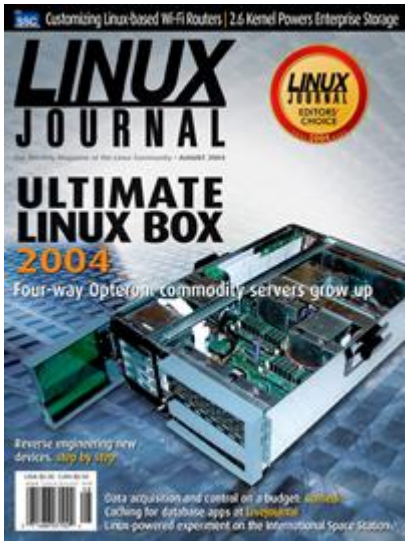
Linux Journal Issue #124/August 2004



Features

Ultimate Linux Box 2004 *by Paul Bibaud, Jesse Keating, Cosmo King, Eric Logan, Micah Spacek, Tim Lee and Don Marti*
We take a peek at a no-compromises system that will give everyone some PC construction ideas.

Linux on Linksys Wi-Fi Routers *by James Ewing*
This sub-$100 wireless box has 16MB of RAM and a 125MHz processor. Put it to work.

Indepth

2004 Editors' Choice Awards *by LJ Staff*
Our newly expanded team of experts comes to some surprising conclusions on the year's best products and projects.

Linux Serial Consoles for Servers and Clusters *by Matthew E. Hoskins*
Keep your servers under control with one cable, not a rackload.

Distributed Caching with Memcached *by Brad Fitzpatrick*
Speed up your database app with a simple, fast caching layer that uses your existing servers' spare memory.

Data Acquisition with Comedi *by Caleb Tennis*
Whatever you're discovering or inventing, now you can use any data acquisition card with the same API.

Declic: Linux 2.6 on the International Space Station *by Taco Walstra*
Linux fits into this new research program in several ways, from meeting real-time requirements with the 2.6 kernel to offering a prototyping platform for microcontroller code.

Embedded

Driving Me Nuts *by Greg Kroah-Hartman*

Toolbox

At the Forge Weblogs and Slash *by Reuven M. Lerner*
Kernel Korner Storage Improvements in 2.6 and for 2.7 *by Paul E. McKenney*
Cooking with Linux The Ultimate Cooking Box *by Marcel Gagné*

Columns

Linux for Suits Missing Pieces *by Doc Searls*
EOF Open Source Is for Pigs *by Evan Leibovitch*

Departments

From the Editor
Letters
upFRONT
On The Web
Best of Technical Support
New Products

Archive Index

Advanced search

LINUX JOURNAL

Ultimate Linux Box 2004

Paul Bibaud

Jesse Keating

Cosmo King

Eric Logan

Micah Spacek

Tim Lee

Don Marti

Issue #124, August 2004

As four-processor x86-64 boards take over the high end of homebrew PCs, we take a look at the systems you'll be building next year.

You might say that the trend to 64-bit Linux systems started in 1994 when Jon “maddog” Hall, then at Digital, gave Linus Torvalds an Alpha workstation. But the mass market started to make the move last year when AMD introduced the first x86-64 processors. AMD calls the architecture AMD64, and Intel has followed up with compatible processors under the name IA-32e. We selected a two-way AMD64 system as Ultimate Linux Box last year.

Our Ultimate Linux Boxes have sported two processors since 2000, and it's time to make the move to four. Now that major vendors are offering Linux systems in sizes up to SGI's 256 Itanium processors in its Altix 3000 series, we have to make it clear that this is the ultimate box you can build, not the ultimate box that anyone has ever built.

Although we probably say it every year, there's never been a better selection of Linux-compatible hardware on the market. IBM has launched a major marketing push for Linux on POWER, and some people are talking up Apple's PowerPC-based Power Mac G5 as great for Linux. The Ultimate Linux Box

always has been about a system that readers can build, however; so we're going to go where the commodity hardware is.

Timing for this article was a little awkward. Too late to make it into this year's box, Tyan recently introduced the four-way SMP Thunder K8QS (S4880), which is in a new, larger size known as SSI: 13" × 16" or 330mm × 407mm. Cases that fit are rare. Still, it's the first industry-standard four-way, 64-bit motherboard, and we're thinking about putting one like it into a tower case next year—a big tower case, that is.

But, it's clear that four-way x86-64 motherboards are the new top of the line for Linux box builders, so we're getting a head start on the trend by using one of the rackmount bare-bones systems, the Celestica A8440. Both the Celestica and another four-way, the Newisys 4300, are popular basic boxes on which Linux box builders are developing complete systems.

Recommended Hardware

Although user-group mailing lists and other community fora are great for answering many technical questions, they tend to be less good for advice on what hardware to buy. Unfortunately, you're likely to get secondhand media reviews, justifications of random stuff someone recently bought and just plain errors.

But that's fine, because the Linux scene already has an excellent source of hardware recommendations—the system specs pages on the small Linux vendors' sites. If someone in the Linux business is willing to take phone calls about a particular piece of hardware and stays in business, that's a pretty good sign.

Some of the hardware the small shops use is on the expensive side. You see a lot of Supermicro and Tyan motherboards and Seagate and Maxtor hard drives, for example. But the good news is high-quality PC hardware doesn't command as much of a price premium as it should. Commentators make such a big deal out of PC hardware being a commodity that people ignore the fact that even commodities have different quality levels. As long as the “a PC is a PC” meme stays current, the market undervalues quality hardware.

Linux vendors don't mind home builders free riding on their hard-earned hardware choices, because hardly anyone builds PCs for work. If you bookmark a company's hardware choices as a reference for your home projects, you're likely to come back to them when it's time to order.

Buy or Build?

If you're reading this far, you probably have strong opinions about your system's details, including the visible parts. If you want a cool-looking case or a weird combination of hardware, you're likely to want to build. You can save some money that way too. When you build, you can splash out on expensive boutique ball-bearing fans, heavy but quiet heat sinks and other small hardware that's not cost effective for a vendor to use but that you can justify by spreading its cost over several generations of electronics.

On the other hand, if you're trying to home-brew a digital content creation workstation, you're likely to be out in no-man's-land searching for device drivers for your video card. The top 3-D cards still have full support only with proprietary drivers, so don't expect to treat a high-end 3-D system like a Linux box. Where the low levels of the system are concerned, your workstation might as well be a proprietary UNIX system, because you can't expect community support when some drivers are closed off from your view and the view of the experts on the linux-kernel mailing list. For now, get anything requiring high-performance 3-D from a vendor that has a good working relationship with the video card manufacturer and whose support you trust.

A good middle ground between buying and building is to work with a friendly Linux system vendor that lets you customize the machines you order. It doesn't cost any more to talk to someone on the phone than to use the Web configurator, so it's a good idea to rough out the system you like, place a call and get some feedback.

You might choose to go with a small, friendly vendor for your systems at work and then build your own home machines. One advantage of getting systems from a friendly Linux vendor is burn-in. I suspect the engineers at Linux vendors have unresolved anger issues toward hardware—or maybe they want to cut back on the number of returned systems. Pogo uses a battery of burn-in tests based on the Cerberus Test Control System, which traces its heritage back to the original VA Research.

Enough introduction, it's time for the parts list.

If you buy no other hardware this year...

...get a pre-ban HDTV card. In a major setback for those who choose to build their own entertainment devices, the US Federal Communications Commission has approved the so-called Broadcast Flag regulation for high-definition television (HDTV). That's bad news for Linux boxes, Ultimate and otherwise. Future HDTV-capable tuner cards will be required to enforce a to-be-

determined digital rights management (DRM) regime. This is one product category that won't get better next year; it'll be worse because of mandatory DRM. If you live in the US, if you buy no other PC hardware before the end of 2004, pick up a pcHDTV card.

The pcHDTV HD-2000 Hi Definition Television Card works with the open-source player Xine and it will be illegal to sell next year. You'll still be able to use the card you stashed this year, though. We give you fair warning right now that in 2005 *Linux Journal* will cover projects that require pre-ban cards. Buy now or kick yourself next year.

Ultimate Linux Box 2004 Hardware

- Motherboard/chassis: Celestica A8440 (AMD-8131 Chipset)

  High-end motherboards are going to onboard Gigabit Ethernet. As with other server-oriented hardware, all the commonly used chipsets have good Linux support.
- Memory: 16x PC2700 2048MB ECC REG (32GB)
- Network interfaces: BCM5704 10/100/1000 x 2
- RAID: Adaptec ASR2200S
- Storage: Seagate ST336607LC 36GB U320 SCSI HDD x 4

  These aren't the fastest Seagate drives available, but with 32GB of memory, if we touch them we're either doing something wrong or running a benchmark. It's an easy upgrade to 15,000 RPM drives.
- Video: Appian Rushmore Quad-DVI PCI

  Appian's Rushmore card offers four displays at up to 2048×1536 resolution. With everything working correctly, that would be 25,165,824 pixels or 32 times the area of a conventional 1024 × 768 screen. At press time, we still were dealing with an interesting issue with XFree86 support for this card. Instead of four displays, we were getting two identical copies of two displays. Check out our Web site for the resolution to the X issue.
- Audio: Creative Labs SB Audigy
- Power Supply: 500W Hotswap x 3
- Miscellaneous: PC Floppy drive, IDE DVD-ROM, USB

Conclusions

With Fedora Core release 1.92 (FC2 Test 3) installed, the Ultimate Linux Box put up good numbers on the benchmarks, as might be expected.

Yes, with this much RAM we took the opportunity to build a kernel in a tmpfs partition. 2.6.4 with all defaults set completed in 1 minute 41 seconds. More detailed benchmark results follow.

Dbench 100:

```
#/usr/sbin/dbench 100
Throughput 133.97 MB/sec 100 procs
```

Bonnie++:

```
#bonnie++ -s 65536
```

Openssl Speed:

```
#openssl -speed

                  sign    verify    sign/s verify/s
rsa  512 bits   0.0003s   0.0000s   3720.8  42628.2
rsa 1024 bits   0.0010s   0.0001s   1005.9  16850.9
rsa 2048 bits   0.0057s   0.0002s    174.5   5674.7
rsa 4096 bits   0.0375s   0.0006s     26.7   1691.6
                  sign    verify    sign/s verify/s
dsa  512 bits   0.0002s   0.0002s   5506.3   4731.3
dsa 1024 bits   0.0005s   0.0006s   2033.5   1695.7
dsa 2048 bits   0.0016s   0.0019s    641.4    520.0
```

Hdparm:

```
#hdparm -t /dev/sda

 Timing buffered disk reads:  170 MB in  3.02 seconds =  56.28 MB/sec
```


Acknowledgements

This year's Ultimate Linux Box team from Pogo except Don Marti, all are credited as authors of this article. Cosmo King did the hands-on integration, testing, troubleshooting and benchmarking. The text (and errors) was written by Don Marti.

Resources for this article: /article/7614.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Linux on Linksys Wi-Fi Routers

James Ewing

Issue #124, August 2004

Hacking this reliable, inexpensive platform can be your first step to a successful wireless project. Chain access points together to cover a wide area, crank up the power level, get more working space in Flash memory and more.

Wireless networking has become a mass-market technology, and the price of 802.11 or Wi-Fi gear has fallen to commodity levels. Several thousand competitors with virtually identical products now are vying for your Wi-Fi dollars. In this kind of competitive space it is natural for manufacturers to seek the lowest cost alternatives. Their choice? Linux, of course.

Linux has become the premium OS for inexpensive, feature-packed wireless networking. Linksys, one of the major wireless players, turned to Linux for its 802.11g next-generation Wi-Fi devices. When Cisco bought Linksys in early 2003, it inherited both the Linux devices and an emerging feud over the unreleased GPL source code. After several months of lobbying by open-source enthusiasts, Cisco relented and released the source.

The Linksys WRT54G product (Figure 1) is especially interesting due to its low price and internal hardware. The WRT54G contains a four-port Ethernet hub, an Ethernet WAN port, support for the new high-speed 54MB/s 802.11g wireless protocol and backward compatibility with older 802.11b devices.

Figure 1. For under $100 US, the Linksys WRT54G is a capable Linux platform with 16MB of RAM, a 125MHz processor and support for 802.11b and g.

But what the WRT54G lacks is what makes it interesting. Under the hood the unit sports a 125MHz MIPS processor with 16MB of RAM. This is more than enough horsepower to run some serious applications, so why not add some?

Setting Up the Development Environment

The latest source on the Linksys site is about 145MB in size and contains a complete toolchain for MIPS cross development (see “The Linksys WRT54G Source Tree” sidebar).

Follow the instructions for creating symlink and PATH additions in the README file in the WRT54G/src subdirectory. Then `cd` to the router subdirectory and run `make menuconfig`. Keep the standard options for your first build, and click through to create your configuration files. `cd` up one level to the WRT54G/src directory and type `make`. That's all there is to it. A file called code.bin is created in the WRT54G/image directory containing a compressed cramfs filesystem and a Linux 2.4.20 kernel.

Now comes the scary part—how do you get this new firmware on to your Linksys? There are two methods, by tftp or through the Web-based firmware upgrade interface. I suggest you use the Web upgrade for your first try.

Surf to your Linksys box—the default address is 192.168.1.1—and log in. Select Firmware Upgrade from the Administration menu and upload your code.bin

file. The router now restarts. Congratulations, you have just modded your Linksys box.

The Ping Hack

The existence of Linux on the WRT54G was discovered through a bug in the ping utility in the Diagnostics menu. Firmware versions prior to 1.42.2 allowed arbitrary code to be run from the ping window if surrounded by back-ticks. If you have a box with the older firmware, try typing `` `ls -l /` `` in the ping window's IP address field. *Voilà*—a listing of the root directory magically appears.

