. Windows XP Pro Running in QEMU

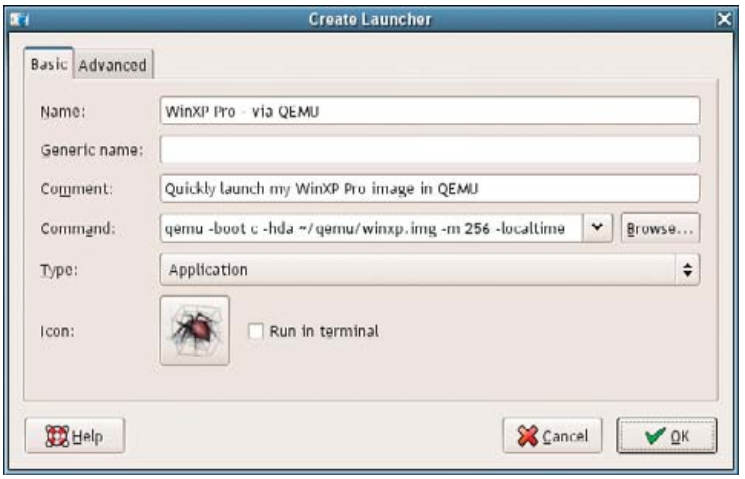


Figure 4. A Simple GNOME Launcher for QEMU

The only difference was to boot from winxp.img instead of from /dev/cdrom. The Outlook installation went like a typical Outlook installation—no real surprises there. When the Outlook installation was completed, I had a functioning Windows machine to call on whenever I needed it.

Now that I had my base operating system and needed application installed, I shut down the machine and created a GNOME launcher, so I could fire up my virtual Windows machine without typing it into my terminal every time. As you can see from Figure 4, I basically removed the CD-ROM info from the command, because I don't need a CD-ROM to be present during normal operation. Refer to the documentation for your distribution on how to create a custom application launcher.

Squeezing More Performance out of QEMU

There are a few ways to squeeze extra performance out of QEMU:

1. More RAM: the first thing I did was add more RAM. I did not want QEMU to have any reason to access my swap partition. Swap partitions are very useful, but too slow for resource-intensive tasks such as emulation. With more RAM, you get not only better performance from your VM, you also can run multiple VMs at the same time (see Figure 5 for an example of this).

Figure 5. QEMU Running Three Virtual Machines Simultaneously

2. KQEMU: the second thing I did to get better performance was to compile in the KQEMU accelerator module.

3. Set up a RAM disk: even with extra memory, there are times when QEMU needs to cache things to disk. To speed up this process, I set up a RAM disk. A RAM disk is a virtual disk drive created from free RAM. To create it, I entered the following into my /etc/fstab and then rebooted my machine:

```
tmpfs        /dev/shm        tmpfs    defaults    0    0
```

Getting Files in to and out of QEMU

A dilemma I ran into after I started using QEMU was how to get files out of my QEMU VM for backup purposes. The first method I tried was to install an SSH secure copy (scp) client for Windows and then use it to transfer files to myself. This works, but it was not as simple as I wanted the process to be.

The second method I tried worked much better. When starting QEMU, there is an option to specify a shared directory. QEMU makes it available to the VM through Samba, so you need to have Samba installed for this to work. The option is -smb dir—where dir is the directory on my host machine for which I want my Windows XP VM to have access. I then added the following line to C:\WINNT\SYSTEM32\DRIVERS\ETC\LMHOSTS on my Windows XP VM:

```
10.0.2.4 smbserver
```

Accessing my shared folder from within my VM was then as easy as navigating to \\smbserver\qemu.

Conclusion

QEMU may lack the graphical configuration and VM setup tools of commercial programs like VMware, but I have found it to be an excellent solution to the I-need-to-run-just-one-Windows-application problem. Judging from the comments I've read on the QEMU forum and on the #qemu channel on the Freenode IRC network, QEMU is well suited to solving many other problems. Give it a try, I think you'll like it.■

Resources for this article: www.linuxjournal.com/article/8884.

Daniel Bartholomew has been using computers since the early 1980s when his parents brought home an Apple IIe (with an 80-column card!). After stints on Mac and Windows machines, he discovered Linux (Slackware) in 1996 and has been using various distributions ever since. He lives with his wife and children in North Carolina.

Xen is a hypervisor virtual machine that runs multiple open-source operating systems.

IRFAN HABIB

In the last half century, microcomputers have become increasingly powerful. Server systems have grown so powerful, that many enterprise servers typically are underutilized. Modern computers are sufficiently powerful to use virtualization to present the illusion of running many virtual systems on a single machine. Each virtual system runs a separate operating system instance simultaneously. So, you can run multiple instances of Linux at the same time on the same machine, or you can run combinations of operating systems, such as Linux, FreeBSD, Windows and so on. This has led to a resurgence of interest in Virtual Machine (VM) technology, which has been around for decades on bigger iron.

The Systems Research Group at the University of Cambridge Computer Laboratory originally developed Xen (open-source virtualization software) as part of the XenoServers Project, funded by the UK-EPSRC.

XenoServers aims to provide a "public infrastructure for global distributed computing". Xen plays a key part in that project, allowing users to partition a single machine efficiently to enable multiple independent clients to run their operating systems and applications in an environment. See **www.cl.cam.ac.uk/xeno** for more information on the XenoServers Project.

Xen is an x86 virtual machine monitor that allows multiple commodity operating systems, such as Linux and MS Windows, to share conventional hardware in a safe and resource-managed fashion. It is designed with minimal performance overhead. As a result, the virtualized instances of operating systems have close to native performance. The Xen folks achieve this by providing a virtual machine abstraction to which operating systems, such as Linux and MS Windows, can be ported with minimal effort. Xen has, according to a number of benchmarks, considerably out-performed competing commercial and freely available solutions.

XenSource

Xen is such an effective means of lowering total cost of ownership through virtualization that the original Xen development team launched a consulting business based on the project. See XenSource (**www.xensource.com**), which is considered "home to the world-wide Xen open source community".

Applications of Xen

One of the major uses of Xen so far has been for consolidation of servers. An organization can shift server software hosted on multiple physically separate servers and locate them onto a single server, by using virtual machines for each individual server. For example, it is now possible for a company to host Sendmail on a FreeBSD installation while hosting the Apache Web server on Red Hat Enterprise Edition, both on the same physical server.

This enables enterprises to reduce their total cost of ownership by using a few servers to do tasks that used to require many servers. Server consolidation also makes it easier to manage systems.

Xen can enable the development of distributed Web services. This gives users the perception that services are hosted on separate systems, but they, in fact, are hosted on the same physical system. This leads to huge savings in IT budgets in deploying service-oriented applications and provides a platform for hosting other network-centric applications.

Xen has been a boon in operating system research. Through Xen, it is now possible to implement new kernel-level algorithms and test them in a virtual environment without affecting the host OS. In Linux kernel development, employing user-mode Linux is popular; however, Xen has out-performed user-mode Linux in a number of benchmarks.

Xen's virtualization capabilities have enabled organizations to keep their servers available 24/7. Organizations can launch a temporary virtual server to keep services available while patching and upgrading an OS on the virtual server they normally use to provide those services.

Xen also enables organizations to run legacy applications on new hardware, protecting their past investments.

Comparison to Other Approaches

Now that we have a taste of the potential applications and advantages Xen offers, let's briefly look how it compares to other approaches and explore some salient features of its internal workings.

Hosting different operating systems on a single server is nothing new. Many desktop PCs nowadays are dual-boot systems, where at least two different operating systems are installed in a single machine, each running a set of software specific to each.

When users require both operating systems to run at the same time, there are several options. They can get two computer systems and dedicate each system to each service. They can use an emulator such as Wine or Win4Lin to run services from MS Windows on Linux, or use CoLinux to run unmodified Linux services on MS Windows.

However, these approaches have certain drawbacks. Getting two servers to host two services is inherently expensive and would lead to underutilization of resources. Wine, Win4Lin or other emulators often have performance, scalability and compatibility problems.

So, the best solution in many cases is to run virtual machine software on a single machine and host both operating systems at once on the same machine.

Proprietary virtualization systems exist, such as VMware Workstation [see page 56 for Mick Bauer's review of VMware Workstation 5.5]. Software, such as VMware, implements what is called full virtualization. VMware virtualizes every aspect of the computer. VMware, therefore, introduces a good deal of overhead. The concurrent operating systems often run more slowly than usual. As hardware becomes cheaper and more powerful and software becomes more optimized, this lag in performance may not be noticeable in the future, but currently it poses a problem.

VMware does have enterprise-level commercial products, such as ESX Server, which have better performance than the VMware Workstation product, and such a product may be able to run virtualized operating systems close to their native performance. However, benchmarks of this product are not available, and VMware Workstation consistently has under-performed Xen in various benchmark tests.

VMware's approach does have one large advantage over Xen's approach. VMware is capable of virtualizing proprietary operating systems. As I discuss later in this article, you have to port an operating system's kernel to Xen for it to work with Xen. You cannot run an operating system on Xen otherwise.

Hypervisor

Xen is a virtual machine hypervisor. That is, it doesn't run on any OS, it makes an OS run on it! Xen runs at the highest priority level the x86 architecture allows (called Ring 0). It makes the OS get the second-highest priority in x86 architecture (called Ring 1).

Xen provides certain libraries to which the OS kernel has to be ported in order to work with Xen. Porting an OS to run on Xen is similar to porting the OS to a new hardware platform; however, the process is simplified because the paravirtual machine architecture is very similar to the underlying native hardware. Although the kernel has to be ported, Xen does not require any modification to user applications, which can run unaltered on a Xen system.

So far, only open-source operating systems have been ported to Xen. Unless Microsoft releases a Xen-enabled Windows version, we might not get the benefit of a completely virtualized MS Windows. So far, Linux ports are available, and FreeBSD, NetBSD and Solaris 10 ports are underway.

The developers of Xen had to overcome

some major challenges to partition successfully a modern machine's resources amongst multiple guest operating systems. First, virtual machines had to be isolated from one another—that is, problems in one machine must not affect the working of other virtual machines. Second, it was necessary to support a variety of different operating systems to accommodate the heterogeneity of popular applications, such as enterprises commonly using a mix of Linux and MS Windows installations to support their working. Third, the performance overhead introduced by virtualization should be small. Xen's approach addresses each one of these challenges successfully. See "Xen and the Art of Virtualization" at **www.cl.cam.ac.uk/Research/SRG/netos/papers/2003-xensosp.pdf**, which discusses Xen's approach in detail.

We'll go through some salient features of Xen's approach.

As stated earlier, Xen is a hypervisor that uses paravirtualization, when an operating system is ported to Xen. Xen has access to some internal OS kernel information in order to manage the system. This porting also gives the guest OS kernel access to real as well as virtual information, which has specific advantages for time-critical tasks. Paravirtualization permits very high-performance virtualization, even on architectures like x86 that don't inherently support virtualization.

Paravirtualization enables Xen to multiplex physical resources at the granularity of an entire operating system and is able to provide performance isolation between each VM. This also allows a range of guest operating systems to coexist, without having any effect on each other. Xen's paravirtualization approach allows users to run applications in a resource-controlled fashion. Furthermore, it provides an extremely high level of flexibility, because users can create dynamically the precise execution environment their software requires. Unfortunate configuration interactions between various services and applications are avoided.

Try It Yourself

As mentioned previously, Xen is primarily developed for the x86 architecture; however, it does not support all x86-based processors—only those that are P6 or newer, including Pentium Pro to Pentium 4 and Intel Celeron and Intel Pentium Xeon processors. Apart from Intel, AMD processors from Athlon to AthlonXP and FX processors are supported, as well as the AMD Duron.

Interested readers may want to try out Xen for themselves, without installing the entire system. The Xen Project provides a live CD demonstration of Xen, which comes with both Debian and CentOS. The live CD version can be a powerful tool for demonstrating the features of Xen. It is possible to boot in to any of the provided distributions and start new instances of either distribution, as many times as the system memory allows.

It is also possible, in the live CD version, to monitor the resource usage of all virtual machines in real time and start applications in each virtual machine.

Installing Xen

For power users who want to get down to installing Xen, the following is a brief guide. Installing Xen is a three-way process. You install Xen and its user-level tools, then configure your bootloader and, finally, define the VM configuration files for each guest OS.

Installing from binary tarballs or an RPM package is the easiest way to install Xen. For binary tarballs, simply do this:

```
bash# tar zxvf xen-3.0-install.tgz
bash# cd xen-3.0-install
bash# sh ./install.sh
```

For an RPM package, do this:

```
bash# rpm -iv xen-3.0-i686.rpm
```

(or whatever the name of the RPM is).

Installing Xen from source is more complicated, as it involves patching and recompiling the Linux kernel. Installing from source is not covered in this article, but is described thoroughly in the Xen User Manual (**www.cl.cam.ac.uk/Research/SRG/netos/xen/documentation.html**).

Configuring the Bootloader

After Xen has been installed, we need to configure the bootloader. For GRUB users, edit the menu.lst file, and add this entry:

```
title Xen
kernel /boot/xen-3.0.gz dom0_mem=32768 module
➥/boot/vmlinuz-2.6-xen0 root=/dev/hda7 ro console=tty0
```

vmlinuz-2.6-xen0 is the kernel image that would have been installed by the tarball binaries or RPM; if you install from source, replace the name of the image here.

Also be sure to replace the name of the root filesystem to suit your system (in this example, it is root=/dev/hda7).

For LILO users, do the following:

```
image="/boot/xen-3.0.gz"
label="Xen"
root="/dev/hda7"
read-only
append="dom0_mem=32768"
```

After Xen has been installed and configured, you are now ready to boot in to Xen and start your first virtual machine.

After rebooting and starting your Xen installation, which resembles a

normal Linux startup, log in to Domain0. That is the most-privileged domain in a Xen system.

From here, users can create virtual machines that will run guest operating systems, and start and stop virtual machines.

To create a new virtual machine, you need to define a configuration file for it. Xen comes with two default configuration files in the /etc/xen directory named xmexample1/ amd xmexample2/. The configuration files contain many parameters, but fortunately, many of them are optional. You need only a few configured parameters to get your virtual machine running. Some important parameters include:

- Kernel: which kernel to boot.
- Root: the root filesystem.
- Disk: on which disk partition the system is installed.
- Memory: define how much memory the virtual machine should use.

A sample configuration file may look like this:

```
kernel = '/boot/vmlinuz-2.6.12.6-xen'

disk = [ 'phy:hda1,hda1,w' ]

root = '/dev/hda1 ro'

memory = 128
```

After declaring a configuration file for the virtual machine, you can boot up the machine by typing the following:

```
bash# xm create -c /root/myOSconf vmid=1
```

where myOsconf is the name of the configuration file.

After this, a new window will pop up, and you will see a normal Linux startup until you reach the login screen, and from there you can enjoy your new guest OS.

Conclusion

Xen is mature, open-source virtualization software that creates many new opportunities for organizations in reducing their total cost of ownership and providing more dependable and high-availability applications. Commodity x86-based systems provide all of this, with a minimum cost of porting an operating system to Xen.

The developers of Xen have tested Xen against other popular virtualization solutions, such as VMware Workstation and user-mode Linux. In all tests conducted, Xen out-performed the other approaches—in standard benchmark tests, such as Spec Int200, Spec Web99, dbench and many more. The results were published in a research paper, available at **www.cl.cam.ac.uk/Research/SRG/netos/papers/2003-xensosp.pdf**.■

Irfan Habib has been an open-source enthusiast for five years. He has great interest in distributed computing technologies, in which he does full-time research, and he loves to explore new solutions to common problems in computing. Comments can be sent to him at irfan.habib@gmail.com.

Embedded Java with GCJ

You don't always need a Java Virtual Machine to run Java in an embedded system.

GENE SALLY

This article discusses how to use GCJ, part of the GCC compiler suite, in an embedded Linux project. Like all tools, GCJ has benefits, namely the ability to code in a high-level language like Java, and its share of drawbacks as well. The notion of getting GCJ running on a embedded target may be daunting at first, but you'll see that doing so requires less effort than you may think.

After reading the article, you should be inspired to try this out on a target to see whether GCJ can fit into your next project. The Java language has all sorts of nifty features, like automatic garbage collection, an extensive, robust run-time library and expressive object-oriented constructs that help you quickly develop reliable code.

Why Use GCJ in the First Place?

The native code compiler for Java does what is says: compiles your Java source down to the machine code for the target. This means you won't have to get a JVM (Java Virtual Machine) on your target. When you run the program's code, it won't start a VM, it will simply load and run like any other program. This doesn't necessarily mean your code will run faster. Sometimes you get better performance numbers for byte code running on a hot-spot VM versus GCJ-compiled code.

One advantage of using GCJ is that you save space by not needing the JVM. You may save money in royalties as well. Furthermore, using GCJ lets you deliver a solution using all open-source software, and that's usually a good thing.

Pitfalls

The first thing embedded engineers reach for when creating a root filesystem for a target is trusty uClibc, a compact implementation of the glibc library. For those new to using Linux on an embedded target, the standard C library can be a bit on the large side when working with targets that may have only 8MB (for example) for a root filesystem. To conserve space on an embedded system's root filesystem, engineers will switch from the standard C library to something smaller, like uClibc. GCJ requires unicode support, which is not supported by uClibc, so glibc is a requirement.

The standard library for GCJ weighs in at 16MB, so even if you could conserve space by switching to a smaller standard C library, it wouldn't make that much difference overall. The standard GCJ library can be trimmed by removing support for executing Java byte code, but the loss of that feature would reduce the overall value of GCJ.

The Host and Target Configuration

Because this article is about using GCJ in an embedded environment, it shows you how to build a cross compiler and a simple root filesystem for the target system. For those new to embedded development, a cross compiler builds code that runs on a processor different from the machine where the compilation occurred. The machine that runs the compiler is called the host and the one where the code runs is called the target.

In this article, the target system is a PPC 745/755-based system running at 350MHz. This particular board comes wrapped in a translucent case with a monitor and hard drive and is otherwise known as an iMac. Okay, this is hardly a prime example of an embedded system, but it does present some of the same challenges you'll encounter with a true embedded system. The things you learn here should apply well to embedded systems using other processors.

The host system is a run-of-the-mill IBM ThinkPad notebook running a Pentium III processor. Yellow Dog Linux is already running on the host system, but with a little sleight of hand, we'll make it use the root filesystem created in the article for testing.

Getting GCJ Ready

First, we need a cross compiler that runs on our Pentium machine that creates code for a PowerPC 750-based processor. Building a cross compiler for a target system can be a very tedious process; a working compiler is more than GCC, it also contains some extra tools (affectionately named binutils) and the standard libraries for the language.

To get a cross compiler up and running quickly, try using the crosstool package, compliments of Dan Kegel. Crosstool does all of the hard work necessary to get a cross compiler built: it fetches the source and patches, applies the patches, configures the packages and kicks off the build. After obtaining and unpacking crosstool, here are the steps for building your GCJ cross compiler:

```
$ export TARBALLS_DIR=~/crosstool-download
$ export RESULT_TOP=/opt/crosstool
$ export GCC_LANGUAGES="c,c++,Java"
$ eval `cat powerpc-750.dat gcc-4.0.1-glibc-2.2.2.dat' sh.all --notest
```

While waiting for the compilation to finish, let's take a look at what we just did to start our crosstool build. TARBALLS_DIR is the location where crosstool downloads its files. Crosstool fetches all of the files it needs for a build by default. RESULT_TOP is the installation directory of the cross compiler. Lastly, GCC_LANGUAGES controls which language front ends will be enabled for the compiler. GCC supports many different language front ends and each front end adds a considerable amount of time to the compilation process, so only the necessary ones were selected for this toolchain build.

The last line, for those lacking their bash script-foo license, dumps

Getting and Unpacking Crosstool

Crosstool is the creation of Dan Kegel. You can find out everything you want to know about crosstool by visiting **kegel.com/crosstool**. The page has a great quick start guide as well as complete documentation. This article used version 0.38 available at **kegel.com/crosstool/crosstool-0.38.tar.gz**.

On the crosstool home page, check out the buildlogs link (**kegel.com/crosstool/crosstool-0.38/buildlogs**) to see what combinations of glibc/gcc successfully build for your target architecture.

the two dat files on the command line and executes the all.sh script with the parameter --notest. To make building a cross compiler easy, crosstool includes configuration files with the correct environment variables set for the target processor and the gcc/glibc combination. In this case, crosstool builds a gcc 4.0.1 with glibc 2.2.2 targeting a PPC 750 processor. Crosstool includes scripts for all major processor architectures and glib/gcc combinations.

When the build finishes, the cross compiler will be installed at $RESULT_TOP/gcc-4.0.1-glibc-2.2.2/powerpc-750-linux-gnu/bin. Add this to your path to make invoking the cross compiler easier.

Configuring the Root Filesystem

The first thing to compile with your newly minted cross compiler is the root filesystem. The root filesystem, in this case, is compliments of BusyBox. For the uninitiated, BusyBox is a single binary that encapsulates mini versions of the most popular UNIX utilities in a surpassingly small executable. Built for people that count bytes, BusyBox has hundreds of knobs to turn to create a filesystem with the utilities you need within your desired space constraints. For the purpose of this article, we change the BusyBox configuration so that it cross compiles, leaving size optimization as an exercise for the reader.

BusyBox is a mainstay of the embedded Linux world and is maintained by Erik Anderson. One way to get BusyBox is to download it at **www.busybox.net/downloads/busybox-1.01.tar.bz2**.

To create a BusyBox root filesystem, you need to invoke `make menuconfig` in the directory where BusyBox was untarred. The menuconfig program works just like the 2.4/2.6 menuconfig kernel configuration interface. Here's what you'll need to do to build the root filesystem.

First, select the build options. Check the Do you want to build BusyBox with a Cross Compiler? box. Fill in the prefix of the cross compiler in the input control that appears when you click this option, in this case, powerpc-750-linux-gnu-. The BusyBox build scripts concatenate the necessary tool name during compilation (gcc, ld and so on). Make sure that the compiler is on your $PATH.

Next, run make and install:

```
make
make install
```

BusyBox puts the newly minted root filesystem at ./_install. You'll notice that BusyBox compiles in much less time than GCC.

Populating the Root Filesystem with Libraries

Almost there! The root filesystem BusyBox creates does not contain any libraries. GCJ programs require some libraries and so does BusyBox, shown in Table 1.