The ping hack allows curious folks to explore their boxes without modifying the source. But exploring by way of the ping window is slow and tedious. What we really need is a shell on the box.

By expanding the ping hack in the source code, a custom firmware image can be created with the full power of a Linux shell over the Web interface. See the on-line Resources section URLs on how to create the command shell.

But why stop there? The default firmware's cramfs filesystem leaves 200K of Flash memory free. There is room for many useful applications, such as telnet or Secure Shell or perhaps even a VPN client or server.

The wl Command

One useful command supplied by Linksys in binary form only is wl. The wl command contains several dozen internal commands that control wireless settings, including the popular power adjustment setting. Typing `wl` with no parameters produces a complete list of its capabilities.

The default power setting on the WRT54G is 28 milliwatts, and this setting cannot be changed externally. But by using the ping hack or a shell, you can change this with wl, using the txpwr subcommand and a number between 1 and 84 milliwatts. This number raises or lowers the default power setting until the next reboot.

Increasing the power setting or replacing the stock antennas may increase your radio output and violate local laws. If you replace the stock antennas and lower the power setting, your unit's range can be extended significantly while remaining within legal radio power limits.

The WRT54G supports two external antennas and automatically balances between them depending on which received the last active packet. When adding a more powerful external antenna, this is not the setup that you want.

You need to force the unit to choose the high power antenna every time. This is done with `wl txant` for receiving and `wl antdiv` for sending. A 0 parameter forces the left antenna coupling and a 1 forces the right, as you face the front panel.

Adding Secure Shell (SSH)

One enterprising individual ported the entire OpenSSH toolchain to the Linksys box. Unfortunately, the size of the OpenSSH binary means that many standard Linksys functions must be removed to make room. Plus, the resulting RAM requirements are at the limits of available memory. What is needed is an SSH server with a small memory footprint, and the Dropbear server fits the bill nicely. Matt Johnson designed the Dropbear SSH dæmon specifically to run in memory-constrained systems such as the Linksys.

The standard Linksys Linux implementation lacks many of the normal files needed for multiuser Linux systems. Two of these—passwd and groups in the /etc directory—are required by the vast majority of Linux applications. In order to run the Dropbear server, we need to add these files to the Flash build.

By creating a passwd file with a root entry and no password and a matching groups file, we can make Dropbear almost happy enough to run. These files are copied to the /etc directory of the Flash image and are read-only on the Linksys.

When running, Dropbear also needs to access a private key that is used for SSH handshaking and authentication, as well as a known_hosts file containing the public keys of approved client machines. Generating the private key with the dropbearkey program is a snap, but storing it on the Linksys is a bit trickier.

The WRT54G contains a hash map of key name and value pairs located in nonvolatile storage called nvram. The bundled nvram utility and API allows us to read and write to this memory area. The Dropbear private key and our public key ID from id_rsa.pub in our home .ssh directory are stored in nvram and copied to /var in the RAM disk on system start.

We compile Dropbear with support for key-file authorization and now have a secure way to log in to the Linksys. If you need password login, the Dropbear code can be patched to read the system password from nvram and to add the ability for password logins as well.

Increasing Flash Memory Compression

After adding such utilities as SSH and telnetd, you soon find your Linksys firmware image bumping the limits of the Flash storage space on the device.

What you need is a filesystem with better compression than cramfs offers, one that is compatible with the Linksys Linux kernel.

The default cramfs filesystem compresses data in 4K blocks, but compressing on 4K boundaries limits the compression ratios that can be achieved. If we could find a filesystem that compressed larger blocks of data but mapped correctly to the page size in the OS, we would be able to put far more data and applications in the firmware.

Phillip Lougher's squashfs filesystem compresses in 32K blocks and is compatible with the 2.4 and 2.6 kernels. If we could move the Linksys firmware from cramfs to squashfs, we might have enough room for a VPN client and server in the system.

The Linksys kernel is a customized 2.4.20 source tree modified by Broadcom. Broadcom is a leading 802.11g chip maker and is responsible for the CPU and radio chips in the WRT54G. The squashfs tar file contains patches for the 2.4.20 through 2.4.22 kernels. Unfortunately, none of these applies cleanly to the Broadcom kernel tree, so a bit of hand editing is necessary. The patch with the fewest errors is the 2.4.22 version, which misses only one hunk when applied. By reading the patch file and finding the missing hunk, you can patch the missing code manually. You also can find a WRT54G-specific squashfs patch on the Sveasoft Web site.

The Linksys WRT54G Source Tree

When you unpack the GPL source from Linksys, a directory structure is created below the main WRT54G subdirectory. Here is an explanation of the important parts.

The main tarball directory is /WRT54G. The main Makefile lives in /release/src. After unpacking the source, read the README file here for instructions on how to compile it.

All of the applications packaged with the Linksys unit are built from /release/src/router. If you want to add applications, do it here and modify the Makefile in this directory. This Linux kernel source tree has been modified by Broadcom, the manufacturer of the wireless chips and CPU in the WRT54G. Add your kernel modifications or patches here, /release/src/linux/linux.

You need to create a symlink from /opt to the brcm directory here, /tools/brcm. Two of the subdirectories under brcm must be added to your PATH. See the README file above for more information.

Patches and updated source code can be downloaded from Sveasoft. See Resources for more information.

The next step is to edit the Broadcom kernel startup code and add a check for squashfs. The do_mount.c file contains nearly identical code and can be used as a guide when patching the startup.c file in the arch/mips/brcm-boards/bcm947xx subdirectory.

After patching the kernel, the router Makefile must be patched to generate a squashfs image and the Linux kernel configuration must be set to include squashfs support.

This is well worth the effort, however. On recompile you should find some 500K free bytes, compared to the stock cramfs filesystem.

The Wireless Distribution System

The standard WRT54G is a wireless access point (AP). This means that it can talk to wireless clients but not to other wireless access points. The ability to link it to other access points using the Wireless Distribution System (WDS) or to act as a wireless client is available using the wl command.

The Wireless Distribution System is an IEEE specification that allows wireless access points to be chained together in a wide area network. Although there is some performance penalty for doing this, the end result is an extended wireless network with a much greater range than is available using single APs.

In order to link two APs together using WDS, their respective MAC addresses must be known. Log in to each box and run the command `wl wds [`*`Mac Addr`*`]`, using the MAC address of the opposite machine's wireless interface. A new device called wds0.2 then appears on each box and can be assigned an IP address. Once the IP addresses are assigned and routing is set up between the two boxes, you are able to ping one from the other.

Each WDS link results in data traffic doubling within the network. Because 802.11g is half duplex, this halves the network throughput. If the APs are operating at 54MB/s, this is not much of a performance hit if you keep the links to three or fewer.

Client Mode Bridging

A simpler form of bridging is to set up one box as a client and have it link to an access point. This is known as an Ethernet bridge, and several products exist specifically for this purpose.

Client mode must be selected in the Linux kernel build menu and compiled in the kernel. Once done, the kernel is built with a Broadcom binary-only module that includes support for both AP and client modes. The command `wl ap 0` sets the box to client mode, and `wl join [SSID]` links it to an access point. If you set routing in the client using the access point's IP address as the default gateway, the client automatically routes to the access point and your bridge becomes active. Multiple AP and client pairs can be set up as an alternative to the WDS method described above.

The Power of Open Source

Linux has worked its way into everything from supercomputers to embedded systems, including the Linksys. The move to Linux is the result of a highest performance vs. lowest cost equation in a highly competitive market. Many similar wireless routers, such as the Belkin F5D7230-4, the Buffalotech WBR-G54 and the ASUS WL-300g and WL-500g, all use Linux in their firmware, and the list expands daily. Unfortunately, none of these companies has complied with GPL requirements and released the source code. Legal issues aside, these products will lag far behind the Linksys open-source products in capabilities and features for some time to come.

Linksys firmware builds containing amazing new features and capabilities appear daily. At the time of this writing, firmware builds for the Linksys WRT54G with support for VPNs, power adjustment, antenna select, client and WDS mode, bandwidth management and a whole lot more are available from multiple sources. The Internet combined with open-source code can change a small SOHO wireless router into a powerful multifunctional device.

One word of caution: using experimental firmware could kill your box and probably violates the Linksys warranty. If you are a casual user and need home or small office access to a wireless network, this definitely is not for you. Use the official Linksys firmware builds instead.

If, however, you are willing to risk your box and experiment with its potential, you may find it is capable of much more than the specifications listed on the product packaging—thanks to the power of Linux and open-source development.

Resources for this article: /article/7609.

James Ewing (james.ewing@sveasoft.com) has been an entrepreneur and software developer for more than 20 years. Originally from California, he moved to Sweden a decade ago and now balances his time between a wife and two children and practicing his authentic rendition of the Swedish chef on the Muppet show.

2004 Editors' Choice Awards

Linux Journal Staff

Issue #124, August 2004

We're excited about some great new Linux hardware and software, but we're still depending on some old favorites too.

It's getting harder and harder to keep up with all the great Linux-related products, services and projects out there. Fortunately, we've expanded our list of contributing editors over the past year, and the panel for Editors' Choice is looking pretty distinguished. So, without further ado, here's Editors' Choice Awards 2004.

Server Hardware: HP ProLiant BL20p G2

The HP ProLiant BL20p G2, which Ibrahim Haddad recommends, features two Intel Xeon processors, onboard RAID, two hot-swap SCSI drives, three Gigabit Ethernet interfaces, plus one more Ethernet connection for management, and optional Fibre Channel. That would be nice in a 1U rackmount server, but this box is a blade, and you can pack eight of them, plus up to six redundant power supplies and your choice of two switches or other interconnect options, in a 6U chassis.

If dinky laptop drives have been your reason not to drink the blade-ade, look again at the new generation of heavyweight blade servers. Maybe it's time to save the pizza boxes for pizza.

Personal Computer or Workstation: IBM ThinkPad T41

Because each editor has different, strongly held opinions about his or her personal work environment, we all were surprised when Doc Searls, Ibrahim Haddad and Robert Love agreed on this: the IBM ThinkPad T41 is the Linux laptop to have. They didn't simply agree on ThinkPad or ThinkPad T series—they all use and like one particular model.

Doc praised the T41's “Industrial-strength looks and race-car feel”, and he loves the high performance. “Everything works in Linux”, Robert commented. What happened to the good old days, when we waited for kernel hackers to buy the unsupported laptops first and get them going for the rest of us? The T41 has a 1400×1050 screen and IBM's famous three-year warranty and fast, competent repair service.

Any hardware whose speed gets compared to greased rodents is at least worthy of an honorable mention, and Greg Kroah-Hartman made that comparison in his vote for the dual-processor version of the Apple Power Mac G5, which is one Linux install away from being a great system. “It's fast, quiet and pretty to look at. With full 64-bit goodness for a very cheap price, what's not to like?” he wrote.

Security Tool: Clam AntiVirus (AV)

Reuven Lerner writes, “ClamAV is giving the commercial virus-checking programs a real run for their money. The combination of ClamAV and SpamAssassin has reduced dramatically the amount of annoying (and potentially dangerous) mail sent through my server.”

With this year's outbreak of e-mail worms for non-Linux platforms, ClamAV has been getting quite a workout, and Linux admins on mailing lists report that database update times are keeping up with or beating the proprietary alternatives. And, yes, commercial support now is available.

Web Browser or Client: Mozilla Firefox

“I am beginning to think that Mozilla is the new Emacs—a cross-platform program that is solid and extensible”, Reuven writes. See the July 2004 issue for a tutorial and sample code to get you started on developing Mozilla-based apps, and see your nearest Linux desktop for a pop-up-free, standards-compliant browsing experience.

Graphics Software: The GIMP

The GIMP Project has released its eagerly anticipated version 2.0 and regained its top spot as our editors' favorite graphics tool. Marcel Gagné writes, “With the addition of EXIF handling, CMYK support and a cleaner, better interface, The GIMP remains unchallenged on my Linux desktop.”

Communication Tool: mutt

Although instant messaging apps and GUI mailers get all the demo time at Linux events, the text-based mailer mutt, which lets you configure practically

anything, remains a cult classic. Greg writes, “without it there is no way I could get through an e-mail feed of over 500 messages a day.”

Don Marti uses Ximian Evolution for its calendar and contact list but sticks with mutt for mail. Use mutt together with Mozilla for convenient attachment viewing, or for a healthy dose of mind-blowing tweaked-out config files, try a Web search for “my .muttrc”.

Desktop Software: GnuCash

“I began to use GnuCash several months ago and was very impressed by its features”, Reuven writes. “It has an impressive array of features and can be programmed using Guile. If you've never managed your finances before or are shaky on the idea of double-entry bookkeeping, the built-in tutorial will help you get started.” A financial tool without double entry is like a paint program without layers.

Software Library or Module: Pango

This is a new category, but it's about time we recognized library maintainers. Library code saves time and prevents errors by letting people “outsource” parts of apps. We're always happy to see developers use a good library instead of reimplementing something from scratch. Reuven writes, “I want to thank all of the hardworking people who have worked on Pango and the other internationalization libraries and software that make non-Western scripts usable with Linux. Thanks to you, billions of people who don't speak, read or write English still can use open-source software. The fact that I can read and write Hebrew e-mail with the standard version of Mozilla or documents with the standard version of OpenOffice.org continues to impress me.”

Development Tool: BitKeeper

Greg writes, “It makes my life dealing with zillions of kernel patches sane. It is the only way I successfully can maintain seven different kernel trees and still have time to sleep.”

Linus Torvalds contributed a stunner of a quote to a BitKeeper company press release—“It's made me more than twice as productive”, he said. As if he was slow before. With that kind of testimonial, BitKeeper deserves a slot in any company's search for a new source code management system.