These libraries match those used by the cross compiler. In this case, the files are stored in the $RESULT_TOP/gcc-4.0.1-glibc-2.2.2/powerpc-750-linux-gnu/powerpc-750-linux-gnu/lib (not a typo!) directory. The easiest way to get them into the root filesystem is simply to copy them:

```
for f in ld.so.1 lib libdl.so.2 libgcc_s.so.1libgcj.so.6
 ➥libm.so.6 libpthread.so.0 ; do

cp
$RESULT_TOP/gcc-4.0.1-glibc-2.2.2/powerpc-750-linux-gnu/
   powerpc-750-linux-gnu/lib/$f
<busybox install directory>/lib

$RESULT_TOP/gcc-4.0.1-glibc-2.2.2/powerpc-750-linux-gnu/bin/power
pc-750-linux-gnu-strip <busybox install directory>/lib/$f

done
```

Table 1. Libraries Required by GCJ and BusyBox

Library File	Function
ld.so.1	Dynamically linked file loader. Invoked when the program is run, loads required libraries and performs dynamic linking.
libdl.so.2	Helper functions for manipulating dynamic libraries.
libgcc_s.so.1	Defines interfaces for handling exceptions.
libgcj.so.6	The GCJ run-time library. Contains implementations of the standard Java library.
libm.so.6	Library of math functions.
libpthread.so.0	POSIX threads library.

You also need to create a folder in the root filesystem, /proc, to use as a mountpoint for the proc filesystem. Keen eyes will notice that I'm not preserving the symlinks used to accommodate different versions of the libraries—that's a shortcut typical in embedded systems where library configuration won't change over the life of the device, unlike a desktop system. Running strip greatly reduces the amount of disk space required by the files, almost by 50%.

At this point, the root filesystem can be copied to the target system into the /tmp/bbox directory. To tell the system to use this as the root filesystem, start a terminal as root and execute the chroot command:

```
chroot /tmp/bbox /bin/ash
```

This command remaps the / mountpoint into /tmp/busybox and runs /bin/ash to get a terminal. Did it work? Congratulations! You've created a complete root filesystem for an embedded system from scratch. Pat yourself on the back.

GCJ also needs the proc filesystem mounted. After the chroot, you need to remount the proc filesystem into the current root filesystem by doing the following:

```
mkdir /proc
mount -t proc none /proc
```

Although this root filesystem resides on a standard drive, the root filesystem deployed on a production embedded system wouldn't be much different. The only changes necessary would be creating inittab, so the board will run the right scripts at the start and add a /dev filesystem with the right device files for the target board.

GCJ Development

After building the cross compiler and root filesystem, building your GCJ application will be a bit anticlimactic. We'll start with the traditional hello world:

```
Class hello {
 Static public void main(String argc[]) {
   System.out.println("hello from GCJ");
 }
}
```

Following Java convention, this class resides in the hello.class file. To compile the file, enter:

```
powerpc-750-linux-gnu-gcj hello.class --main=hello -o hello-java
```

What's going on with --main=hello? Any class could define a method with a suitable entry point. The --main=hello option tells the linker to use the main method in the hello class when linking. Leaving off this option results in a link error, "undefined reference to main", which, to the uninitiated, is confusing, because your class contains a main.

Download this file to the target and run it from the chrooted shell. You'll see:

```
# ./java-test
Hello from GCJ
```

At this point, development carries on much like any other Java project, with the exception of invoking the GCJ cross compiler instead of the native javac compiler.

Conserving Space

In this example, the root filesystem weighs in at more than 20MB. Because many embedded systems use Flash memory, which is considerably more expensive on a per-megabyte basis than disk-based storage systems, a minimally sized root filesystem is frequently an important business requirement. One easy way to reduce the size of your root filesystem is to link your application statically. Although this may seem counterintuitive at first, as you'll have an extra copy of libc code in your application, recall that libgcj.so contains the entire Java standard library. Most applications use a fraction of the standard Java library, so using static linking is a great way to winnow out the unused code in the library. Just be sure to strip the resulting binary; otherwise, you'll be shocked at the size due to the amount of debugging information in libgcj.so.

Wrapping Up

From the article, you've seen that creating software for an embedded system using GCJ is something that can be reasonably accomplished using tools already present in the Open Source community. Although there are a few minor nits, configuring the root filesystem doesn't require a heroic effort; you just need to get a few different libraries from what you otherwise would need. For applications requiring a smaller root filesystem, we've seen how you can use static linking of your application to reduce the root filesystem greatly. In short, GCJ is a practical solution for using Java on a resource-constrained embedded system—worthy of consideration for your next project.■

Gene Sally has been working with Linux in one form or another for the last ten years. These days, Gene focuses his attention on helping engineers use Linux on embedded targets. Feel free to contact Gene at gene.sally@gmail.com.

Why is LPI the World's #1 IT certification?

Stable.

All Linux Professional Institute certification programs are created with extensive community and enterprise input, rigorous psychometric scrutiny, and professional implementation procedures

LPI strives to remain an independent and unbiased certification body. As a result, LPI is supported by a broad range of enterprise customers, governments, testing centers, book publishers, courseware suppliers , and educational institutions from around the globe.

Innovative.

LPI's programs follow Linux Standard Base (LSB) specifications, so people who earn our certifications are qualified to work on all major distributions. Through our strong grass-roots base and corporate support both inside and outside the world of open-source, LPI goes beyond "vendor-neutral" to truly address community and enterprise needs.

We are the first IT certification to obtain professional accreditation and advocate the adoption of Open Source Standards through our work with such organizations as the Free Standards Group. We also committed to the development of innovative open source software tools which will improve and streamline our test development processes.

Growing.

We have delivered over 100,000 exams and 30,000 certifications around the world. Our exams are available in several languages, at more than 7,000 locations, in more than 100 countries. You can take LPI exams where and when you want.

LPI serves Linux professionals and the IT industry. We have attracted broad support from prominent members of the Linux community and enterprise corporations, as demonstrated by our Strategic Advisory Council and sponsors. In addition, we are establishing a Technical Advisory Council to ensure we address the future needs of the industry. Our vendor-independent stance ensures we focus solely on the skills and knowledge necessary for an IT professional rather than on the promotion of a specific software vendor or distribution.

Linux Professional Institute

For more information,
Please contact us at
Info@lpi.org or visit us at
www.lpi.org.

Is Linux Voice over IP Ready?

A Voice over IP primer with special attention to using it on Linux.

MACHTELT GARRELS

What is Voice over IP really? What do you need for Voice over IP? What do you mean, I can't call my girlfriend? What's all that buzz about open and proprietary protocols? Can I start my own telecom service? This article addresses these questions and compares the most popular Linux applications for calling and conferencing.

What Is Voice over IP about?

Internet or digital telephony, or Voice over IP, often abbreviated as VoIP, allows parties to exchange voice data streams over the network. The big difference is that with VoIP the data flows over a general-purpose network, the Internet, whereas conventional telephony uses a dedicated network of voice transmission lines.

Under special circumstances, a VoIP network can be connected with the conventional telephone network. However, at the time of this writing, that is certainly not the standard. In other words, it is likely you will not be able to call people who are using a conventional telephone. Although currently various applications are available, both free and proprietary, telephony over the Internet has some major drawbacks. Most noticeably, the system is unreliable, it can be slow, or there can be a lot of noise on the connection. Therefore, it cannot be used to replace conventional telephony. Think about emergency calls, for instance. Although some providers take some precautions, there is no guarantee you will be able to reach the party you want to call. This is worsened, because in VoIP, there is no agreement yet on a standard for assigning numbers, like the E.164 standard we have for assigning and identifying traditional land lines and mobile phone numbers.

Even if there is some form of integration between VoIP and conventional telephony networks, this is still different for mobile networks. The major problem is that wireless network coverage is not as well developed as cellular network coverage. Additionally, there is the issue of costs when accessing the Internet from your mobile phone. For me, it would amount to 0.50 EUR (+/– $.60 US) per 100K of traffic. It is possible that integration of VoIP in the third-generation telephony network will ease these troubles.

You also should be aware that there is no encryption in VoIP. So, it is fairly easy for anyone to eavesdrop on conversations.

The bottom line is although VoIP is useful, it is not a replacement for land-line telephony (yet).

Let's look at what you'll need to get VoIP up and running.

On the Server Side

First of all, you need a provider offering the service. Some popular providers offer the service for free, and some require a subscription fee. Among the free ones are the following: SIPphone, Skype, SIP Broker and Google.

Most free services, however, do not allow you to connect with the conventional telephone network. This so-called full phone service is usually not free. Among the most popular full phone service providers are the following: Vonage, Lingo, AOL TotalTalk and SIPphone.

These lists are certainly not exhaustive, as new local and global providers join the pool on a near-daily basis. Also, many SMEs are currently setting up a VoIP network for internal use within companies. If you want to set up your own VoIP network, you might want to look into Asterisk server software or sipX, which are open-source PBX implementations.

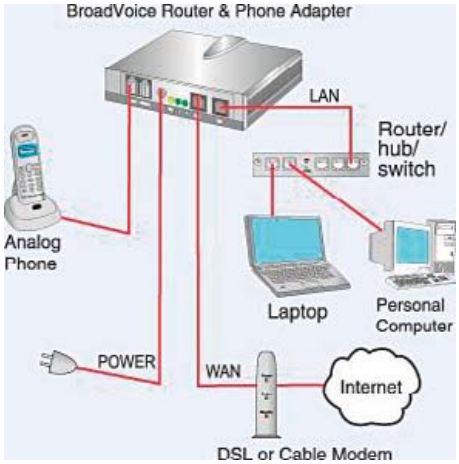


Figure 1. A Typical VoIP Solution (image courtesy of BroadVoice)

Alternatively, if you want to use only the soft phone, meaning the audio system of your computer (audio boxes and microphone or headset) and accompanying software, check out Ekiga, formerly GnomeMeeting, as announced January 8, 2006, in the GnomeMeeting blog. Although Ekiga supports a range of hardware, it is usually set up to support (video) conferencing features implemented on the software level. Like sipX and Asterisk, it is open-source software.

Note: PBX stands for Private Branch eXchange, the system that centralizes all of a business' telephone sets.

On the Client Side

Depending on your network architecture, some applications might work better than others, due to the protocols they use. Most standards-based solutions use either the H.323 or Session Initiation Protocol (SIP). Apart from these two standards, there are a lot of proprietary protocols, such as Skype (from the company with the same name) and SCCP from Cisco. The main difference between them is that SIP stores the client IP address in its packages, resulting in difficulties when you are behind a firewall.

Microsoft NetMeeting and GnomeMeeting use H.323; Microsoft's Messenger and Apple's iChat and SIPphone uses SIP. Server software usually implements several different protocols.

Apart from your network architecture, available bandwidth also might be a limiting factor, as some applications are optimized for low bandwidth, and others expect to be on a broadband connection. This depends on the codecs the VoIP systems use for handling sound.

As far as client hardware is concerned, use a headset. Although your PC, especially if it is a laptop, might have a microphone and speakers built in, you will be far more comfortable using a headset, as it will suppress echo and noise from your environment. If you have the choice, opt for a USB headset. A USB headset is a separate audio device to your system. It functions independently from existing audio hardware, so it avoids any conflicts that might occur between VoIP and normal sound processing.

If the applications you use provide the features, you can redirect audio streams as desired. For instance, you can make the ring tone for alerting you that you have a call come through the normal speakers. When you pick up the call, the voice of the calling party is redirected to your headset.

Your Sound System

Prior to experimenting with VoIP applications, you probably will have to use a bit of trial and error to find settings that are comfortable for you. Make sure that you can record and play a sample of your own voice before you start, as the VoIP programs also will use the recording function of your hardware. Activate it in the volume control application that comes with your distribution.

Linux generally has two types of sound architecture: the older Open Sound System or OSS, which works with every UNIX-like system, and the newer Advanced Linux Sound Architecture or ALSA, which has better support for Linux, as the name indicates. One application may support OSS and another, ALSA. When you have a choice, we advise you to select the use ALSA option in VoIP programs. Select ALSA or OSS settings for sound and recording levels accordingly in your distribution's volume control panel.

We tested four applications, based on popularity. We tested all of them on Fedora Linux.

Ekiga, aka GnomeMeeting

Installation: use the package manager from Fedora. Alternatively, download Debian, Mandrake or Red Hat packages. Ekiga requires the pwlib, OpenH323 and libavc1394 packages.

Getting started—registration: the application shows up in the menus as Video Conferencing. We experienced GConf errors the first time we used it. The solution to that problem is described in the GnomeMeeting FAQ. Once we solved that problem, we could get started with the First Time Configuration Druid.

You can register in the general GnomeMeeting users directory (a telephone book on a central server) or skip this step. My audio devices were recognized automatically, and it was easy to select the headset. You don't need to know the device names of your hardware. For beginning users, it is a great relief not having to worry about /dev/dsp1 and those sorts of names. As shown in Figure 2, all applicable devices can be neatly selected from a list.

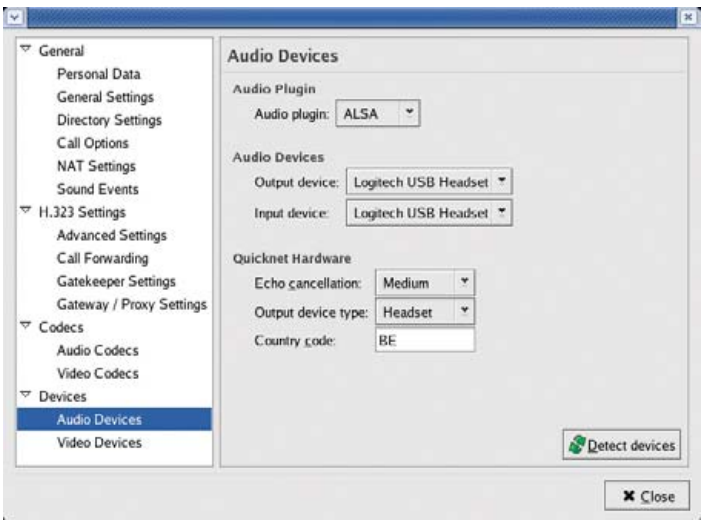


Figure 2. GnomeMeeting—Configuration

Presumably, your machine needs to be configured as an LDAP client (Lightweight Directory Access Protocol, or Active Directory on MS Windows) in order to be able to contact the central GnomeMeeting directory. Lacking that, you need to know the hostname or host IP address and user names of the people you want to call. If you don't use LDAP, you will receive error messages when you try to call someone, even if you can

Advertiser Index

Figure 3. GnomeMeeting—Interface

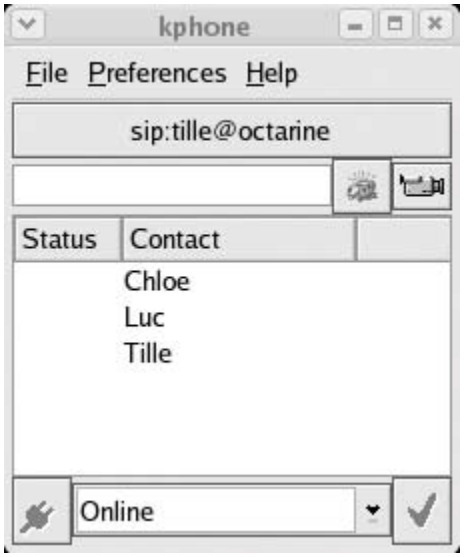


Figure 4. KPhone—Minimalistic Interface

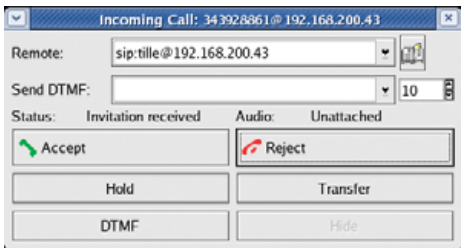


Figure 5. KPhone—Accepting a Call

make a successful call.

Impressions: at first there was quite some noise on the connection, even when calling another host in the same subnet, but we could minimize the noise by adjusting the audio volume. There is a mute button for suspending and resuming audio transmission. Luckily, the system with URLs to contact people is well documented in the help files. The application itself doesn't make it easy to use.

KPhone

Download using your favorite system tool, such as Synaptics on Ubuntu.

Installation: the package manager does the installation for you. You also can download RPM packages and install them using your distribution's tools. After the installation is finished, the KPhone selection turns up in the application menus.

Getting started: your own address is displayed in the little KPhone window, which makes it easy to exchange with other users. It also serves as an example for connecting with other users.

The phone book in this application is easy to use. In the most basic case, simply let others call you, and received and missed caller IDs will show up in the phone book automatically.

Impressions: KPhone has a very sober interface, which makes it easy to use and configure the program.

At one time, I obviously must have configured the wrong audio device while trying to configure KPhone to use my USB headset instead of the built-in speakers and microphone on my laptop. There is no list from which to choose audio devices; this was rather frustrating. KPhone also segfaulted on me a couple of times, even after it had worked fine earlier. I could not get my USB headset to work. Admittedly, I did not use the latest version. Newer versions, which need to be compiled from source on many systems, at the time of this writing, are reported to work better and have much improved sound quality. KPhone has matured a lot in the newest releases and probably will become even more popular than it is already as binary packages are made available.

Skype

Download from the Skype site (see the on-line Resources); packages are available for SUSE, Fedora, Mandriva and Debian.

Installation: I opened the downloaded file directly in the system installer. It shows up in the Internet menu in GNOME after the installation is finished.

Getting started—registration: register from inside the client. Choose a user name and password. Enter your e-mail address if you want to be reminded of your password later on. In the contact list, select or search for the person whom you want to call. You can ask permission to be notified when that person is on-line.

Figure 6. Skype—Connection Established

Impressions: Skype is easy to search by name, city and country. When you start it up, you will see a list of missed calls and contacts that are currently on-line, which is quite handy.

Skype can be configured to use PC speakers for incoming call alerts and a headset for actual communication.

In the call list, contacts can be displayed by name, or you can sort by incoming, outgoing and failed calls.

On the downside, the application does not seem to be very clean. After a while, I could not log in anymore, and it turned out that five instances of Skype had been running simultaneously on my computer, even though I used the buttons and menus to quit Skype. Also, it

Figure 7. Skype—Contact List

Figure 8. Skype—Call List

does not seem to be very stable on Linux. I had what appeared to be remote sound problems, but the problem was local and could be solved by stopping and starting Skype.

X-Lite

Download from the CounterPath Web site (see Resources).

Installation: extract the archive to a folder in your home directory; the default name is xten-lite. In this folder, you will find the executable file, xtensoftphone.

Getting started—registration: right-click

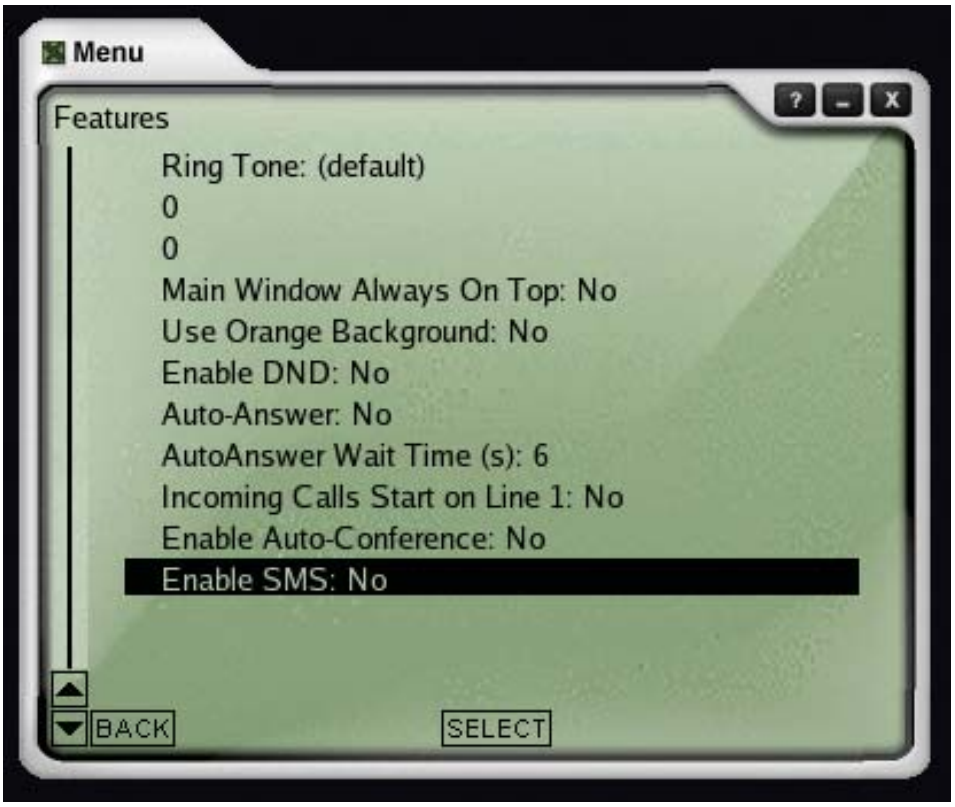


Figure 9. X-Lite Controls

Figure 10. X-Lite during a Call

on the soft-phone image that appears at startup. This starts the Audio Tuning Wizard, which allows you to select audio devices. Select /dev/dsp1 when using a headset. Adjust the speaker volume and voice recording volume according to your needs.

You can register at **support.xten.net** to join the CounterPath community, or your system administrator might have set up a private service. I used the X-Lite interface for testing with the Asterisk service at work. In both cases, you need to provide a login name and password, which you get either from the CounterPath registration on the Web site or from your administrator.

Impressions: X-Lite is the only application on this list that actually tries to look like a cell phone. You can select a codec according to your needs. For instance, choose the GSM codec for low-bandwidth usage or when you are in a conference call. Select the g711a or g711u codec when

Table 1. Comparison

	GnomeMeeting	KPhone	Skype	X-Lite
Maintainer(s)	Damien Sandras	Wirlab Research Center	Skype Technologies S.A.	CounterPath Solutions, Inc.
Licence	GPL	GPL	proprietary freeware	proprietary freeware
Platforms	GNOME, KDE	Linux (Qt)	MS Windows, Linux, BSD, Mac (Qt)	MS Windows, Pocket PC, Mac, Linux
Protocol	H.323, SIP	SIP	proprietary	SIP
Behind firewall/proxy	yes	possible	possible	possible
PC-to-phone calls	possible	no	non-free	non-free
Video conferencing	yes	limited	no	in Pro version (non-free)
Rating from 1–10	9	8	7	8

you are in a one-on-one call, and bandwidth is not really an issue.

Comparison

In Table 1, several aspects of the four applications are compared. For readability, features are restricted to those affecting telephone capabilities. All applications have many more features. I list only those that are different among applications.

Conclusion

Overall, the experience was quite positive. Although the open-source programs KPhone and GnomeMeeting are somewhat more difficult to use, because you need to know about URLs and such, it is easier to get documentation on exactly what you need and to get that documentation directly from the makers of the program, instead of having to be satisfied with a general help page and some vague complaints or tips from users.

For Skype and X-Lite, you need to connect to a server. The nice thing about GnomeMeeting and KPhone is that you can use them directly from client to client, even if you do not register on a server, be it one that you set up on your own network or an external one. Maybe I am a bit paranoid, but I don't trust Skype. The company says that its service will remain free, but it says nothing about its software. As it uses a proprietary, poorly documented protocol, I worry about vendor lock-in. For the time being, I'll stick with GnomeMeeting. Why? It worked from my first attempt, it is stable, it does everything it promises to do, and it's Belgian, like me.■

Resources for this article:
www.linuxjournal.com/article/8887.