Database: PostgreSQL

“I continue to love PostgreSQL and prefer it over MySQL because of its features, stability, scalability, Unicode compatibility and adherence to standards”, Reuven

writes. “That said, the MySQL team is making impressive inroads, and I expect to see them close the gap with PostgreSQL in the coming years. But for now, I strongly recommend PostgreSQL to anyone who needs a relational database.”

Marcel concurs. “PostgreSQL is still number one for me”, he writes. “This is a grown-up, powerful database, and the first I turn to when I need to create or use database-enabled applications.”

Mobile Device: Sharp Zaurus SL-6000 PDA

The latest Zaurus is Ibrahim Haddad's choice. Unlike previous Zauri, this one features USB host support, so you can use it with your USB devices for storage, networking and input. The screen is a pixel-licious 480×640, four times the area of the original Zaurus and the same as the Japan-only Zaurus SL-C700 we reviewed last year.

Game: Really Simple Syndication (RSS)

Our editors are all business and turned up their noses at selecting favorite games. These are the kind of people you want to hire to roll out your company desktop systems. But even though it might not look like *Quake* or *Frozen Bubble* when the boss walks by, there's a new hit game that Linux people are playing on the Net, and whether you want to call it blogging or social software, players are everywhere. It's like painting *Dungeons and Dragons* figures or collecting baseball cards, but with real people.

The glue tying it all together is a simple XML-based syndication format called RSS, which sites such as Technorati and software projects such as Planet are using to bring together Web content in new ways. Who's a blog king and who's a bozo? Pop in to Technorati to check the score.

Reuven points out that the all-in-one social network sites LinkedIn, Orkut and Ryze aren't particularly useful, but he says they're “all scratching the surface of something new and interesting.” It gets really interesting when social networking info crosses site boundaries and anyone can crawl it. Game on!

Technical Book (tie): *Real-World XML* and *Hacking the Xbox*

Paul Barry called Andrew “bunnie” Huang's *Hacking the Xbox* “a darned good read” in our January 2004 issue. The book is a matter-of-fact introduction to current issues in making hardware do what you want and not what fits into some company's business model.

Reuven writes that *Real-World XML* by Steven Holzner is “another big, thick book about XML, which doesn't really need big, thick books. But it offers some good explanations, sample code and discusses applications, including SOAP.”

If you're into well formed documents, get Huang's book; if you're into well formed solder joints, get Holzner's. Expand your mind.

Nontechnical Book: *Free Culture*

Did some of the members of Beatallica want to be a Beatles tribute band, while others wanted to be a Metallica tribute band? We can't go see them perform “Got to Get You Trapped Under Ice” and “Everybody's Got a Ticket to Ride Except for Me and My Lightning”, because Beatallica is in hiding for fear of record company lawyers.

It wasn't always like this. Back when Walt Disney directed *Steamboat Willy*, a parody of Buster Keaton's *Steamboat Bill, Jr.*, copyright law was different and encouraged creativity, not lawyer bills. Professor Lawrence Lessig, in *Free Culture*, explains copyright in a way that will help you, the Linux and Internet native, explain today's copyright issues to people who are new to the whole sorry scene. Lessig represents the often-ignored middle ground in the copyright debate.

Technical Web Site: LWN

LWN wins again. We can say the same thing about this site that we said last year: a great mix of links to the best Linux stories from other sites, including *Linux Journal*'s, plus original technical content. A recent series profiles the various free software choices in calendars, image viewers and drawing programs.

Nontechnical or Community Web Site: Groklaw

If you sold your TV when *L.A. Law* went off the air, this is the site for you. Get sucked in to the courtroom drama surrounding failing UNIX vendor The SCO Group, formerly Caldera, and the company's long-shot lawsuits against AutoZone, Daimler-Chrysler, IBM and Novell. Will SCO dodge a lawsuit from Red Hat? Did Novell transfer UNIX copyrights to SCO? Will Grace get back together with Victor? Greg says Groklaw is “now the home page for more IBM executives than any other site.”

Mailing List or Other Support Forum: linux-kernel List

Greg weighs in to support the linux-kernel mailing list: “It's high volume, oftentimes rude, but always informative and never boring. And if a user is willing to be nice, quite helpful”, he says. So be nice. Or else.

Project of the Year: Ardour

The digital audio workstation Ardour was the centerpiece of the Linux-based recording studio in Aaron Trumm's article in the May 2004 issue. In his column for the *Linux Journal* Web site, Dave Phillips wrote, “Ardour has become a center of attention for those of us who wish to use Linux in a professional audio setting”, and “That Ardour has come so far and evolved so well is a testament to the talents and dedication of its programming crew.” Congratulations to Paul Davis and the rest of the Ardour team.

Product of the Year: EmperorLinux Toucan

Remember that IBM ThinkPad T41, the laptop everyone likes? Doc bought his through EmperorLinux, a company full of friendly people who set up major-brand laptops with your distribution of choice, with a patched and tested kernel that supports the laptop hardware. Emperor sells its Linux-enabled T41 as the “Toucan”, and it will set up the system with any of six different distributions or dual-boot with a Microsoft OS. Best of all, EmperorLinux is quick to reply to support calls on Linux issues and the original manufacturer's warranty remains in effect for the hardware.

Now that the T41 is a hit on the Linux scene, will IBM sell EmperorLinux an OS-less version so Linux customers don't have to pay for a legacy OS license? Maybe if they knock off reading Groklaw for a few minutes and do the deal, we'll get lucky next year.

Resources for this article: /article/7613.

Archive Index Issue Table of Contents

Advanced search

Linux Serial Consoles for Servers and Clusters

Matthew E. Hoskins

Issue #124, August 2004

The more Linux servers you're responsible for, the more that serial consoles can save you money, space and headaches by easing remote administration duties.

Managing large numbers of Linux and UNIX systems takes a lot of organization, automation and careful use of technology. A significant chunk of one's time as a system administrator is spent building infrastructure to make managing those systems easier. Doing so improves flexibility, recoverability and reduces downtime. All of this hopefully results in less stress and longer vacations. This article discusses one of those simple technologies that helps accomplish all of the above, serial consoles.

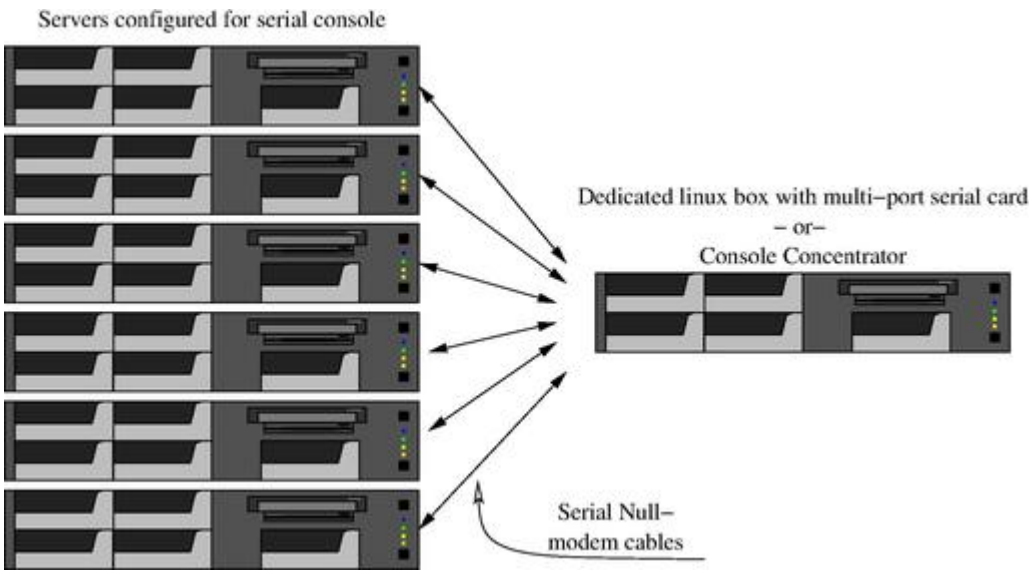


Figure 1. Managing Many Servers from a Console Server

Serial consoles always have been a standard feature of enterprise UNIX hardware. Modern high-density server and cluster configurations sometimes can squeeze more than 50 servers in a 19" rack, so having monitors and keyboards for each one is unthinkable. Although KVM (keyboard/video/mouse)

switches can connect many servers to a small number of monitors and keyboards, they are expensive and even more so with remote access features. Serial consoles allow you to take racks or shelves of servers and have all their consoles available all the time, from anywhere.

Consoles Defined

The console is a simple I/O device, initialized early in the kernel boot process to convey informational messages as the system comes up. Once the operating system starts running startup scripts, the console can be used to recover an ailing system or to get input from the system administrator interactively, like Red Hat's Kudzu does. One compelling feature of serial consoles is never having to drive in to work because a system is hung up at reboot asking for input for fsck. After the system is up completely, the console usually becomes a login terminal, sometimes a graphical one. The console also can be used as a last resort method of reporting problems inside the kernel. Under these panic conditions it is not possible always to write to log files or network log servers, so messages are reported to the console. For this reason and many more, consoles on servers should be a simple device, and a serial port is the simplest device included on standard system configurations. For those last-minute panic messages, one could add a console device that supports buffering and logging, so you never miss a moment of the excitement.

Hardware Support

What we are talking about here is booting up a system without using an attached keyboard, mouse or video monitor. Some motherboards may complain without a directly attached keyboard, but this requirement usually can be changed in the BIOS configuration. In fact, with the recent popularity of USB keyboards, most BIOS versions do not care about missing keyboards. If you are using a system that was designed to be a server, you may be even more fortunate. Several vendors have started adding extra functionality to their BIOS versions to better support serial consoles from power-on. These features sometimes include power-on system test (POST) output and BIOS configuration access over the serial port. Depending on your needs, you can select your hardware accordingly by checking the vendor specifications. Even without BIOS support, you still can use serial consoles quite effectively on almost any PC system.

This is not a perfect solution, though, and your average PC hardware does not provide all of the features available in typical enterprise-class UNIX hardware. PC BIOS versions do not have the concept of a boot monitor (see the “What Is a Boot Monitor?” sidebar), nor can you perform a hardware halt of the OS as you can in enterprise UNIX hardware. For many applications this is okay, but when

more functionality is needed add-on hardware options are available, and I discuss them later.

What Is a Boot Monitor?

Classic UNIX hardware (Sun, HP, SGI and so on) usually has a feature called a boot monitor. Think of it as a tiny operating system built into Flash memory on the motherboard of a server or workstation. Sometimes called a boot console or prom console, they function as a PC BIOS and a bootloader in one. They are responsible for understanding all manner of boot devices and getting the kernel image into RAM and running. Most boot monitors can boot from the network for diskless operation or recovery from a failed boot device. One key feature of boot monitors is they stand between the console and the kernel, so it often is possible to suspend or halt the running OS and drop to the boot monitor with a magic keystroke. Then, even better, the OS resumes where it left off. This allows you to diagnose hardware problems or forcibly reboot the system even if the kernel has died. In a PC system, only a skeletal set of hardware support exists in the BIOS, and the rest must be provided by small chunks of code provided in Flash memory on the hardware itself.

Most PC hardware BIOS versions can be configured only with a directly attached keyboard and video monitor. Luckily most come with usable default settings, so this is not normally an issue. If it is, you may need to have the system initially configured with direct attached video and keyboard and then switch to serial console. In my experience, I have rarely needed to do this; it needs to be done only once during initial hardware setup.

Software Configuration Overview

As packaged by most distributions, the Linux kernel and bootloader select the directly attached video controller and keyboard as console, but this is easily changed. When a PC-based system boots, the bootloader is the first program to be loaded off the disk. The three major bootloaders in popular use on Linux systems are GRUB, LILO and SYSLINUX (used on boot floppies); all of them support serial consoles. Next, the Linux kernel needs to be told to use a serial port for its console, which can be handled at compile time or by passing kernel command-line options from the bootloader configuration. Finally, if you want to be able to log in on the console, you need to configure a getty process to run after the system is up.

Kernel Configuration

We discuss the kernel configuration next because it is a prerequisite to understanding the bootloader config later on. The most flexible way to configure the kernel console is with the options passed on the kernel command

line. You can append arguments to the command line from the bootloader. Here is an example of the kernel command-line syntax:

```
console=ttyS0,9600n8
```

This tells the kernel to use ttyS0 (the first serial port discovered by the kernel), running at 9,600 baud, no parity and 8 bits. The kernel defaults to one stop bit. This is the most common speed and configuration for a serial console, which is why most serial ports and terminals default to 9600n8. It is possible to append more than one `console=` argument to the command line; kernel messages then are output to all of them, but only the last one is used for input.

Bootloader Configuration: GRUB

GRUB is a flexible bootloader with excellent support for serial consoles. When properly configured, GRUB allows multiple devices to be used as a console. Listing 1 shows an example grub.conf file (usually /boot/grub/grub.conf and symlinked to /etc/grub.conf) as configured by the Red Hat/Fedora Core installer. Yours may be slightly different.

Listing 1. An Ordinary grub.conf File

```
# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub
# after making changes to this file
# NOTICE:You have a /boot partition.
#        This means that all kernel
#        and initrd paths are relative
#        to /boot/, eg.
#        root (hd0,1)
#        kernel /vmlinuz-version ro root=/dev/hda6
#        initrd /initrd-version.img
#boot=/dev/hda
default=0
timeout=10
splashimage=(hd0,1)/grub/splash.xpm.gz
title Red Hat Linux (2.4.20-8)
        root (hd0,1)
        kernel /vmlinuz-2.4.20-8 ro root=LABEL=/
        initrd /initrd-2.4.20-8.img
```

The first thing to do is remove all splashimage directives. In some early versions, these directives confuse GRUB and make it default to the video console. Then add a serial and terminal line. The serial line initializes the serial port to the proper baud and settings. In the terminal line, we configure GRUB to send prompts to both the serial port and to the keyboard and monitor. You can press any key on either, and it becomes the default console. The `--timeout=10` argument tells GRUB to default to the first device listed in the terminal line after ten seconds. We also modified the kernel command line to include the option that tells the Linux kernel to use the serial port as console. Listing 2 shows the complete modified grub.conf file.