Machtelt Garrels has published two books for Linux beginners, which are also available as guides in the Linux Documentation Project (LDP) collection. Currently, she works as a freelancer in the UNIX Competence Center of the Belgian Railways and has a part-time assignment as Translation and Vendor Coordinator for the Linux Professional Institute (LPI). Her spare time goes to—don't laugh—knitting and stitching for friends and family, because you can't sit in front of a computer all your life.

Manipulating the Networking Environment Using RTNETLINK

How to use RTNETLINK to develop applications that control networking.

ASANGA UDUGAMA

NETLINK is a facility in the Linux operating system for user-space applications to communicate with the kernel. NETLINK is an extension of the standard socket implementation. Using NETLINK, an application can send/receive information to/from different kernel features, such as networking, to check current status and control them.

In this article, I describe how a programmer can use the networking environment manipulation capability of NETLINK known as RTNETLINK. I discuss some areas of use of RTNETLINK, the relevant socket operations, the functionality, how RTNETLINK messages are formed and finally, provide a set of sample code that uses RTNETLINK. RTNETLINK for the IP version 4 environment is referred to as NETLINK_ROUTE, and for the IP version 6 environment, it is referred to as NETLINK_ROUTE6. The explanations given here are applicable for both IP versions 4 and 6.

Developers of network layer protocol handlers can use RTNETLINK to modify and monitor different components of networking, such as the routing table and network interfaces. There are many existing and upcoming protocol standards at the Internet Engineering Task Force (IETF) that can be implemented in user space. These implementations will require manipulating the routing and knowing what is being modified by other processes. Some of these protocol categories are as follows:

- Dynamic routing protocols: protocols of this category, including the Routing Information Protocol (RIP), Open Shortest Path First (OSPF) and Exterior Gateway Protocol (EGP) actively manage the routing environment of a host while communicating with other equally capable hosts or routers in the network or Internet.

- Mobility protocols: hosts that are mobile and connect to different networks at different times use protocols such as Mobile IP (MIP), Session Initiation Protocol (SIP) and Network Mobility (NEMO) to manage routing to maintain connectivity and continuity of communications.

- Ad hoc networking protocols: hosts that are mobile and located in places where there is no networking infrastructure, such as routers and WLAN access points, require peer-to-peer communications with differently configured hosts. Mobile computers of rescue workers in an earthquake-struck area or other such emergencies can use ad hoc networking protocols. These protocols, such as the Ad hoc On-demand Distance Vector (AODV) and Optimized Link State Routing (OLSR), require managing the routing to find and communicate with other hosts using neighboring hosts as routers and gateways.

It helps reduce the complexity of the kernel code if you implement these protocols in user space. Further, it simplifies the development and testing of these protocols because of the availability of many user-space development tools. Problems, such as kernel crashes, that are likely with kernel-based code when testing or when used by end users will not occur in a user-space protocol handler.

Socket Operations

The socket implementation of Linux allows two end points to communicate. The socket API provides a standard set of functions and data structures. With RTNETLINK, the two end points in communication are user space and kernel space. The following sequence of socket calls have to be made when manipulating the networking environment through RTNETLINK:

1. Open socket.
2. Bind socket to local address (using process ID).
3. Send message to the other end point.
4. Receive message from the other end point.
5. Close socket.

The socket() function opens an unattached end point to communicate with the kernel. The function prototype of this call is as follows:

```
int socket(int domain, int type, int protocol);
```

The domain refers to what type of socket is being used. For RTNETLINK, we use AF_NETLINK (PF_NETLINK). type refers to the type of protocol used when communicating. This can be raw (SOCK_RAW) or datagram (SOCK_DGRAM). This is not relevant for RTNETLINK sockets and either can be used. protocol refers to the exact NETLINK capability that we use; in our case, it is NETLINK_ROUTE. This function returns an integer with a positive number called the socket descriptor, if the socket opening was successful. This descriptor will be used in all the future RTNETLINK calls until the socket is closed. If there was a failure, a negative value is returned, and the system error variable errno included in errno.h is set to the appropriate error code.

The following is an example of a call to open an RTNETLINK socket:

```
int fd;
...
fd = socket(AF_NETLINK, SOCK_RAW, NETLINK_ROUTE);
```

Once the socket is opened, it has to be bound to a local address. The user application can use a unique 32-bit ID to identify the local address. The function prototype of bind is as follows:

```
int bind(int fd, struct sockaddr *my_addr,
                                 socklen_t addrlen);
```

To bind, the caller must provide a local address using the sockaddr_nl structure. This structure in the linux/netlink.h #include file has the following format:

```
struct sockaddr_nl
{
  sa_family_t     nl_family; // AF_NETLINK
  unsigned short  nl_pad;    // zero
  __u32           nl_pid;    // process pid
```

```
  __u32           nl_groups; // multicast grps mask
};
```

The nl_pid must contain a unique ID, which can be created using the return of the getpid() function. This function returns the process ID of the current user process that opened the RTNETLINK socket. But, if our process consists of multiple threads with each thread opening different RTNETLINK sockets, a modified process ID can be used.

Once this structure is filled, the binding can be done. The bind function returns zero if the operation succeeded. A negative number is returned in the case of failure, and the system error variable is set. The following is an example of calling bind:

```
struct sockaddr_nl la;
...
bzero(&la, sizeof(la));
la.nl_family = AF_NETLINK;
la.nl_pad = 0;
la.nl_pid = getpid();
la.nl_groups = 0;
rtn = bind(fd, (struct sockaddr*) &la, sizeof(la));
```

If the operation you require is multicast-based, you must set nl_groups to join the multicast group associated with the required RTNETLINK operation. For example, if you want to be notified of the changes to the routing table by other processes, you must OR (|) the RTMGRP_IPV4_ROUTE and RTMGRP_NOTIFY.

Sending routing RTNETLINK messages to the kernel is done through the use of the standard sendmsg() function of the socket interface. The following is the prototype of this function:

```
ssize_t sendmsg(int fd, const struct msghdr *msg,
                                        int flags);
```

msg is a pointer to a msghdr structure. The following is the format of this structure:

```
struct msghdr
{
  void *msg_name;         //Address to send to
  socklen_t msg_namelen;  //Length of address data

  struct iovec *msg_iov;  //Vector of data to send
  size_t msg_iovlen;      //Number of iovec entries

  void *msg_control;      //Ancillary data
  size_t msg_controllen;  //Ancillary data buf len

  int msg_flags;          //Flags on received msg
};
```

The msg_name is a pointer to a variable of the type struct sockaddr_nl. This is the destination address of the sendmsg() function. Because this message is directed to the kernel, all variables of sockaddr_nl will be initialized to zero, except the nl_family member variable. The field msg_namelen should contain the size of a struct sockaddr_nl.

msg_iov should contain a pointer to a struct iovec, which is filled with the RTNETLINK message relevant to the request being made. The caller is allowed to place multiple RTNETLINK requests, if required. msg_iovlen points to the number of struct iovec structures that were placed in msg_iov. The rest of the variables are initialized to zero.

To receive RTNETLINK messages, the recv() function is used. Here is the prototype of this function:

```
ssize_t recv(int fd, void *buf, size_t len,
                                        int flags);
```

The second and third variables are a pointer to a buffer to place the bytes read and the length of this buffer, respectively. For RTNETLINK, the buffer will contain a set of RTNETLINK messages that have to be read one after the other using a set of macros provided in the netlink.h and rtnetlink.h #include files. flags is a set of flags to indicate how the receive should be performed. For RTNETLINK, this simply can be initialized to zero.

Once the socket communications are complete, the socket has to be closed using the close() function. Here's the prototype of this function:

```
int close(int fd);
```

RTNETLINK Functionality

A programmer who develops applications that use RTNETLINK must include the following #include files at a minimum:

```
#include <bits/sockaddr.h>
#include <asm/types.h>
#include <linux/rtnetlink.h>
#include <sys/socket.h>
```

These files contain the different definitions, such as data types and structures, required to make RTNETLINK calls. Here is a short explanation of what is defined in these files relevant to RTNETLINK:

- bits/sockaddr.h: provides the definitions for the addresses used by socket functions.
- asm/types.h: provides the definitions of the data types used in the header files related to NETLINK and RTNETLINK.
- linux/rtnetlink.h: provides the macros and data structures that are used in RTNETLINK. Because RTNETLINK is based on NETLINK, this also includes the linux/netlink.h. netlink.h defines the general macros and structures that are used in all the NETLINK-based capabilities.
- sys/socket.h: provides the function prototypes and the different data structures related to the socket implementation.

The operations that can be invoked using RTNETLINK are defined in the rtnetlink.h file. Each of the operations provides three possibilities of manipulation: add/update, delete or query. These three possibilities are identified by NEW, DEL and GET. Following are the manipulation operations allowed by RTNETLINK.

General networking environment manipulation services:

- Link layer interface settings: identified by RTM_NEWLINK, RTM_DELLINK and RTM_GETLINK.
- Network layer (IP) interface settings: RTM_NEWADDR, RTM_DELADDR and RTM_GETADDR.
- Network layer routing tables: RTM_NEWROUTE, RTM_DELROUTE and RTM_GETROUTE.
- Neighbor cache that associates network layer and link layer addressing: RTM_NEWNEIGH, RTM_DELNEIGH and RTM_GETNEIGH.

Traffic shaping (management) services:

- Routing rules to direct network layer packets: RTM_NEWRULE, RTM_DELRULE and RTM_GETRULE.
- Queuing discipline settings associated with network interfaces: RTM_NEWQDISC, RTM_DELQDISC and RTM_GETQDISC.
- Traffic classes used together with queues: RTM_NEWTCLASS,

RTM_DELTCLASS and RTM_GETTCLASS.

- Traffic filters associated with a queuing: RTM_NEWTFILTER, RTM_DELTFILTER and RTM_GETTFILTER.

Forming and Reading RTNETLINK Messages

RTNETLINK employs a request-response mechanism to send and receive information to manipulate the networking environment. A request or a response of RTNETLINK consists of a stream of message structures. These structures are filled by the caller, in the case of a request, or filled by the kernel, in the case of a response. To place information into these structures or to retrieve information, RTNETLINK provides a set of macros (using #define statements). Every request must contain the following structure at the beginning:

```
struct nlmsghdr
{
  __u32  nlmsg_len;   //Length of msg incl. hdr
  __u16  nlmsg_type;  //Message content
  __u16  nlmsg_flags; //Additional flags
  __u32  nlmsg_seq;   //Sequence number
  __u32  nlmsg_pid;   //Sending process PID
}
```

This structure provides information about what type of RTNETLINK message is specified in the rest of the request. It is also called the NETLINK header. Here is a brief explanation of these fields:

- nlmsg_len: should contain the length of the whole RTNETLINK message, including the length of the nlmsghdr structure. This field can be filled using the macro NLMSG_ALIGN(len), where len is the length of the message that follows this structure.
- nlmsg_type: a 16-bit flag to indicate what is contained in the message, such as RTM_NEWROUTE.
- nlmsg_flags: another 16-bit flag that further clarifies the operation specified in nlmsg_type, such as NLM_F_REQUEST.
- nlmsg_seq and nlmsg_pid: these two fields are used to identify an RTNETLINK request uniquely. The caller can place the process ID and a sequence number in these fields.