Listing 2. A grub.conf File That Supports Serial Console

```
#boot=/dev/hda

# Options added for serial console
serial  --unit=0 --speed=9600 \
        --word=8 --parity=no --stop=1
terminal --timeout=10 serial console

default=0
timeout=10
title Red Hat Linux (2.4.20-8)
        root (hd0,1)
        kernel /vmlinuz-2.4.20-8 ro \
                root=LABEL=/ console=ttyS0,9600n8
        initrd /initrd-2.4.20-8.img
```

Bootloader Configuration: LILO

The LILO bootloader, although much more mature than GRUB, is less feature-rich. We must configure LILO and pass options to the kernel to use a serial port. To do this, we add:

```
serial=<port>,<bps><parity><bits>
```

where port 0 is the first serial port detected by LILO. Also, the `append=` line is modified to include the kernel options. After modifying the /etc/lilo.conf file, be sure to run LILO to update the bootloader. The completed lilo.conf file is shown in Listing 3.

Listing 3. lilo.conf with Serial Console Support

```
serial=0,9600n8
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=50
message=/boot/message
linear
default=Linux

image=/boot/vmlinuz-2.4.20-8
        label=2.4.20-8
        read-only
        initrd=/boot/initrd-2.4.20-8.img
        append="root=LABEL=/ console=ttyS0,9600n8"
```

Bootloader Configuration: SYSLINUX

SYSLINUX is a bootloader designed for use with DOS/FAT formatted bootable floppies. Red Hat/Fedora Core Linux uses SYSLINUX for both the install boot and rescue floppies. In order to install or recover from boot floppies over serial console, the floppies need to be modified. We have added the `console=` and `text` directive to the append line, and we have removed the extra boot

selections present in Red Hat's original file. The first line initializes and directs SYSLINUX to use serial port 0 (aka /dev/ttyS0) and defaults to 9600n8. Using this modified boot floppy, we can install the OS over the serial console. Red Hat's text installation option works quite nicely this way. Using the above modifications, you can convert any SYSLINUX boot floppy to use serial consoles. This procedure also works for ISOLINUX, which is a spinoff of SYSLINUX used on bootable CD-ROMs.

Listing 4. syslinux.cfg File Configured for Serial Console

```
serial 0
default Linux
prompt 1
timeout 100
label Linux
  kernel vmlinuz
  append initrd=initrd.img lang= text   \
        devfs=nomount ramdisk_size=8192 \
        console=ttyS0,9600n8
```

Enabling Logins and Tuning

As stated before, the console can become a login terminal after the system is up. For this to happen, the getty entries in /etc/inittab must be modified. The standard /etc/inittab starts mingetty on virtual consoles only. Because mingetty is not suitable for serial terminals, we must use something else. Many getty-type programs are available, but agetty is included with almost every Linux distribution, so we use it. Also, make sure the system boots to nongraphical mode, normally runlevel 3. Some Linux distributions default to an X login, usually at runlevel 5, if any X packages were installed. The default runlevel is determined on the initdefault line. To enable agetty on serial lines, you can modify the initdefault line in /etc/inittab:

```
id:3:initdefault:
```

and add a line for agetty:

```
co:2345:respawn:/sbin/agetty ttyS0 9600 vt100
```

This tells agetty to start waiting for logins on /dev/ttyS0 at 9,600bps, using vt100 terminal emulation. You may want to keep the original mingetty entries to allow a directly attached keyboard and monitor to be used for logins. If not, simply comment them out. Where root can log in from is controlled strictly; in order for root to log in from ttyS0, you must add the device to the /etc/securetty file.

Finally, if your system has created a /etc/ioctl.save file, delete or rename it. This file is used to save console settings between reboots. If the system was booted

using a directly attached keyboard and monitor, this file attempts to restore improper settings. A new one is created when you reboot using the serial console.

Tweaking for Red Hat/Fedora Core

Red Hat's bootup scripts use escape sequences, so the OK, PASS and FAIL messages show up in color. This can confuse serial consoles, so it is best to disable it. Simply modify /etc/sysconfig/init, and change the `BOOTUP=` line to say `BOOTUP=serial`. This will prevent the use of color messages.

Cabling

Serial cabling can cause some confusion. Basically, there are two kinds of serial ports, DCE (Data Communication Equipment) and DTE (Data Terminal Equipment). The ports differ in how specific signals are connected to pins on the connector. Data communication with serial ports uses separate transmit and receive wires, so when connecting two pieces of equipment together, one must make sure the transmit wire on one side connects to the receive wire on the other side. As long as you are connecting a DCE device to a DTE device you can use a regular straight-through cable, where each pin is connected to the same pin on the other side of the cable. If you are connecting devices of the same type, however, you must use a special cable or adapter, called a null modem, so the signals are swapped properly. DTE devices usually are terminals, computers and printers. DCE devices are designed to connect directly to computers, such as modems and serial mice.

In addition to the data transmit and receive wires, a number of handshaking signals are used to control the flow of data, so one side is not talking too fast for the other to understand. These signals also must be swapped by the null modem. To add to the confusion, two popular connectors are in use for serial ports, the 9-pin DB9 and the 25-pin DB25. These can come in both male and female varieties. In almost every case, the devices used for serial consoles (terminals, computers and console servers) are all DTE, which means you need a null modem of some sort. These are available in the form of adapters and cables. Most off-the-shelf units work fine, but if you want to solder your own, check the on-line Resources section for links to pinouts and cable diagrams.

Putting It All Together

At this point, we have described a Linux system that can boot up without a directly attached keyboard and monitor. It uses the first serial port for all informational messages as the system boots and accepts logins from that console once the system is up. But to what should you connect that console port? There is a world of possibilities. If you have no particular need for remote-

console access, you simply can leave the port unconnected until you need to maintain the system. You can use a computer or laptop connected over a null modem with the minicom program to access your system's console. Simply configure minicom to speak to an unused serial port, set the speed to 9,600 baud, 8 bits, no parity and 1-stop bit (aka 9600-8n1). Cable the systems together, then watch the system boot and eventually ask you to log in.

For remote access to a server's console, you can set up a console concentrator, which is a lot like a terminal server. It can be a homegrown Linux box with multiport serial cards, giving you as many ports as you have servers. With this kind of setup, you can access all your servers' consoles by logging in to a single dedicated Linux box.

Specialized Hardware

If you like the idea of remote access to your consoles but want more of an appliance, a number of products can help. Cyclades (www.cyclades.com) makes a console concentrator called AlterPath; it is reasonably priced and comes in 1, 4, 8, 16, 32 and 48-port models. The AlterPath units run Linux internally from Flash memory. A Web interface is used for configuration, or you can modify the configuration files directly through a shell login.

The most flexible way to configure the Cyclades unit is to present the consoles using Cyclades' modified SSH dæmon. This way you can SSH directly to each connected server's console port, which is identified by a textual name you choose. So, to connect to a server identified as server hooked to a Cyclades unit with a hostname of cyclades as the user matt, the command would look like: `ssh matt:server@cyclades`. (The colon syntax is a Cyclades modification to sshd, allowing you to pass a port name.) This setup is easy to use, and you even can set up SSH private key authentication.

Other vendors make console concentrators or servers, including Digi (www.digi.com), Equinox (www.equinox.com) and Raritan (www.raritan.com). All of these vendors offer network-attached serial console products.

As mentioned earlier, serial consoles on standard PC hardware lack some of the features available on enterprise UNIX hardware. One solution is PC Weasel (www.realweasel.com), which comes in the form of a PCI or ISA card. This device emulates a video card and translates all output to the serial port as normal terminal escape sequences. Input from the serial port is translated into PC keyboard scan codes. Because it looks like a video card to the system, the system allows it full access to BIOS and POST. Additional features allow you to do a remote hard reset. The PC Weasel also has its own processor, so it is available even if the host into which it is plugged crashes.

Specialized Software

If you would like to build your own console concentrator, some options are available to make it a little better than a simple box with a lot of serial ports. Conserver (www.conserver.com) is an open-source software package for managing systems connected to serial consoles. It supports SSL encryption and is highly configurable.

Resources for this article: /article/7507.

Matthew E. Hoskins is a Linux/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 (also known as MattGlue). He can be reached at matt@njit.edu.

Archive Index Issue Table of Contents

Advanced search

Distributed Caching with Memcached

Brad Fitzpatrick

Issue #124, August 2004

Cut the load on your Web site's database by adding a scalable object caching layer to your application.

Memcached is a high-performance, distributed caching system. Although application-neutral, it's most commonly used to speed up dynamic Web applications by alleviating database load. Memcached is used on LiveJournal, Slashdot, Wikipedia and other high-traffic sites.

Motivation

For the past eight years I've been creating large, interactive, database-backed Web sites spanning multiple servers. Approximately 70 machines currently run LiveJournal.com, a blogging and social networking system with 2.5 million accounts. In addition to the typical blogging and friend/interest/profile declaration features, LiveJournal also sports forums, polls, a per-user news aggregator, audio posts by phone and other features useful for bringing people together.

Optimizing the speed of dynamic Web sites is always a challenge, and LiveJournal is no different. The task is made all the more challenging, because nearly any content item in the system can have an associated security level and be aggregated into many different views. From prior projects with dynamic, context-aware content, I knew from the beginning of LiveJournal's development that pregenerating static pages wasn't a viable optimization technique. It's impossible due to the constituent objects' cacheability and lifetimes being so different, so you make a bunch of sacrifices and waste a lot of time precomputing pages more often than they're requested.

This isn't to say caching is a bad thing. On the contrary, one of the core factors of a computer's performance is the speed, size and depth of its memory hierarchy. Caching definitely is necessary, but only if you do it on the right

medium and at the right granularity. I find it best to cache each object on a page separately, rather than caching the entire page as a whole. That way you don't end up wasting space by redundantly caching objects and template elements that appear on more than one page.

In the end, though, it's all a series of trade-offs. Because processors keep getting faster, I find it preferable to burn CPU cycles rather than wait for disks. Modern disks keeping growing larger and cheaper, but they aren't getting much faster. Considering how slow and crash-prone they are, I try to avoid disks as much as possible. LiveJournal's Web nodes are all diskless, Netbooting off a common yet redundant NFS root image. Not only is this cheaper, but it requires significantly less maintenance.

Of course, disks are necessary for our database servers, but there we stick to fast disks with fast RAID setups. We actually have ten different database clusters, each with two or more machines. Nine of the clusters are user clusters, containing data specific to the users partitioned among them. One is our global cluster with non-user data and the table that maps users to their user clusters. The rationale for independent clusters is to spread writes. The alternative is having one big cluster with hundreds of slaves. The difficulty with such a monolithic cluster is it only spreads reads. The problem of diminishing returns appears as each new slave is added and increasingly is consumed by the writes necessary to stay up to date.

At this point you can see LiveJournal's back-end philosophy:

1. Avoid disks: they're a pain. When necessary, use only fast, redundant I/O systems.
2. Scale out, not up: many little machines, not big machines.

My definition of a little machine is more about re-usability than cost. I want a machine I can keep using as long as it's worth its space and heat output. I don't want to scale by throwing out machines every six months, replacing them with bigger machines.

Where to Cache?

Prior to Memcached, our Web nodes unconditionally hit our databases. This worked, but it wasn't as optimal as it could've been. I realized that even with 4G or 8G of memory, our database server caches were limited, both in raw memory size and by the address space available to our database server processes running on 32-bit machines. Yes, I could've replaced all our databases with 64-bit machines with much more memory, but recall that I'm stubborn and frugal.

I wanted to cache more on our Web nodes. Unfortunately, because we're using mod_perl 1.x, caching is a pain. Each process and thus, each Web request, is in its own address space and can't share data with the others. Each of the 30–50 processes could cache on its own, but doing so would be wasteful.

System V shared memory has too many weird limitations and isn't portable. It also works only on a single machine, not across 40+ Web nodes. These issues reflect what I saw as the main problem with most caching solutions. Even if our application platform was multithreaded with data easily shared between processes, we still could cache on only a single machine. I didn't want all 40+ machines caching independently and duplicating information.

Memcached Is Born

One day, sick of how painful it is to cache efficiently in mod_perl applications, I started dreaming. I realized we had a lot of spare memory available around the network, and I wanted to use it somehow. If you're a Perl programmer strolling through CPAN, you find an abundance of Cache::* modules. The interface to almost all of them is a dictionary. If you're fortunate enough to have missed Computer Science 101, a dictionary is the name of the abstract data type that maps keys to values. Perl people call that an associative array or a hash, short for hash table. A hash table is a specific type of data structure that provides a dictionary interface.

I wanted a global hash table that all Web processes on all machines could access simultaneously, instantly seeing one another's changes. I'd use that for my cache. And because memory is cheap, networks are fast and I don't trust servers to stay alive, I wanted it spread out over all our machines. I did a quick search, found nothing and started building it.

Each Memcached server instance listens on a user-defined IP and port. The basic idea is you run Memcached instances all over your network, wherever you have free memory and your application uses them all. It's even useful to run multiple instances on the same machine, if that machine is 32-bit and has more total memory than the kernel makes available to a single process. For example, while we were learning our lesson on scaling out and not up, we picked up a ridiculously expensive machine that happens to have 12GB of memory. Nowadays, we use it for a number of miscellaneous tasks, one of which is running five 2GB Memcached instances. That gives us 10GB more memory in our global cache from a single machine, even though each process on 32-bit Linux usually can address only 3GB of memory.

The trick to Memcached is that for a given key, it needs to pick the same Memcached node consistently to handle that key, all while spreading out storage (keys) evenly across all nodes. It wouldn't work to store the key foo on

machine 1 and then later have another process try to load foo from machine 2. Fortunately, this isn't a hard problem to solve. We simply can think of all the Memcached nodes on the network as buckets in a hash table.

How Memcached Works

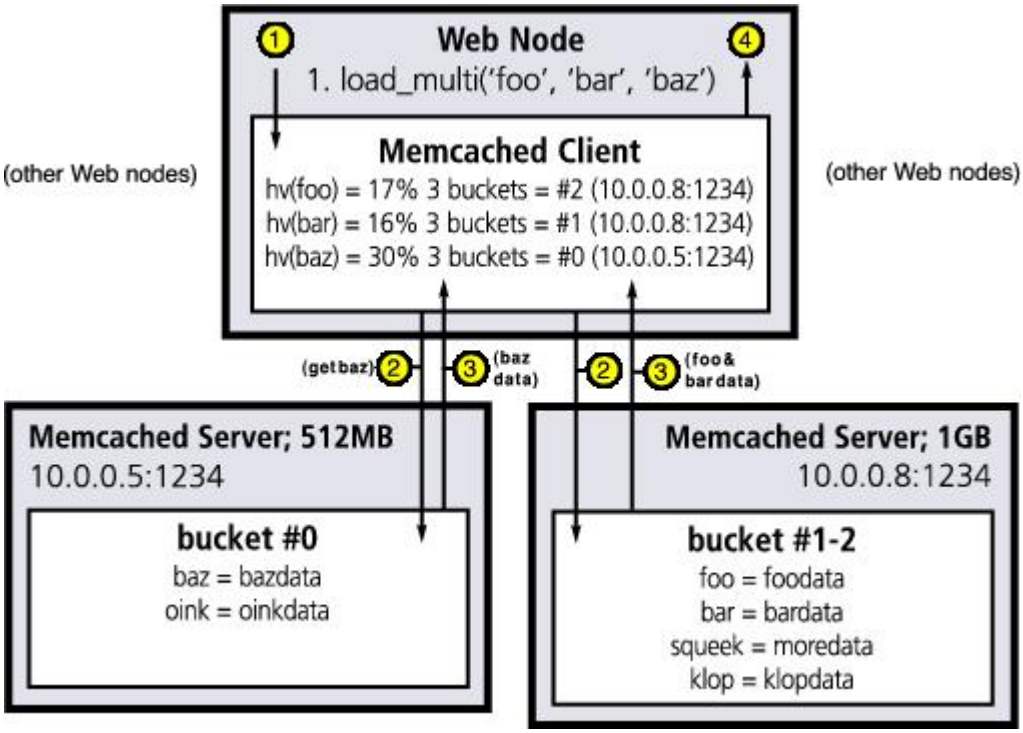


Figure 1. The Memcached client library is responsible for sending requests to the correct servers.

Step 1: the application requests keys foo, bar and baz using the client library, which calculates key hash values, determining which Memcached server should receive requests.

Step 2: the Memcached client sends parallel requests to all relevant Memcached servers.

Step 3: the Memcached servers send responses to the client library.

Step 4: the Memcached client library aggregates responses for the application.

If you know how a hash table works, skim along. If you're new to hashes, here's a quick overview. A hash table is implemented as an array of buckets. Each bucket (array element) contains a list of nodes, with each node containing [key, value]. This list later is searched to find the node containing the right key. Most hashes start small and dynamically resize over time as the lists of the buckets get too long.

A request to get/set a key with a value requires that the key be run through a hash function. A hash function is a one-way function mapping a key (be it numeric or string) to some number that is going to be the bucket number. Once the bucket number has been calculated, the list of nodes for that bucket is searched, looking for the node with the given key. If it's not found, a new one can be added to the list.

So how does this relate to Memcached? Memcached presents to the user a dictionary interface (key -> value), but it's implemented internally as a two-layer hash. The first layer is implemented in the client library; it decides which Memcached server to send the request to by hashing the key onto a list of virtual buckets, each one representing a Memcached server. Once there, the selected Memcached server uses a typical hash table.

Each Memcached instance is totally independent, and does not communicate with the others. Each instance drops items used least recently by default to make room for new items. The server provides many statistics you can use to find query/hit/miss rates for your entire Memcached farm. If a server fails, the clients can be configured to route around the dead machine or machines and use the remaining active servers. This behavior is optional, because the application must be prepared to deal with receiving possibly stale information from a flapping node. When off, requests for keys on a dead server simply result in a cache miss to the application. With a sufficiently large Memcached farm on enough unique hosts, a dead machine shouldn't have much impact on global hit rates.

Our Setup

LiveJournal.com currently has 28 Memcached instances running on our network on ten unique hosts, caching the most popular 30GB of data. Our hit rate is around 92%, which means we're hitting our databases a lot less often than before.

On our Web nodes with 4GB of memory, we run three Memcached instances of 1GB each, then mod_perl using 500MB, leaving 500MB of breathing room. Running Memcached on the same machine as mod_perl works well, because our mod_perl code is CPU-heavy, whereas Memcached hardly touches the CPU. Certainly, we could buy machines dedicated to Memcached, but we find it more economical to throw up Memcached instances wherever we happen to have extra memory and buy extra memory for any old machine that can take it.

You even can run a Memcached farm with all instances being different sizes. We run a mix of 512MB, 1GB and 2GB instances. You can specify the instances and their sizes in the client configuration, and the Memcached connection object weights appropriately.

Speed

Of course, the primary motivation for caching is speed, so Memcached is designed to be as fast as possible. The initial prototype of Memcached was written in Perl. Although I love Perl, the prototype was laughably slow and bloated. Perl trades off memory usage for everything, so a lot of precious memory was wasted, and Perl can't handle tons of network connections at once.

The current version is written in C as a single-process, single-threaded, asynchronous I/O, event-based dæmon. For portability and speed, we use libevent (see the on-line Resources section) for event notification. The advantage of libevent is that it picks the best available strategy for dealing with file descriptors at runtime. For example, it chooses kqueue on BSD and epoll on Linux 2.6, which are efficient when dealing with thousands of concurrent connections. On other systems, libevent falls back to the traditional poll and select methods.

Inside Memcached, all algorithms are O(1). That is, the runtime of the algorithms and CPU used never varies with the number of concurrent clients, at least when using kqueue or epoll, or with the size of the data or any other factor.

Of note, Memcached uses a slab allocator for memory allocation. Early versions of Memcached used the malloc from glibc and ended up falling on their faces after about a week, eating up a lot of CPU space due to address space fragmentation. A slab allocator allocates only large chunks of memory, slicing them up into little chunks for particular classes of items, then maintaining freelists for each class whenever an object is freed. See the Bonwick paper in Resources for more details. Memcached currently generates slab classes for all power-of-two sizes from 64 bytes to 1MB, and it allocates an object of the smallest size that can hold a submitted item. As a result of using a slab allocator, we can guarantee performance over any length of time. Indeed, we've had production Memcached servers up for 4–5 months at a time, averaging 7,000 queries/second, without problems and maintaining consistently low CPU usage.

Another key requirement for Memcached was that it be lockless. All objects are multiversioned internally and reference counted, so no client can block any other client's actions. If one client is updating an object stored in Memcached while a dozen others are downloading it, even with one client on a lossy network connection dropping half its packets, nobody has to wait for anybody else.

A final optimization worth noting is that the protocol allows fetching multiple keys at once. This is useful if your application knows it needs to load a few hundred keys. Instead of retrieving them all sequentially, which would take a fraction of a second in network round-trips, the application can fetch them all in one request. When necessary, the client libraries automatically split multi-key loads from the application into separate parallel multi-key loads to the Memcached instances. Alternatively, applications can provide explicit hash values with keys to keep groups of data on the same instance. That also saves the client library a bit of CPU time by not needing to calculate hash values.

Client Libraries

The client/server interface to Memcached is simple and lightweight. As such, there are client libraries for Perl, PHP, Python and Java. I also hear that a coworker of mine has been working on a Ruby client, due out soon.

All of the clients support object serialization using their native serialization methods. Perl uses Storable, PHP uses serialize, Python uses Pickle and Java uses the Serializable interface. Most clients also support transparent compression, optionally only past a certain size threshold. Both serialization and compression are possible because Memcached lets client modules store opaque flags alongside stored items, indicating how they should handle the data coming out.

Using Memcached

Installing Memcached alone is no panacea; you have to do some work to use it. Profile your application and database queries to see where you're killing the most time and then cache from there. You also have to handle updating and purging your cache, because immediate cache coherency is important for most applications. If your application's internal API is already pretty clean, and you don't haphazardly hit the database all over your code, adding Memcached support should be easy. In your getter functions, simply try Memcached first. On a miss, hit the database and then populate Memcached. In your setter functions, update both the database and Memcached. You may find race conditions and cache coherency problems to deal with, but the Memcached API provides means to deal with them.

Memcached also is useful for storing data you don't really need to put on disk. For example, LiveJournal uses it to prevent accidental duplicate submissions of requests by storing the transaction's result code in Memcached, keyed by a transaction signature. Another example of Memcached as a primary data store, as opposed to a cache, is warding off dumb and/or malicious bots, often spammers. By keeping track of the last times and actions of each IP address and session, our code automatically can detect patterns and notify us of attacks

early on, taking automatic action as necessary. Storing this information in the database would've been wasteful, burdening the disks unnecessarily. Putting it in memory is fine, however, because the data is safe to lose if a Memcached node fails.

I asked the mailing list what interesting things they're using Memcached for, and here's what they said:

- Many people use it like we do on LiveJournal, as a typical cache for small Web objects.
- One site is using it to pass the currently playing song from their Java streaming server to their PHP Web site. They used to use a database for this, but they report hitting Memcached is much nicer.
- A lot of people are caching authentication info and session keys.
- One person reported speeding up mail servers by caching known good and known bad hosts and authentication details.

I continue to receive interesting e-mails and suggestions, so I'm happy that people are finding good uses for it.

Alternatives

If you can get away with running your threaded application on a single machine and have no use for a global cache, you probably don't need Memcached. Likewise, SysV shared memory may work for you, if you're sitting on a single machine.

A few people have suggested that MySQL 4.x's query cache might negate the need for Memcached. The MySQL query cache is emptied every time a relevant table is updated in any way. It's mostly a feature useful for read-only sites. LiveJournal is incredibly write-heavy, as are most high-traffic sites nowadays. Also, as with other databases, the MySQL caches together can't exceed the maximum address space the kernel provides, usually 3GB on a 32-bit machine, which gets to be cramped.

Another option for some people is MySQL's in-memory table handler. It wasn't attractive for my uses because it's limited to fixed-length records, not allowing BLOB or TEXT columns. The total amount of data you can store in it also is limited, so we still would've needed to run a bunch of them and fan out keys amongst them.

Acknowledgements

I'd like to thank Anatoly Vorobey for all of his hard work on the Memcached server, Lisa Phillips for putting up with early crash-prone versions and all the users on the mailing list who have sent in patches, questions and suggestions.

Resources for this article: /article/7559.

Brad Fitzpatrick has been hacking database-driven Web sites for eight years. In addition to riding his bike, Brad enjoys trying to think up alternative solutions to problems that otherwise might involve salespeople. Unless you're pitching blue pills or informing him of dead servers, Brad welcomes your mail at brad@danga.com.

Archive Index Issue Table of Contents

Advanced search

Data Acquisition with Comedi

Caleb Tennis

Issue #124, August 2004

One standard platform provides a uniform API for many data acquisition boards. You even can try it out with the standard PC parallel port.

Most scientists and engineers love data. The more data you can feed them, the more they smile. In a laboratory setting, data means everything. In order to spot trends, analyze strange phenomena and draw final conclusions, a lab person needs to make sure they have acquired a complete set of data.

The concept of data acquisition therefore encompasses a broad scope of ideas. Most scientists and engineers, however, agree that data acquisition is the result of the measurement of some natural process. This could be as simple as the measurement of a temperature, for example, or as complex as the measurement of impurities in molten steel.

In the computing world, data acquisition most commonly is done by measuring a voltage. To do so, it is necessary to have some sensor or measurement device that is capable of producing a voltage that the computer can measure. It's also important to know the correlation between the measured parameter and the sensor's voltage output. Ideally, the correlation is linear, as in a temperature sensor where 1 measured degree Celsius corresponds to .1 volts.

Modern motherboards have onboard sensors, such as National Semiconductor's LM78, which assess the overall health of the system. These sensors measure such conditions as cooling fan speeds, processor core voltages and temperatures and hard drive rotation speeds. This information is acquired by the chip and can be reported to the processor through a serial bus. The open-source project lm_sensors (secure.netroedge.com/~lm78) provides the software for monitoring many aspects of motherboards.

Typical personal computers have no common interface for analog data acquisition, however. In order to make some external voltage measurement, a

new interface is necessary. Data acquisition (DAQ) cards designed for either the PCI or ISA bus fill this gap. Many manufacturers make cards well suited for taking external measurements.

Table 1. Common Data Acquisition Channel Types

Name	Description
Analog Inputs	Measure external signals, such as a voltage
Analog Outputs	Send a variable signal
Digital Inputs/ Outputs	A discrete on/off signal; commonly 0 for off, 5 volts for on
Counters	Can count a number of pulses or measure frequency
Timers	Can measure the amount of time elapsed between two digital pulses

The Comedi Project

Most Linux users have experienced firsthand the complications surrounding having a single type of system (a printer, for example) and multiple models, makes, vendors and drivers. Any attempt at standardization becomes a large project. If the project receives enough support, it becomes the standard. Some vendors, like National Instruments, have released Linux drivers for their DAQ products, while others have not.