Following the nlmsghdr structure are the structures relevant to the operation being requested. Depending on the type of RTNETLINK operation, the caller must include one or more of the following structures. These are called the RTNETLINK operation headers:

- struct rtmsg: retrieving or modifying entries of the routing table requires the use of this structure.
- struct rtnexthop: a next hop in a routing entry is the next host to consider on the way to the destination. A single routing entry can have multiple next hops. Each next hop entry has many types of attributes, such as the network interface in addition to the next hop IP address.
- struct rta_cacheinfo: each route entry consists of status information that the kernel updates regularly, mainly usage information. Using this structure, a user can retrieve this information.
- struct ifaddrmsg: retrieving or modifying network layer attributes associated with a network interface requires the use of this structure.
- struct ifa_cacheinfo: similar to a route entry, a network interface also consists of information about its status, which is updated by the kernel. This structure is used to retrieve this information.
- struct ndmsg: retrieving or modifying the association information between link layer addressing and network layer addressing of neighbors, referred to as neighbor discovery, is specified through this structure.
- struct nda_cacheinfo: holds the kernel updated information related to each neighbor discovery entry.
- struct ifinfomsg: retrieving or modifying the link layer attributes related to a network interface requires the use of this structure.
- struct tcmsg: retrieving or modifying traffic shaping attributes is supplied using this structure.

Following the RTNETLINK operation header are the attributes related to the operation, such as an interface number and IP address. These attributes are specified using the struct rtattr. There is one structure for each attribute. This structure has the following format:

```
struct rtattr
{
  unsigned short  rta_len;
  unsigned short  rta_type;
};
```

Immediately following this structure is the value of the attribute. An attribute such as an IP version 4 address will occupy a 4-byte area. The variable rta_len should contain the size of this structure plus the size of the attribute. rta_type should contain the value identifying the attribute, which are given in the enumerations defined in rtnetlink.h. enum rtattr_type_t and other enumerations provide the attribute identifiers, such as IFA_ADDRESS and NDA_DST, to be used in this field. The maximum number of attributes that you can attach is up to only the macro RTATTR_MAX. An example of attaching an attribute is as follows:

```
rtap->rta_type = RTA_DST;
rtap->rta_len = sizeof(struct rtattr) + 4;
inet_pton(AF_INET, dsts,
     ((char *)rtap) + sizeof(struct rtattr));
```

Information that is received from an RTNETLINK socket is again a stream of structures. A programmer has to identify and extract information by moving a pointer along this byte stream. To simplify this process, RTNETLINK provides a set of macros to make the buffer positioning easier:

- NLMSG_NEXT(nlh, len): returns the pointer to the next NETLINK header. nlh is the header returned previously, and len is the total length of the message. This will be called in a loop to read every message.
- NLMSG_DATA(nlh): returns the pointer to the RTNETLINK header related to the requested operation given the NETLINK header. If a route entry is being manipulated, this will return a pointer to a struct rtmsg.
- RTM_RTA(r), IFA_RTA(r), NDA_RTA(r), IFLA_RTA(r) and TCA_RTA(r): return a pointer to the start of the attributes of the respective RTNETLINK operation given the header of the RTNETLINK message (r).
- RTM_PAYLOAD(n), IFA_PAYLOAD(n), NDA_PAYLOAD(n), IFLA_PAYLOAD(n) and TCA_PAYLOAD(n): return the total length of the attributes that follow the RTNETLINK operation header given the pointer to the NETLINK header (n).
- RTA_NEXT(rta, attrlen): returns a pointer to the start of the next

attribute given the last returned attribute (rta) and the remaining size (attrlen) of the attributes.

Considering a simple example where an RTNETLINK request to retrieve the routing table was sent, the reply is processed in the following manner:

```
char *buf; // ptr to RTNETLINK data
int nll; // byte length of all data
struct nlmsghdr *nlp;
struct rtmsg *rtp;
int rtl;
struct rtattr *rtap;

nlp = (struct nlmsghdr *) buf;
for(;NLMSG_OK(nlp, nll); nlp=NLMSG_NEXT(nlp, nll))
{

  // get RTNETLINK message header
  rtp = (struct rtmsg *) NLMSG_DATA(nlp);

  // get start of attributes
  rtap = (struct rtattr *) RTM_RTA(rtp);

  // get length of attributes
  rtl = RTM_PAYLOAD(nlp);

  // loop & get every attribute
  for(;RTA_OK(rtap, rtl); rtap=RTA_NEXT(rtap, rtl))
  {

    // check and process every attribute

  }
}
```

RTNETLINK Sample Walk-Through

The sample code presented here focuses on three of the operations that can be performed on the routing table:

- get_routing_table: reads the main routing table in the system.
- set_routing_table: inserts a new routing entry to the table.
- mon_routing_table: monitors the routing table changes.

All three samples use a similar main() function that calls a set of subfunctions to form RTNETLINK messages and send, receive and process the received messages. To simplify the explanation, no error handling is considered. These samples perform on the IP version 4 environment of the system (AF_INET). Here is the main() function:

```
int main(int argc, char *argv[])
{

  // open socket
  fd = socket(AF_NETLINK, SOCK_RAW, NETLINK_ROUTE);

  // setup local address & bind using
  // this address
  bzero(&la, sizeof(la));
  la.nl_family = AF_NETLINK;
  la.nl_pid = getpid();
  bind(fd, (struct sockaddr*) &la, sizeof(la));

  // sub functions to create RTNETLINK message,
  // send over socket, receive reply & process
  // message
  form_request();
  send_request();
  recv_reply();
  read_reply();

  // close socket
  close(fd);
}
```

Similar to the above function, the two functions that perform the socket communications are almost common to all the samples. These two functions simply send a formed message to the kernel and receive messages sent by the kernel. Exceptions here are the set_routing_table and mon_routing_table samples. In set_routing_table, a receive phase is not considered. In the mon_routing_table, a send phase is not present as it attempts to monitor only the state of the routing environment to see what is being changed. This information is mulitcast by the kernel to all the RTNETLINK sockets that are in the appropriate receiving state.

First, here's the code for send_request():

```
void send_request()
{
  // create the remote address
  // to communicate
  bzero(&pa, sizeof(pa));
  pa.nl_family = AF_NETLINK;

  // initialize & create the struct msghdr supplied
  // to the sendmsg() function
  bzero(&msg, sizeof(msg));
  msg.msg_name = (void *) &pa;
  msg.msg_namelen = sizeof(pa);

  // place the pointer & size of the RTNETLINK
  // message in the struct msghdr
  iov.iov_base = (void *) &req.nl;
  iov.iov_len = req.nl.nlmsg_len;
  msg.msg_iov = &iov;
  msg.msg_iovlen = 1;

  // send the RTNETLINK message to kernel
  rtn = sendmsg(fd, &msg, 0);
}
```

And here's the recv_reply():

```
void recv_reply()
{
  char *p;

  // initialize the socket read buffer
  bzero(buf, sizeof(buf));

  p = buf;
  nll = 0;

  // read from the socket until the NLMSG_DONE is
  // returned in the type of the RTNETLINK message
  // or if it was a monitoring socket
  while(1) {
    rtn = recv(fd, p, sizeof(buf) - nll, 0);
```

```
    nlp = (struct nlmsghdr *) p;

    if(nlp->nlmsg_type == NLMSG_DONE)
      break;

    // increment the buffer pointer to place
    // next message
    p += rtn;

    // increment the total size by the size of
    // the last received message
    nll += rtn;

    if((la.nl_groups & RTMGRP_IPV4_ROUTE)
                      == RTMGRP_IPV4_ROUTE)
      break;
  }
}
```

The above functions and the following ones use a set of globally defined variables. These are used for all the socket operations as well as for forming and processing RTNETLINK messages:

```
// buffer to hold the RTNETLINK request
struct {
  struct nlmsghdr  nl;
  struct rtmsg     rt;
  char             buf[8192];
} req;

// variables used for
// socket communications
int fd;
struct sockaddr_nl la;
struct sockaddr_nl pa;
struct msghdr msg;
struct iovec iov;
int rtn;

// buffer to hold the RTNETLINK reply(ies)
char buf[8192];

// RTNETLINK message pointers & lengths
// used when processing messages
struct nlmsghdr *nlp;
int nll;
struct rtmsg *rtp;
int rtl;
struct rtattr *rtap;
```

The get_routing_table sample retrieves the main routing table of the IPv4 environment. The form_request() function is as follows:

```
void form_request()
{
  // initialize the request buffer
  bzero(&req, sizeof(req));

  // set the NETLINK header
  req.nl.nlmsg_len
              = NLMSG_LENGTH(sizeof(struct rtmsg));
  req.nl.nlmsg_flags = NLM_F_REQUEST | NLM_F_DUMP;
  req.nl.nlmsg_type = RTM_GETROUTE;

  // set the routing message header
  req.rt.rtm_family = AF_INET;
  req.rt.rtm_table = RT_TABLE_MAIN;
}
```

The received message for the RTNETLINK request in the buf variable to retrieve the routing table is processed by the read_reply() function. Here is the code of this function:

```
void read_reply()
{
  // string to hold content of the route
  // table (i.e. one entry)
  char dsts[24], gws[24], ifs[16], ms[24];

  // outer loop: loops thru all the NETLINK
  // headers that also include the route entry
  // header
  nlp = (struct nlmsghdr *) buf;
  for(;NLMSG_OK(nlp, nll);nlp=NLMSG_NEXT(nlp, nll))
  {

    // get route entry header
    rtp = (struct rtmsg *) NLMSG_DATA(nlp);

    // we are only concerned about the
    // main route table
    if(rtp->rtm_table != RT_TABLE_MAIN)
      continue;

    // init all the strings
    bzero(dsts, sizeof(dsts));
    bzero(gws, sizeof(gws));
    bzero(ifs, sizeof(ifs));
    bzero(ms, sizeof(ms));

    // inner loop: loop thru all the attributes of
    // one route entry
    rtap = (struct rtattr *) RTM_RTA(rtp);
    rtl = RTM_PAYLOAD(nlp);
    for(;RTA_OK(rtap, rtl);rtap=RTA_NEXT(rtap,rtl))
    {
      switch(rtap->rta_type)
      {
        // destination IPv4 address
        case RTA_DST:
          inet_ntop(AF_INET, RTA_DATA(rtap),
                                      dsts, 24);
          break;

        // next hop IPv4 address
        case RTA_GATEWAY:
          inet_ntop(AF_INET, RTA_DATA(rtap),
                                      gws, 24);
          break;

        // unique ID associated with the network
        // interface
        case RTA_OIF:
          sprintf(ifs, "%d",
                    *((int *) RTA_DATA(rtap)));
        default:
          break;
      }
    }
    sprintf(ms, "%d", rtp->rtm_dst_len);
```

```
    printf("dst %s/%s gw %s if %s\n",
                            dsts, ms, gws, ifs);
  }
}
```