Comedi, or Control and Measurement Device Interface, is the standard suite of data acquisition drivers and libraries for Linux. Started in 1996 by David Schleef, Comedi attempts to support multiple vendors and models of cards through a common interface. In fact, the overall API design is a balance between modularity and complexity. Like other Linux driver projects, some of the work is the result of a lot of reading of hardware manuals, some is the result of reverse engineering and some is the result of manufacturers' assistance in providing Comedi support for their products.

How It Works

Comedi is separated into two parts. Comedi itself is the package of drivers that are loaded into kernel space, and comedilib gives user-space access to those drivers. It is through comedilib that the transparency of Comedi shines. Programs using Comedi can be written in C or C++. Perl and Python bindings also exist for Comedi.

Comedi breaks things down into channels, subdevices and devices. A channel is the lowest level of measurement or control. Multiple channels of the same type are grouped into a common set, called a subdevice. Then multiple subdevices are grouped together into a complete device. When using Comedi, first a Comedi driver is loaded into memory. Then, `/usr/sbin/comedi_config` is run to bind the driver to a Comedi device, such as /dev/comedi0. Finally, functions are available in comedilib to access the various devices on the DAQ card.

A Lab Example

One example of an application for DAQ and Comedi is the Analytical Engineering, Inc. (AEI) airflow laboratory. In the AEI lab, airflow is generated by a fan and is forced through orifices of varying sizes. Using a custom-written software application, a technician can monitor the pressure buildup across the orifice. In turn, this pressure buildup can be used to calculate the approximate amount of air flowing across the orifice. This calculation is vital, because it allows a technician to determine whether various meter calibrations are correct.

However, the actual mass flow is more difficult to calculate completely. This number requires knowledge of two air pressures, three airflow temperatures, humidity, barometric pressure and altitude.

Off-the-shelf components exist for converting these measurements to voltage; one of the most popular interfaces is 5B. Using 5B modular blocks, it's possible to transform all of these measurements to voltages the DAQ card can read.

Figure 1. Airflow Measurement Device

Using Comedi, reading these voltages becomes as trivial as using the comedi_data_read function. Calling this function and specifying a certain channel produces a resultant value, 3,421 for instance. But what does this number mean?

DAQ cards measure with a certain bit precision, 12 bits being the most common. They also specify a range or ranges of voltages over which they can be programmed to measure. Because a 12-bit number is represented from 0 to 4,095, it's easy to see that 3,421 is simply 3,421/4,095 * 100% of full scale (4,095). If the range of voltages is specified as [0, 5], then 3,421 would represent 4.177 volts.

Utilizing this information and knowing that the 5B block for temperature maps as [0 volts – 5 volts] → [0°C – 100°C], a small amount of programmatic math delivers a temperature of 83.56°C. Couple all of these measurements together, add a nice GUI interface and repeat the DAQ process every second.

More complex data acquisition can be performed as well. When acquiring data, it's important to make sure you sample fast enough so as not to miss any important information that occurs between samples. To support this, Comedi offers a command interface that can be used to set up synchronized sampling. Based on the sophistication of the DAQ card, timing can be handled by software interrupts or on-card interrupts.

Listing 1. Sample Program for Acquiring Voltage from One Channel

```
#include <stdio.h>
#include <comedilib.h>

const char *filename = "/dev/comedi0";
int main(int argc, char *argv[])
{
  lsampl_t data;
  int ret;
  comedi_t *device;

  /* Which device on the card do we want to use? */
  int subdevice = 0;
  /* Which channel to use */
  int channel   = 0;
  /* Which of the available ranges to use */
  int range     = 0;
  /* Measure with a ground reference */
  int analogref = AREF_GROUND;

  device = comedi_open(filename);
  if(!device){
    /* We couldn't open the device - error out */
    comedi_perror(filename);
    exit(0);
  }

  /* Read in a data value */
  ret=comedi_data_read(device,subdevice,
    channel,range,analogref,&data);

  if(ret<0){
    /* Some error happened */
    comedi_perror(filename);
    exit(0);
  }

  printf("Got a data value: %d\n", data);
  return 0;
}
```

Comedi shines in most data acquisition applications. In fact, Comedi's limit generally resides in the hardware on which it's being run. Less expensive cards typically have a slower scan rate ability. For fast data acquisition, most of the higher priced cards come with onboard DMA, allowing an onboard processor to handle the acquisition and allowing Comedi simply to route the acquired buffered data.

Listing 2. Code Snippet Demonstrating More Advanced Scanning by Using Commands and Triggers

```
  /* Goal: Set up Comedi to acquire 2 channels, and
     scan each set twice.  Perform the acquisition
     after receiving a trigger signal on a digital
     line.
  */

  comedi_cmd c, *cmd=&c;
  unsigned int chanlist[2];

  /* CR_PACK is a special Comedi macro used to
     setup a channel, a range, and a ground
     reference
```

```
    */

    chanlist[0] = CR_PACK(0,0,0);
    chanlist[1] = CR_PACK(1,0,0);

    /* Which subdevice should be used? */
    /* Subdevice 0 is analog input on most boards */
    cmd->subdev       = 0;
    cmd->chanlist     = chanlist;
    cmd->chanlist_len = n_chan;

    /* Start command when an external digital line
       is triggered.   Use digital channel specified
       in start_arg
    */

    cmd->start_src = TRIG_EXT;
    cmd->start_arg = 3;

    /* begin scan immediately following trigger */
    cmd->scan_begin_src = TRIG_FOLLOW;
    cmd->scan_begin_arg = 0;

    /* begin conversion immediately following scan */
    cmd->convert_src = TRIG_NOW;

    /* end scan after acquiring
       scan_end_arg channels
    */
    cmd->scan_end_src =     TRIG_COUNT;
    cmd->scan_end_arg =     2;

    /* Stop the command after stop_arg scans */
    cmd->stop_src =         TRIG_COUNT;
    cmd->stop_arg =         2;

    /* Start the command */
    comedi_cmd(device, cmd);
```

Fast scan rates don't translate to fast processing, however. Due to the non-deterministic nature of the stock Linux kernel, it's virtually impossible to handle acquisition and processing in real time—that is, to maintain strict scheduling requirements for a process. Help is available, however. The Linux Real-Time Application Interface (RTAI) and RTLinux are two of a small number of add-on packages that allow for better timing control in the kernel. Both packages provide interfaces to Comedi.

The basic idea behind these real-time interfaces is simple. Instead of running the kernel as *the* monolithic process, run it as a child of a small and efficient scheduler. This design prevents the kernel from blocking interrupts and allows it to be preempted. Then, any application that needs real-time control of the system can register itself with the scheduler and preempt the kernel as often as it needs to.

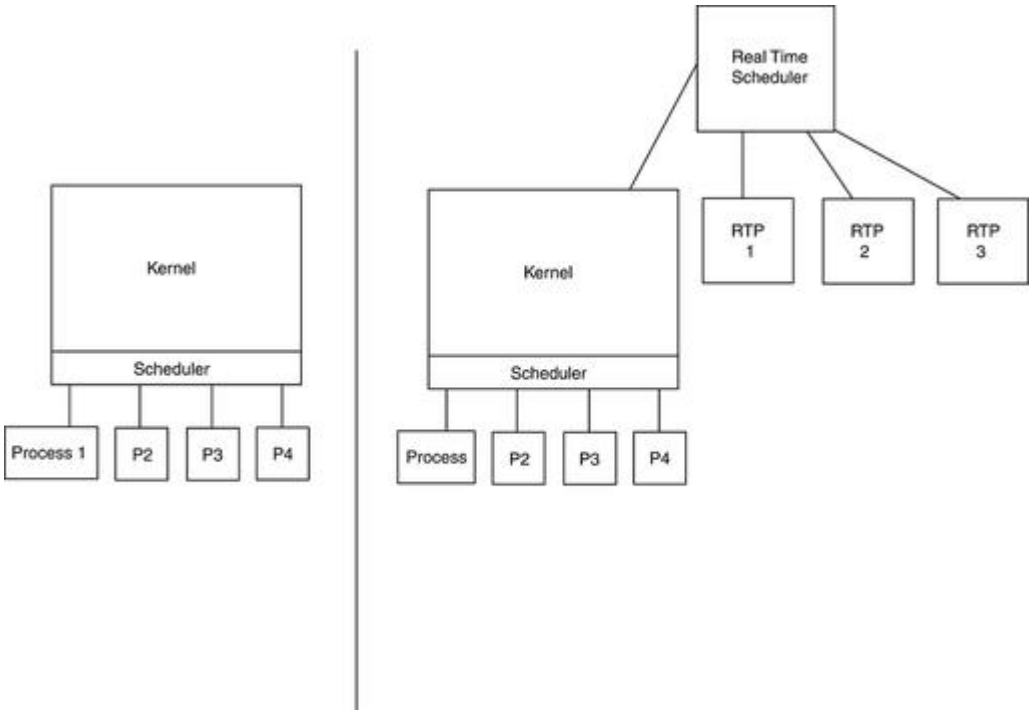


Figure 2. Normal Linux Process Scheduling vs. Real-Time Linux Process Scheduling

A Lab Example

AEI maintains a number of testing chambers for diesel engines, known as test cells. In a cell, an engine is equipped with a number of temperature and pressure measurement devices. A frequency measurement device also is used to measure the rotational speed of the engine. Finally, the engine is connected to a dynamometer, which simulates actual driving conditions by varying the resistance against the spinning engine. This results in generated torque, which is measured as well.

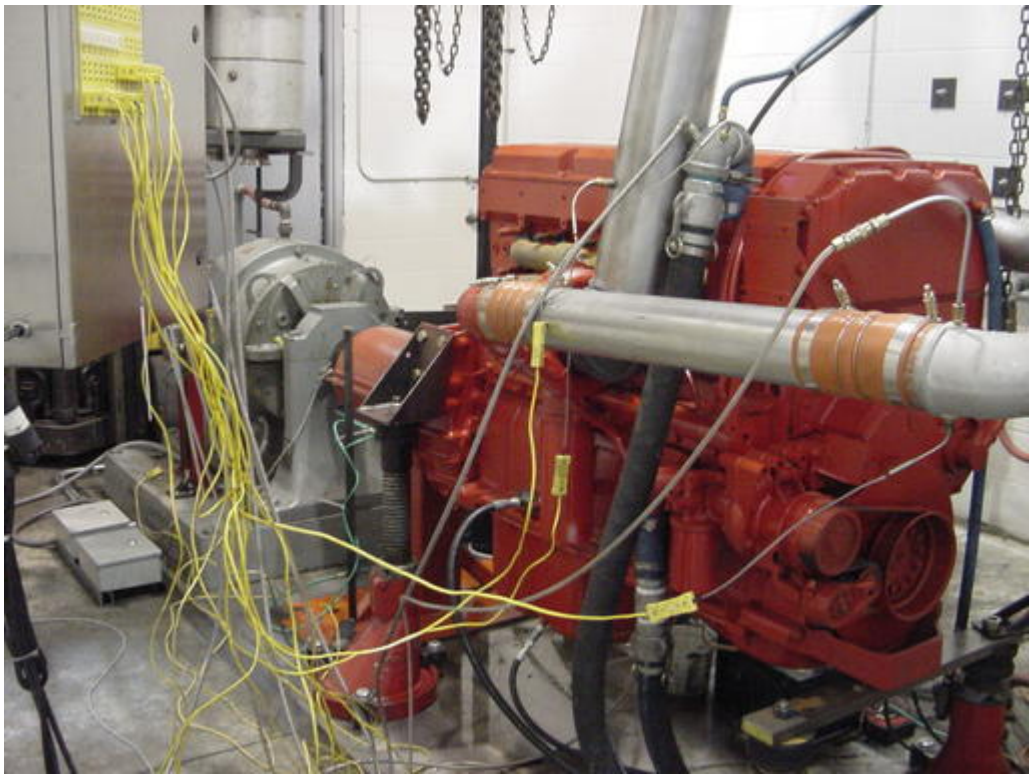


Figure 3. An Engine Being Instrumented

The actual scan rate of the engine data is slow, only 20 times per second. If the measurement of this data were the only required job, the overall setup would be straightforward. However, a number of variable parameters must be tuned and controlled with the newest acquisition of each set of numbers. The engine throttle position and dynamometer load amounts must be varied slightly to maintain the engine speed at a specific condition. Valves in the cell controlling cooling water flow must be adjusted to keep engine coolant temperatures at constant levels. Safety measures must be checked to determine that no catastrophic problem has occurred.

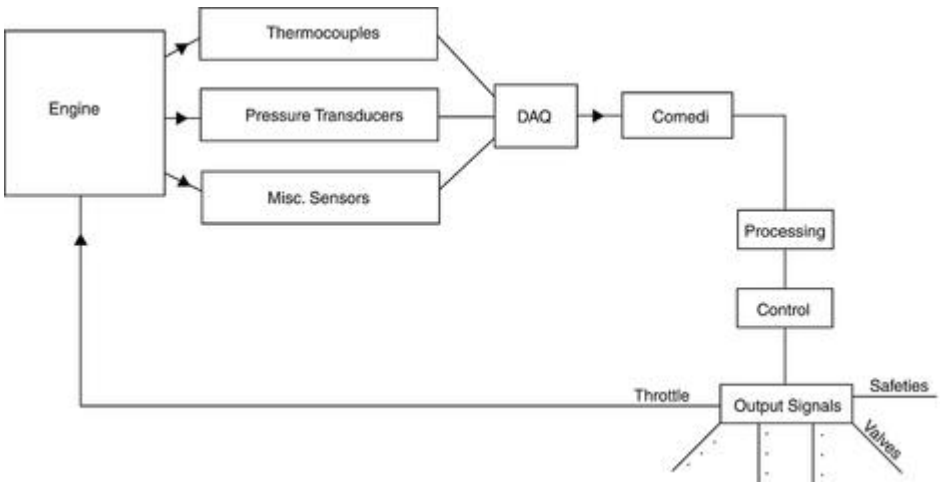


Figure 4. Overview of Engine Measurement and Control with Comedi

All of these checks and new control values must be taken care of before the kernel can return to handling the rest of its scheduling. If the Linux kernel were

to handle this scheduling on its own, it is quite possible that everything would work properly. However, it's impossible to determine beforehand when each stage of the process will be executed. With real-time extensions, however, the problem becomes trivial.