The set_routing_table sample sends an RTNETLINK request to insert an entry to the routing table. The route entry that is inserted is a host route (32-bit network prefix) to a private IP address (192.168.0.100) through interface number 2. These values are defined in the variables dsts (destination IP address), ifcn (interface number) and pn (prefix length). You can run the get_routing_table sample to get an idea about the interface numbers and the IP network in your system. Here's the form_request():

```
void form_request()
{
  // attributes of the route entry
  char dsts[24] = "192.168.0.100";
  int ifcn = 2, pn = 32;

  // initialize RTNETLINK request buffer
  bzero(&req, sizeof(req));

  // compute the initial length of the
  // service request
  rtl = sizeof(struct rtmsg);

  // add first attrib:
  // set destination IP addr and increment the
  // RTNETLINK buffer size
  rtap = (struct rtattr *) req.buf;
  rtap->rta_type = RTA_DST;
  rtap->rta_len = sizeof(struct rtattr) + 4;
  inet_pton(AF_INET, dsts,
     ((char *)rtap) + sizeof(struct rtattr));
  rtl += rtap->rta_len;

  // add second attrib:
  // set ifc index and increment the size
  rtap = (struct rtattr *) (((char *)rtap)
            + rtap->rta_len);
  rtap->rta_type = RTA_OIF;
  rtap->rta_len = sizeof(struct rtattr) + 4;
  memcpy(((char *)rtap) + sizeof(struct rtattr),
           &ifcn, 4);
  rtl += rtap->rta_len;

  // setup the NETLINK header
  req.nl.nlmsg_len = NLMSG_LENGTH(rtl);
  req.nl.nlmsg_flags = NLM_F_REQUEST | NLM_F_CREATE;
  req.nl.nlmsg_type = RTM_NEWROUTE;

  // setup the service header (struct rtmsg)
  req.rt.rtm_family = AF_INET;
  req.rt.rtm_table = RT_TABLE_MAIN;
  req.rt.rtm_protocol = RTPROT_STATIC;
  req.rt.rtm_scope = RT_SCOPE_UNIVERSE;
  req.rt.rtm_type = RTN_UNICAST;
  // set the network prefix size
  req.rt.rtm_dst_len = pn;
}
```

The mon_routing_table sample reads the RTNETLINK messages received when other processes change the system's main routing table. This function will use the same read_reply() function to process the messages. The main() function requires a slight change. Because this operation involves listening to multicast messages of the kernel, the local address to which we bind, it also must include the two flags RTMGRP_IPV4_ROUTE and RTMGRP_NOTIFY. Here is the required change:

```
la.nl_groups = RTMGRP_IPV4_ROUTE | RTMGRP_NOTIFY;
```

Once mon_routing_table is executed, run a `route add` or a `route del` command from another shell prompt to see the results.

Conclusion

RTNETLINK is a simple, yet versatile way of manipulating the networking environment of a Linux host. User-space network protocol handlers are ideal candidates for using RTNETLINK. The advanced IP routing command suite, referred to as IPROUTE2, is based on RTNETLINK. More information about the different operations and flags of RTNETLINK can be found at NETLINK(7) and RTNETLINK(7).

The sample code for this article is available at **ftp.ssc.com/pub/lj/listings/issue145/8498.tgz**.

Acknowledgement

I sincerely thank Professor Carmelita Goerg.■

Asanga Udugama (adu@comnets.uni-bremen.de) is a researcher/software developer for ComNets, University of Bremen, Germany. He currently works on implementing and improving the IETF standards for mobility-related network layer protocols. He has a few reference implementations to his credit. At the moment, he is awaiting the arrival of his dream wheelchair (Meyra X3 with Servomatic).

Top Ten Tips for Getting Started with PHP

Here are ten tips that will help you avoid some of the most common pitfalls when coding Web applications in PHP.

MARCO FIORETTI

There is little doubt that PHP is one of the easiest languages to use to start generating dynamic Web content. PHP, in combination with Linux, Apache and MySQL is so popular, it has spawned the expression LAMP (Linux, Apache, MySQL and PHP). Many pages go on-line without any need for their authors to set up or program anything themselves. They simply find some pre-cooked piece of code with a search engine, paste it as is into an HTML template, upload everything to their Web server, and they are done.

Or so they believe. Even previous programming experience may not help much, because coding for a desktop or for the Web are two very different paradigms. Therefore, pretty often, when people cut and paste PHP code, nothing happens (nothing good, at least). The pages load *very* slowly or worse, the programmer's choice of PHP code opens a new security hole.

The tips below are written especially for users who already know the basics of programming, but who have never touched PHP before. They might be roughly divided in three categories: how to start correctly, how not to hurt yourself and, finally, how to make their code more efficient. Due to space constraints and the fact that there already is plenty of good on-line and paper documentation for PHP, most tips explain only what to look for and why.

1 Check Whether Everything Was Installed and Configured Correctly

One common source of confusion for PHP beginners is to upload their first Web page on some server and see only the PHP/HTML source code in the browser instead of the expected content. This happens because the Web server doesn't recognize the file as something that should be passed to the PHP interpreter. The reason for this is that the system administrator forgot to associate the PHP file with the PHP interpreter. You can do this in the Apache configuration file or in a local .htaccess file. Here is a sample configuration line:

```
AddType application/x-httpd-php .php3 .php
```

As a matter of fact, it is possible to know how things stand simply by uploading this really short page to your Web space:

```
<HTML>
<HEAD>
<TITLE>PHP Configuration Check</TITLE>
</HEAD>
<BODY><? php phpinfo() ?>
</BODY>
</HTML>
```

With any luck, the result will be similar to what is shown in Figure 1. The phpinfo() function prints out how PHP was compiled and the value of all configuration variables. This function gives you a lot of useful information. Its output probably will be the very first thing you'll be asked for whenever you seek support on an on-line PHP forum.

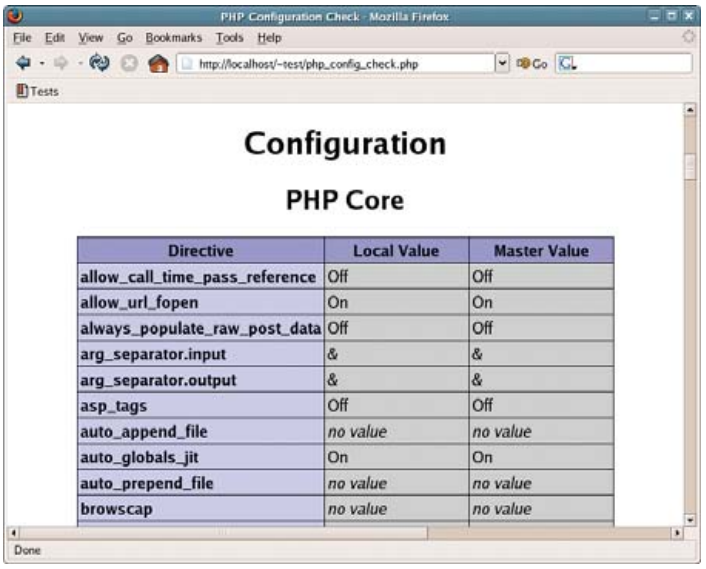


Figure 1. Sample PHP Information Generated by the phpinfo() Function

2 Let PHP and the Script Tell You about Your Errors

In order to speed up debugging, you can tell both PHP and the Apache Web server which errors must be reported and when. The error_reporting variable in the php.ini configuration file can be seen as a series of (bit) flags. Each of them can be set individually to detect (or not) a specific category of errors. This instruction, for example:

```
error_reporting = E_ALL
```

sends anything from simple warnings to serious bugs to the browser, but only if the other variable display_errors is turned on. General PHP settings in the php.ini file can be overridden at the Web server level. When using Apache, the instruction equivalent to the one above would be (in httpd.conf):

```
php_flag  display_errors      on
php_value error_reporting  2047
```

Should you have no access to the PHP/Web server configuration, as often happens, the same result can be accomplished by adding this command to your scripts:

```
error_reporting(E_ALL);
```

Speaking of Web servers, remember also to check their error logs to

know exactly which line of code caused a script to crash.

If a script still fails after all these tricks have ceased to find any error, almost surely the bug is in the script logic itself. Somewhere, some variable is assigned a value that you thought not possible for it, and this confuses the rest of the code. This also applies when the variable is actually some SQL statement built on the fly and passed to a database server.

The solution is to display that variable on your browser. You can do this easily with the print() instruction normally used to send HTML code to the browser. The die() statement does the same thing as print(), but it also stops the script immediately afterward.

3 Headers before Anything Else!

You can generate and transmit any kind of HTTP header before even starting to build the actual Web page. However, you must remember that header() has to be called before any HTML code or PHP output, including blank or empty lines! Code like this, for example:

```
<?php /* any PHP command(s) here */ ?>

<?php header("Content-type: image/png"); ?>
```

will not work. The mere presence of the empty line between the two encoded PHP statements will cause PHP to transmit standard HTTP headers, which almost always will not be what you wanted (otherwise you would not have used that function). Note that the blank line may even be...in another file. That is, the same thing will happen if you load PHP code from some external file that doesn't end exactly with the closing ?> PHP tag.

This is a frequent cause of headaches for programmers who build sites that use cookies. The only way to make cookies work is to handle them before your PHP program sends header information. If you don't realize that a simple blank line sends header information, you can stare at your code for hours wondering why you are having problems with cookies. After all, you do handle cookies before you deliberately send the header. What you don't necessarily notice is that there's a blank line in your program (or included file) that is sending headers without your knowledge, which is why your cookies don't work.

4 Always Check User Data (and Beware of E-mail Addresses)

You should always validate data that your pages receive from the Web. JavaScript routines that validate form input on the user browser are useless security-wise. Nothing prevents a cracker from sending malicious data directly to your code. Imagine a PHP shopping cart that can show all the items below the $HIGHEST_PRICE decided by the user. If, without previous checks, you merrily performed a database query with a $HIGHEST_PRICE whose value is something like "delete * from my_database;", don't complain when your on-line store looks empty!

You can validate data using a combination of three techniques. The first is to analyze the data with regular expressions that explicitly define only the formats that are allowed; a phone number or year of birth, for example, can contain only digits, so pass it through the function is_digit().

The second is to use other functions like EscapeShellCmd(), which can block "data" from executing unwanted system commands, or mysql_escape_string() on variables that must be inserted into an SQL statement.

The last type of validation strictly depends on the actual meaning of a variable and the context in which it is used. Only you can help yourself here. For example, 5555555 is made only of digits, but (in North America) it is not a valid phone number. It should be allowed only if the user declared to be from another country. Similarly, although 18 is a perfectly valid $AGE, a script offering discounts to senior citizens should refuse it, right?

E-mail addresses are particularly troublesome from this point of view. There are several functions that validate their syntactical correctness, like the one at **www.zend.com/tips/tips.php?id=224&single=1**. They do nothing, however, to guarantee that an address does belong to the person who sent it, or that it exists at all, such as Luke.Skywalker@whitehouse.gov. Well, it's probably a safe assumption that there is no Luke.Skywalker in the White House, anyway. Always ask users to reply to a confirmation message or open a socket to their mail server to check whether they exist.

5 Properly Manage Quotes and Escapes

What will appear in your browser if you load this very simple PHP code?

```
<? php
$HOME = 'a sweet place';
print "1: $HOME<br>"; // double quotes
print '2: $HOME<br>'; // single quotes
?>
```

The answer is these two lines of text:

```
1: a sweet place
2: $HOME
```

Double quotes make PHP replace any variable inside them with its current value. The content of single quotes is treated like one monolithic, opaque block that can be copied or printed only, not modified. The same applies when you use quotes to build the keys of an associative array. $my_array["$HOME"] and $my_array["$HOME"] will be different elements. That's it. Still, it is very easy to forget this distinction and use one when you meant the other, or no quote at all. Therefore, when something doesn't have the value you expected, check the quotes first.

Because user data cannot be trusted, PHP can be set up to escape with slashes automatically with all the $_POST sent by an HTML form to the script. Actually, even internal data could contain slashes, to escape special characters, which must be removed before processing them. The solution is to use the stripslashes function, as in this example straight from the on-line PHP manual:

```
<?php
$str = "Is your name O\'reilly?";
// Outputs: Is your name O'reilly?
echo stripslashes($str);
?>
```

6 Let the Database Do the Work Instead of Your Script

As stated above, PHP is used together with MySQL so often that the LAMP acronym is one of the most well-known combinations in Web design. Consequently, one of the best ways to write faster PHP scripts is to learn MySQL well enough that it works as much as possible instead of PHP. These two snippets of code illustrate the concept:

```
<?php //find all the books that Asimov wrote after 1980
$sql = "select YEAR, BOOK from MY_BOOKSHELF where AUTHOR
➥LIKE 'Asimov' ; ";
  if ($sql_res = mysql_query("$sql")) {
    while ($r  = mysql_fetch_array($sql_res)) {
      if ($r[YEAR] > 1980) {// print the book title ;}
   }
?>
```

And:

```
<?php //find with MySql all the books that Asimov wrote
➥after 1980
$sql = "select BOOK from MY_BOOKSHELF where AUTHOR LIKE
➥'Asimov'  AND YEAR > 1980;";
  if ($sql_res = mysql_query("$sql")) {
    while ($r = mysql_fetch_array($sql_res)) {
      // just print all the returned titles ;
}
?>
```

The second version will run much faster than the first, because database engines are designed to select as quickly as possible all and only the data matching any combination of criteria. They'll always be much faster than PHP is in this kind of task. Therefore, make sure that as much as possible of your selection logic is inside the SQL query, not in the PHP code that builds and uses it. Of course, this whole tip applies as is to any other database engine you would use with PHP.

7 Write Portable File Management Code

Line endings in text files are encoded differently on each family of operating systems. Binary files, such as images or compressed archives, are much worse, in the sense that even one corrupted character can make the whole file useless. Practically speaking, this means it is up to you to write code that will manage file contents in the same way on any platform you might use. This remains true even if you are sure that you and your Web server will always and only run GNU/Linux. Otherwise, you could find no error in your image or text processing code until you use it to upload a file from the Windows or Apple computer of a friend!

As far as PHP is concerned, the solution is to make proper use of the t (text mode translation) and b (binary) flags of the fopen() system call. The gory details are at **www.php.net/function.fopen**. Note that the page explicitly suggests: "for portability, it is also strongly recommended that you re-write code that uses or relies upon the t mode."

8 Know String Processing Functions

Web pages still are mostly made of text, and the same is true for many databases. This is why optimizing text analysis and processing is one of the easiest ways to make all of your scripts run faster. Regular expressions are made to order for such jobs, but they look like hieroglyphics and may not even always be the optimal solution. PHP, although not going to the same extremes (uh, we mean power and flexibility of Perl), has more than one function working just like regular expressions, only much quicker. We refer to str_replace(), strcmp(), strtolower(), strtoupper(), strtr(), substr(), trim(), ucfirst() and several others. Take some time to study them in the manual, it will be well worth it.

9 Keep Layout and Programming Separate

A sure way to make the source of any Web site unreadable and difficult to update is to interlace large chunks of PHP and HTML code, even if each piece of PHP is used only once in the page, as in this example:

```
myfile.php>
<!— lots of HTML code for static header, logo, menus...>
<?php lots of PHP code generating a list of the latest news ?>
<!— lots of HTML code for the central part of the page...>
<?php lots of PHP code creating a per-user list of the
   most popular pages ?>
<!— lots of HTML code for the user feedback form...>
```

Instead of making this error, encapsulate every piece of PHP code in one or more functions, then put them all in one separate file (without any HTML code), which will be loaded with the include_once command. The result will be much cleaner and easier to maintain:

```
myfile.php>
<?php include_once ("common_code.php"); ?>
<!— lots of HTML code for the static page header, logo,
menus...>
<?php show_latest_news (); /* only one function call */ ?>
<!— lots of HTML code for the central part of the page...>
<?php show_most_popular_pages (); /* only one function call */ ?>
<!— lots of HTML code for the user feedback form...>
```

Anther big advantage of this approach is that, by simply including common_code.php as shown above, any page of your Web site will be able to use those same functions. Even more important, should any function be modified, the new version would be available immediately in all the pages.

10 Check the Results of Function and System Calls

Last but not least, *all* PHP functions must return acceptable data to the code that called them. The tricky part of this apparently superfluous statement is the fact that the meaning of acceptable depends on the whole script, and it may be different at any time. Here is a very dumb, but effective example of what we mean:

```
function subtraction($A, $B) {
         $diff = $A - $B;
         return($diff);
}
   $C = 1/subtraction(3, 3);       // ERROR! Division by Zero!
   $D = 1/(1 - subtraction(3,3);
```

Although calculating $C will make the script crash, calculating (with the same operands), $D will not. The point is that before doing anything with a variable, you should check that it has an acceptable value. In the example above, this would mean assigning the subtraction result to an auxiliary variable and proceeding with the division only if it is non-null.

It is even more important to check return values from system calls, that is, the built-in functions provided to allow interaction with external processes and files. Should you forget to check a return value, data could be thrown away without anyone noticing, as in this example:

```
<?php
$HANDLE = fopen("newuser.txt","w")); // open a file
fwrite($HANDLE, "New User Data");    // write to it
?>
```

If fopen fails (because, for example, the disc is full or you had no permission to write) the New User Data is lost for good. Before writing, check that $HANDLE is not null:

```
<?php
if (!$HANDLE = fopen("newuser.txt","w")) { die "File
➥access failed: newuser.txt"; }
fwrite($HANDLE, "New User Data");
?>
```

Happy PHP coding!■

Marco Fioretti is a hardware systems engineer interested in free software both as an EDA platform and, as the current leader of the RULE Project, as an efficient desktop. Marco lives with his family in Rome, Italy.

The 64-Bit Question

Linux is blowing its opportunity to be the best AMD64 platform for all needs.

Nick Petreley, Editor in Chief

I haven't upgraded my main workstation in years, so it is about time for a change. I've heard nothing but great things about the AMD64 processor, so that's the route I take.

I decide to shop for a lower-speed dual-core AMD64 processor instead of a high-speed single-core processor. I want dual-core, because I'm not looking for the ultimate gaming machine. I want a good "lets me compile, burn a DVD, and maybe do two or three other things at the same time and still have a very responsive desktop" machine. I find that I can put together a decent dual-core AMD64 box with a PCI Express x16 NVIDIA display card, two huge SATA drives and 4GB of RAM for a surprisingly reasonable price. I get most of my stuff on-line from **www.newegg.com**, which has some pretty good deals.

A few days later, most of my stuff arrives. It takes about a day to tinker together the new computer between editing sessions on the old one. With the exception of a couple of dumb mistakes, everything comes together, and the new computer is running fine.

My SATA DVD drive hasn't arrived yet. I use my old IDE DVD drive to install the AMD64 version of Kubuntu. This turns out to be an accidental lucky break, as you'll see in a moment.

I upgrade Kubuntu AMD64 to use the K8-SMP version of the kernel in order to take advantage of my dual-core processor. Big mistake. The SMP kernel crashes. A lot. I check the forums, and it crashes a lot for many Kubuntu/Ubuntu users. Kubuntu released an updated version of the kernel, but it still crashes, just not as often.

So I compile my own 2.6.15.4 version of the kernel. It doesn't support the graphical boot screen, and it reports some meaningless errors. I don't care. My version of the kernel is stable and it supports all my hardware.

My SATA DVD drive arrives. I remove my IDE drive and install the SATA drive. My Linux kernel can't recognize the drive. Why? Because there is a kernel driver parameter called atapi_enabled that needs to be set to 1 in order for the kernel to recognize the DVD/CD drive. I modify the source code to change the default and recompile. That works. (You don't have to modify the source, but it's beyond the scope of this rant to explain why I chose that method over the alternatives.)

Then I attempt to install the AMD64 version of Debian. Thanks to the default atapi_enabled=0 in the Debian kernel on the DVD, the installation program can't find the DVD/CD drive in order to install the software. I can't find any way to change the parameter to 1. I read that the kernel boot option libata.atapi_enabled=1 should work, but it doesn't. So I can't install Debian AMD64 (or probably any other distribution) unless I put an additional IDE DVD/CD drive in the machine. That's nuts.

Then I run Firefox under Kubuntu. The latest official Kubuntu version is 1.07. I want to use 1.5 or later, and there is no AMD64 version available. So I compile one myself. That's fine, but no AMD64 version of Firefox can run Flash because there is no 64-bit version of Flash. There is an AMD64 version of Java, but it doesn't include a plugin library, so I can't run AMD64 Java from this browser, either. The Ubuntu forums explain how to get 32-bit Firefox working with Flash and Java without having to chroot to a 32-bit environment, but those instructions don't work for me. I eventually track the problem down to another kernel configuration option.

I suppose I can avoid all these problems by running i386 versions of Linux on this machine. I suppose I could also have avoided all these problems if I had researched existing support for AMD64 and chose not to go with an AMD64 chip. But I didn't think I had to research AMD64 support. The AMD64 is practically an old chip now by modern standards. There are lots of notebook computers that use a version of the AMD64, so don't tell me it's a server-only chip.

Granted some of the problems are external to Linux, such as the lack of an AMD64 Java plugin library and lack of AMD64 Flash plugin. But there's no excuse for Linux being unable to recognize a SATA DVD/CD drive, no matter what chip you're using. And since Ubuntu shows that it is possible with customization, there's no excuse for any AMD64 distribution failing to let you run 32-bit Firefox with plugins right out of the box.

The 64-bit version of Windows is still struggling. So there is still a window of opportunity for Linux to be the first, best AMD64 desktop platform. It's high time distro developers (and third parties) got off their arses to help make this possible.■

Nicholas Petreley is Editor in Chief of *Linux Journal* and a former programmer, teacher, analyst and consultant who has been working with and writing about Linux for more than ten years.

The World's Fastest InfiniBand Switch

From a Company You've Trusted for 23 Years

▲ 72 Port FasTree™ Configuration

Microway's **FasTree**™ DDR InfiniBand switches run at 5GHz, twice as fast as the competition's SDR models. FasTree's non-blocking, flow-through architecture makes it possible to create 24 to 72 port modular fabrics which have lower latency than monolithic switches. They aggregate data modulo 24 instead of 12, improving nearest neighbor latency in fine grain problems and doubling the size of the largest three hop fat tree that can be built, from 288 to 576 ports. Larger fabrics can be created linking 576 port domains together.

Working with PathScale's InfiniPath HTX Adapters, the number of hops required to move MPI messages between nodes is reduced, improving latency. The modular design makes them useful for SDR, DDR and future QDR InfiniBand fabrics, greatly extending their useful life. Please send email to fastree@microway.com to request our white paper entitled *Low Latency Modular Switches for InfiniBand.*

Microway's **QuadPuter**® includes four AMD single or dual core Opteron™ processors, 1350 Watt redundant power supply, and up to 5 redundant, hot swap hard drives-all in 4U. Dual core enables users to increase computing capacity without increasing power requirements, thereby providing the best performance per watt. Constructed with stainless steel, QuadPuter's RuggedRack™ architecture is designed to keep the processors and memory running cool and efficiently. Hard drives are cooled with external air and are front-mounted along with the power supply for easy access and removal. The RuggedRack™ is also available with an 8-way motherboard and up to 128 GB of memory for power- and memory-hungry SMP applications.

Call us first at 508-746-7341 for quotes and benchmarking services. Find technical information, testimonials, and newsletter at microway.com.

◀ QuadPuter® Navion™ with hot swap, redundant power and hard drives and dual core Opterons, offering the perfect balance between performance and density

AMD 64 Opteron™

PathScale™ Accelerating Cluster Performance™

Microway®

Technology you can count on℠

508.746.7341 microway.com