A real-time kernel is not without its downsides. While the real-time scheduler is executing some process at a fixed interval, the Linux kernel basically is put on hold. This means that a real-time process must be fast and efficient, and it must relinquish control back to the kernel as quickly as possible. Failure to do so results in sluggishness in the non-real-time portion of the system. If something goes wrong in the real-time process and control never goes back to the kernel, a complete system lockup can occur as well.

A Practical Example

Laboratory aside, sometimes it's interesting and fun to put Comedi to work at home. Low-end multipurpose data acquisition cards can be purchased for $99–$299 US, depending on brand, complexity and acquisition rate. Some examples of home projects include monitoring temperature in various parts of the house or scanning a magnetic sensor on a garage door to remind you that it's still open.

One interesting aspect of the personal computer is that parallel port lines can be controlled individually. Using Comedi, it's trivial to turn on and off these digital lines. When used with some form of relay, these digital lines can turn off and on anything imaginable.

Although parallel ports toggle between 0 and 5 volts, they typically do not have the capacity to source much electrical current. That said, it's a bad idea to connect the parallel port line directly to a device to turn it on or off without adding some kind of buffer circuitry. Many Web sites exist that explain how to create these circuits.

I use Comedi, an old 486 and two parallel ports to create an annual holiday light show. Lights are hung on the house in normal fashion, and a pair of wires for each set of lights is run back into the control room (a spare bedroom, in this instance). These power wires are connected to a custom-built circuit board that houses mechanical relays that send the power to the lights when they receive a 5-volt signal from the parallel port. A simple C program uses Comedi function calls to control the parallel port lines digitally, that is, to turn on and off the lights. Simple text files tell the program when to turn various lights on and off. And, the neighborhood receives a treat.

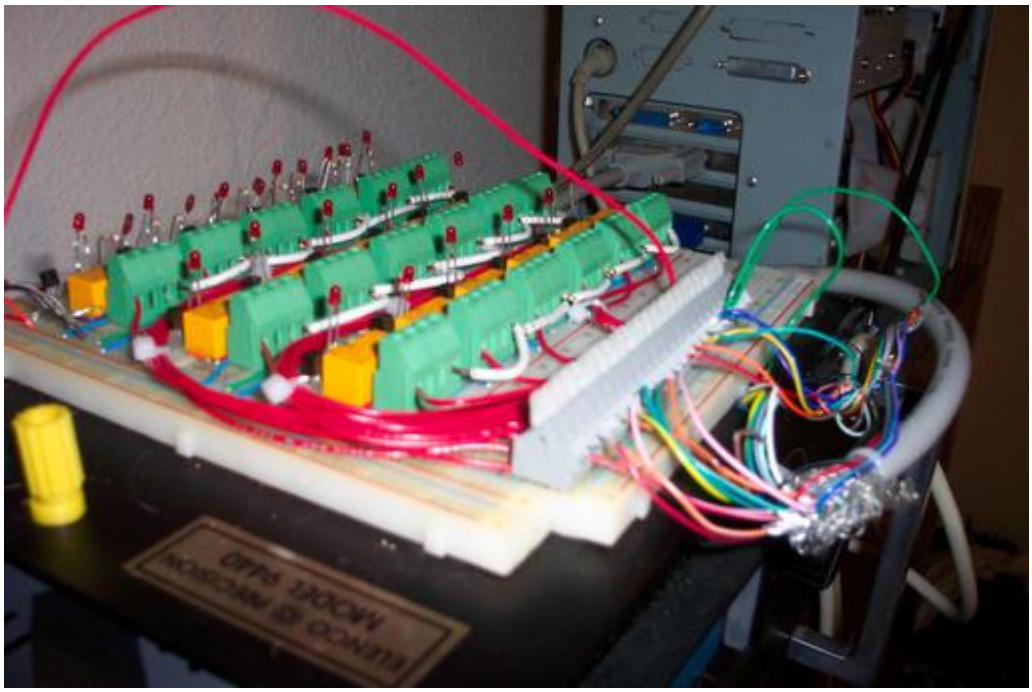


Figure 5. Interface Board for Parallel Port Light Show Display

Conclusion

Data acquisition is extremely valuable in the laboratory. The generic interface that Comedi provides allows great ease of use in Linux for a large number of available DAQ cards. As the popularity of Linux grows, the importance of having an interface such as Comedi's becomes vital.

Furthermore, as the low-end DAQ cards become even less expensive, Linux-based data acquisition becomes more and more appealing to hobbyists and do-it-yourselfers. What used to be an expensive set of software and hardware now is a viable method of implementation for a multitude of applications.

Resources for this article: /article/7610.

Caleb Tennis has been using Linux since 1996. He was the release coordinator of the KDevelop Project and now is focusing his attention on maintaining KDE for Gentoo. Besides overseeing engineering at a diesel engine test facility, he also teaches Linux part-time at a local college.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Declic: Linux 2.6 on the International Space Station

Taco Walstra

Issue #124, August 2004

The Declic experiment program on the International Space Station needs Linux to integrate reliably with a variety of microcontrollers with minimal intervention from the crew. Here's how the development is going.

In October 2001, three French scientists defined a new project for the study of phase transitions of fluids under microgravity conditions. Declic (Dispositif pour l'Etude de la Croissance et des Liquide Critiques) permits a wide experimental program, operated from the French USOC control centre in Toulouse in close relationship with the other control centers located at NASA and the European Space Agency (ESA). Scientists can do telescience experiments with real-time data sent from the Declic facility to ground, with almost no help from astronauts.

The only astronaut help needed is some exchanges of experiment boxes, the so-called inserts. ALI, one of the inserts, stands for Alice-like insert and refers to the previous experiments, Alice and Alice-2, from the Mir Space Station. Alice stands for Analyse des Liquides Critiques dans l'Espace (analyses of critical fluids in space). A critical fluid is a fluid at a specific temperature and density where the transition between fluid and gas behaves differently compared with the same fluid on Earth.

The French governmental space organization CNES is developing Declic, and it awarded the contract to the European aerospace organization EADS, a joint venture of the German Daimler-Chrysler Aerospace AG, the French Aerospatiale Matra and Spanish CASA. EADS is using four subcontractors for the actual development and is doing the integration tests and project control in Bordeaux. The University of Amsterdam in the Netherlands had experience with several of the previous critical point programmes, therefore we are developing a substantial part of Declic: two thermostat boxes where the experiments take place, the electronics, software for thermal regulation and parts of the data acquisition for scientific research. Two other subcontractors

are working on optics, data processing electronics, software for video cameras, data storage and the ISS interface. The fourth subcontractor is developing a complete experiment insert for solidification experiments. Electronics and software for this experiment also are being developed at our institute.

Figure 1 offers a simple overview of the several parts of the Declic facility, which basically contains two large boxes. The first box holds the experiment insert, which is surrounded by optics, video cameras and different sorts of sensors for observing the scientific phenomena. The fluids enclosed in a safe containment inside the insert are stabilized at a high-precision temperature. It's no simple house thermostat; it's a high-accuracy thermal control system that can keep fluids within 10 micro Kelvin of a specific temperature.

The second box (see also Figures 8 and 9) contains the electronics for data handling and temperature control. The electronics and software situated in this box is what I describe in this article. Two important subsystems are located in this second box, the power and data handling system (PDHS) and the central regulation electronics (CRE). The PDHS consists of a CompactPCI industrial Pentium PC running Linux, some microcontrollers and commercial PCI cards. It collects data coming from video cameras and the CRE, stores it on hard disk and interfaces with the ISS computers. Although a real-time link to ground exists, most of the data needs to be stored on hard disk. A removable hard disk will travel by space shuttle to give the scientists their valuable measurement data.

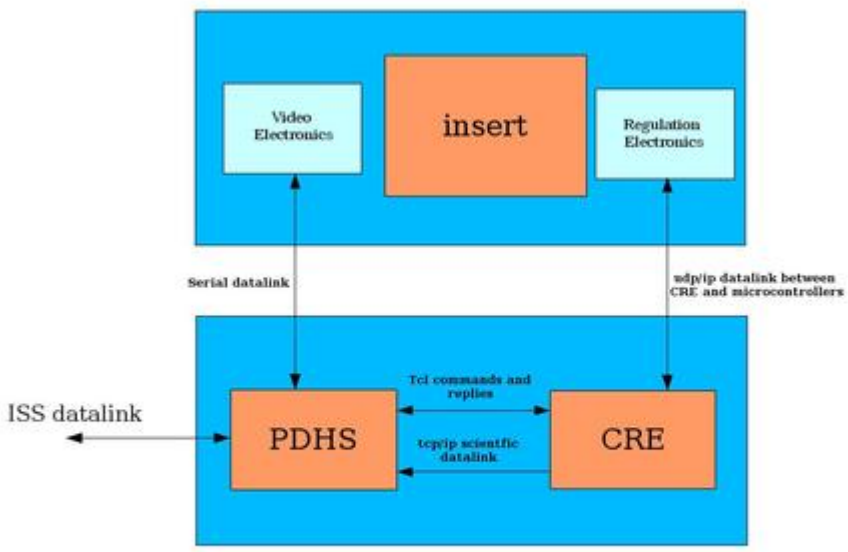


Figure 1. Block diagram of the Declic facility. The power and data handling system is a Pentium-based system running Linux.

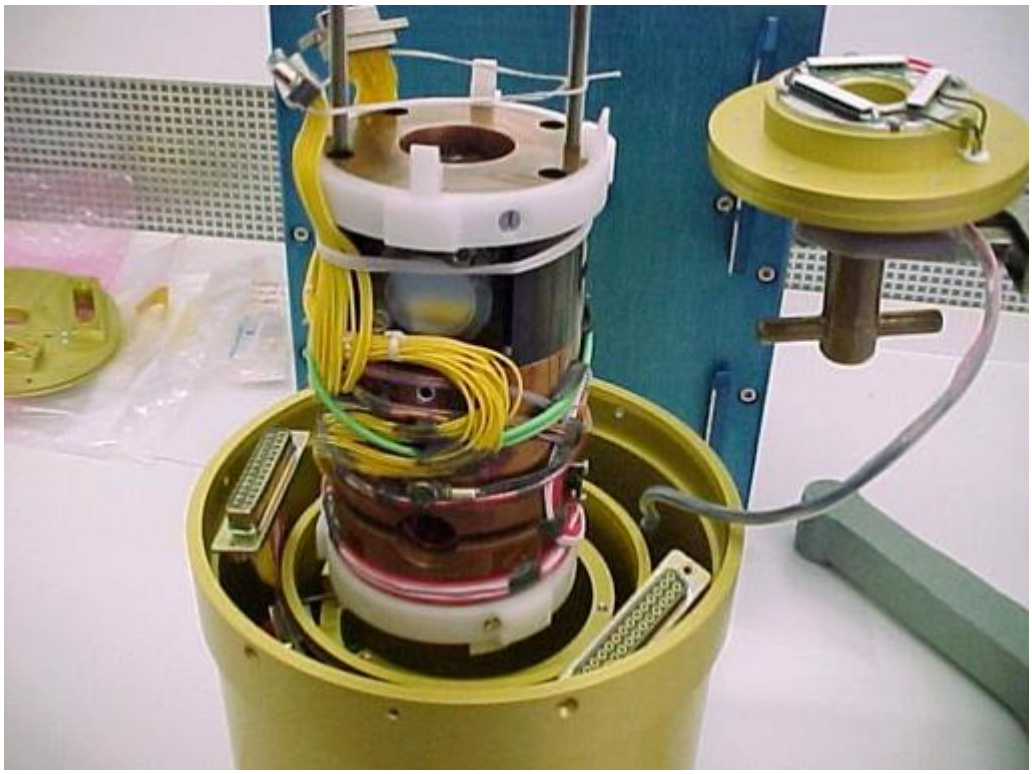


Figure 2. The sample cell unit of the ALI insert at the University of Amsterdam. The sample cell contains the liquid to be studied at a critical temperature. The blue box behind the SCU is the insert box.

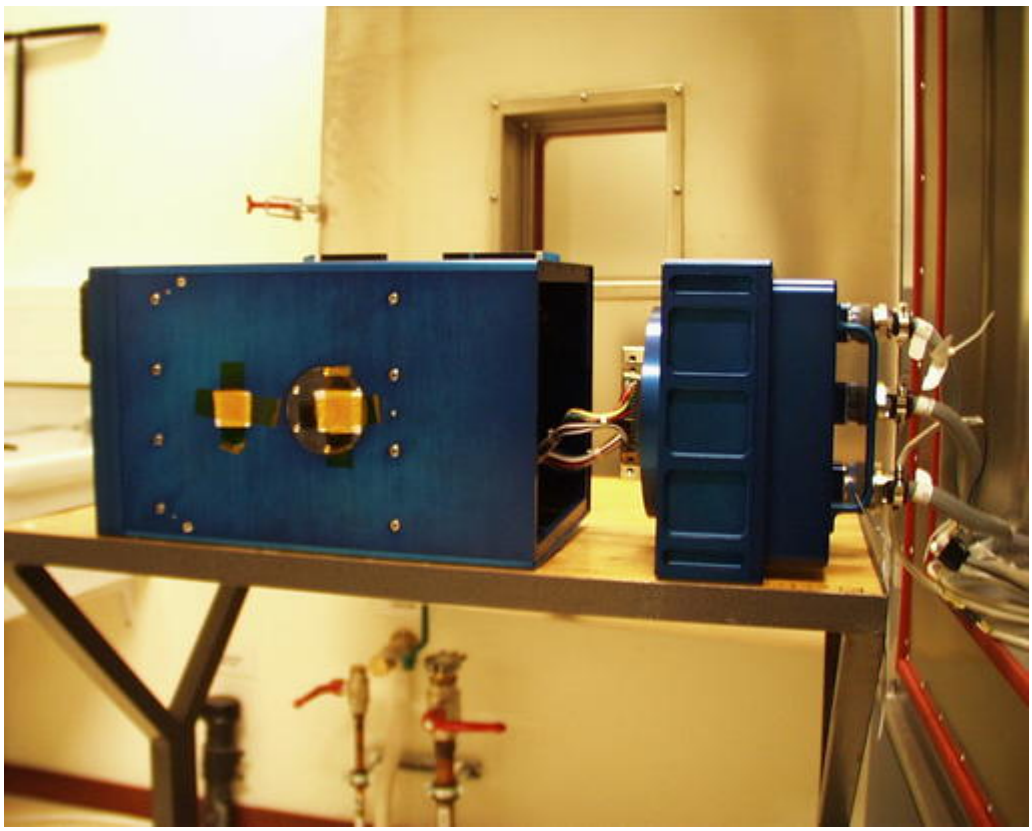


Figure 3. The ALI insert for study of liquids at critical point at low temperatures. The right side contains the microcontrollers and electronics for thermal control and scientific data acquisition.

Figure 4. Bart van Deenen Doing Thermal Testing of the ALI Insert

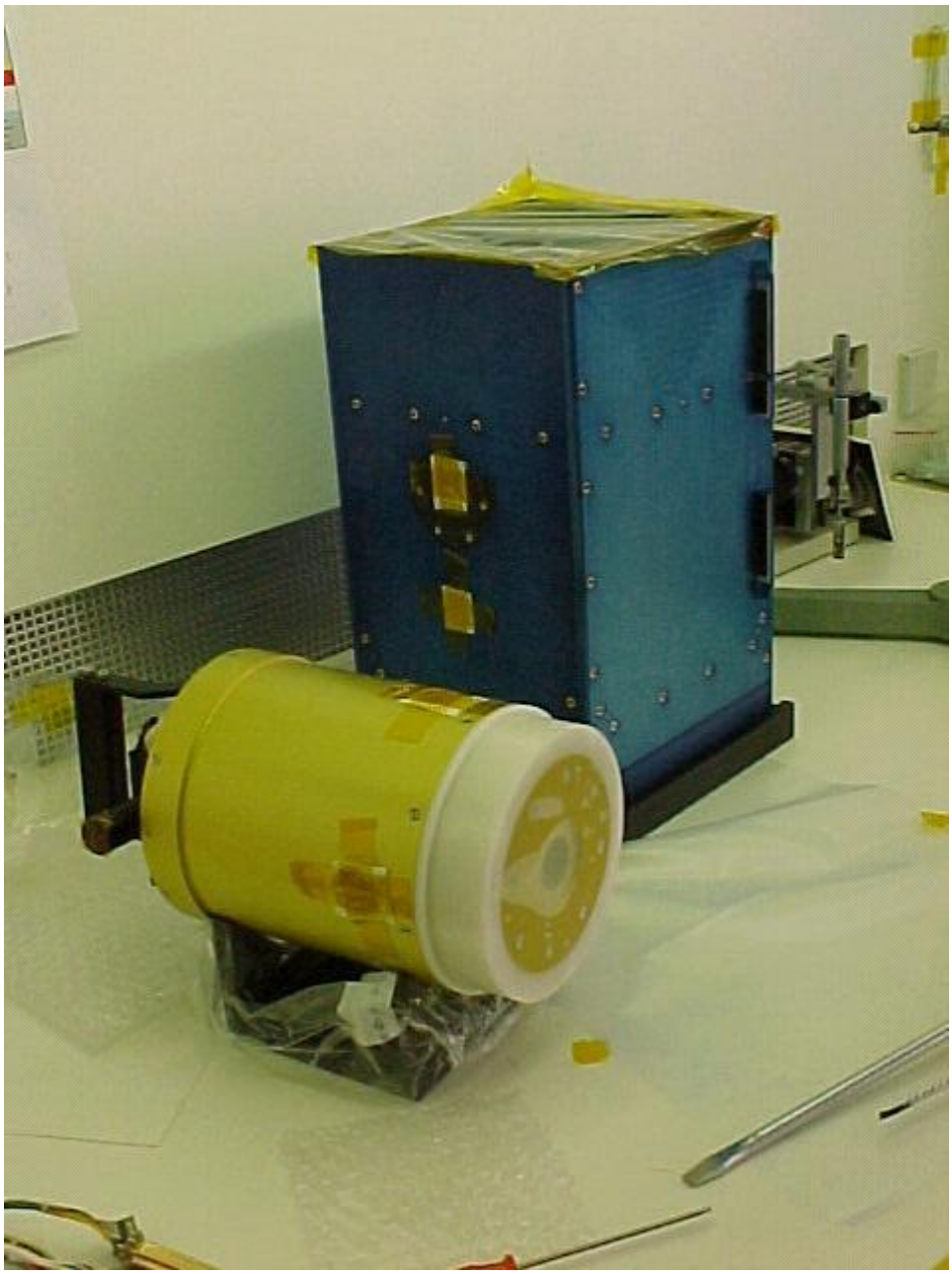


Figure 5. ALI Insert Box with Sample Cell Unit

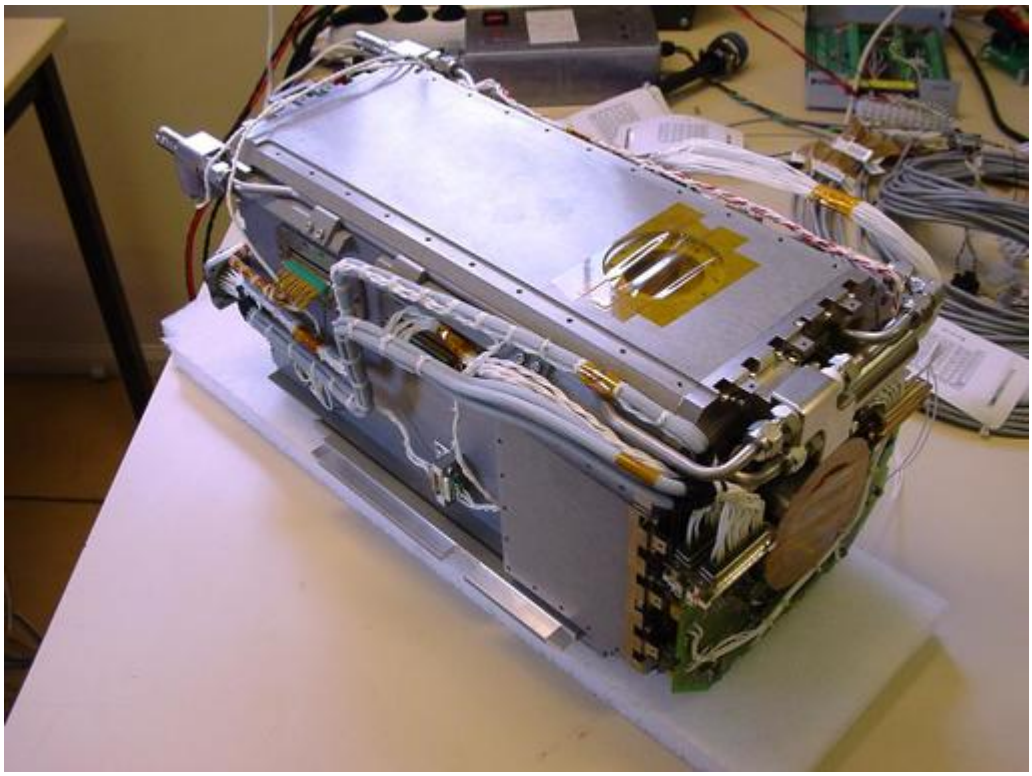


Figure 6. The DSI insert for solidification experiments at COMAT in Toulouse (France). The hole at the top is for accessing video cameras once located inside the Declic facility.

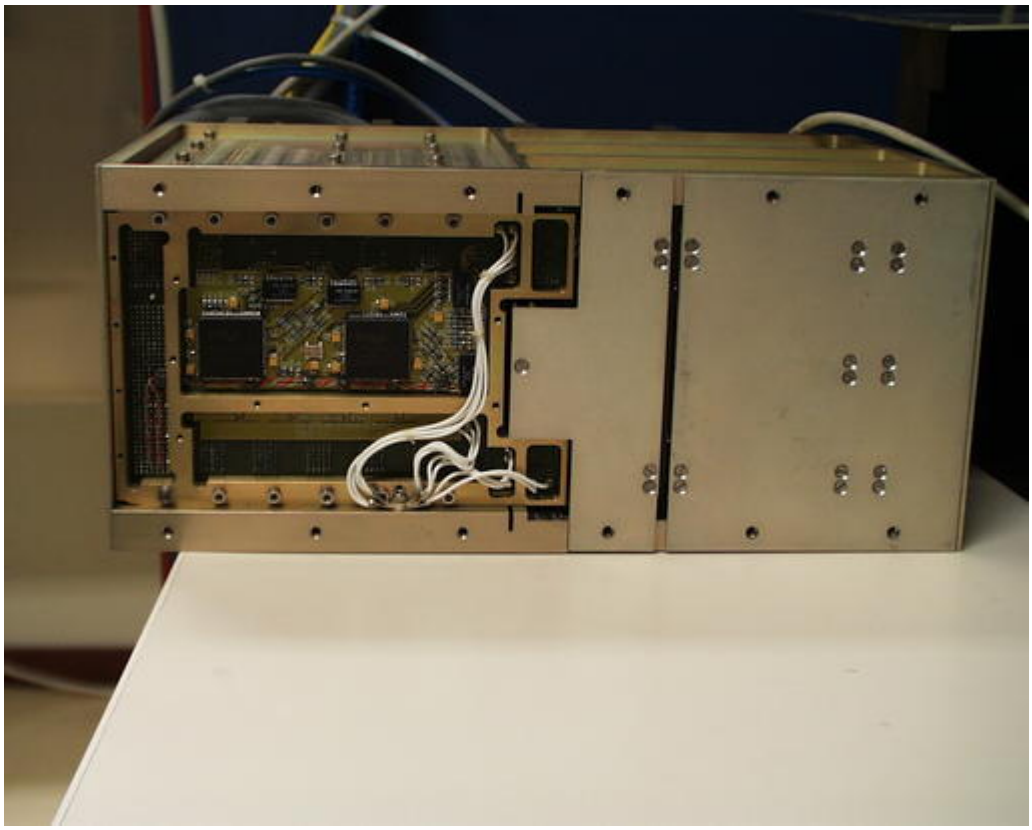


Figure 7. CRE Electronics Box to Be Located near the PDHS

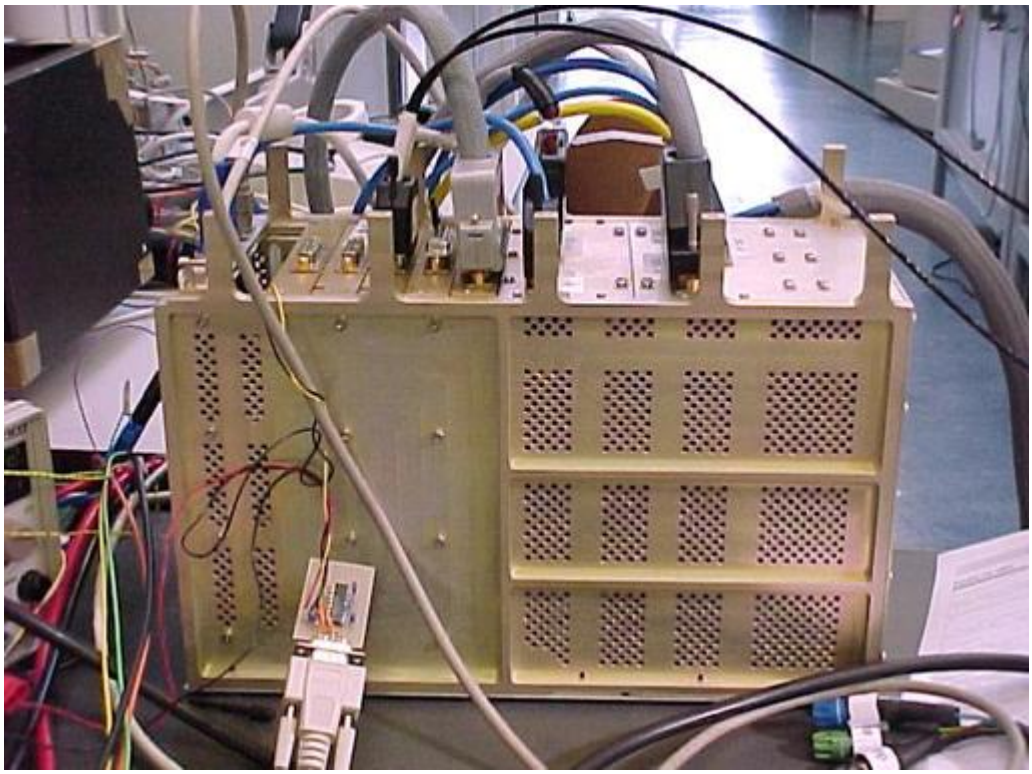


Figure 8. CRE Electronics Box during Testing

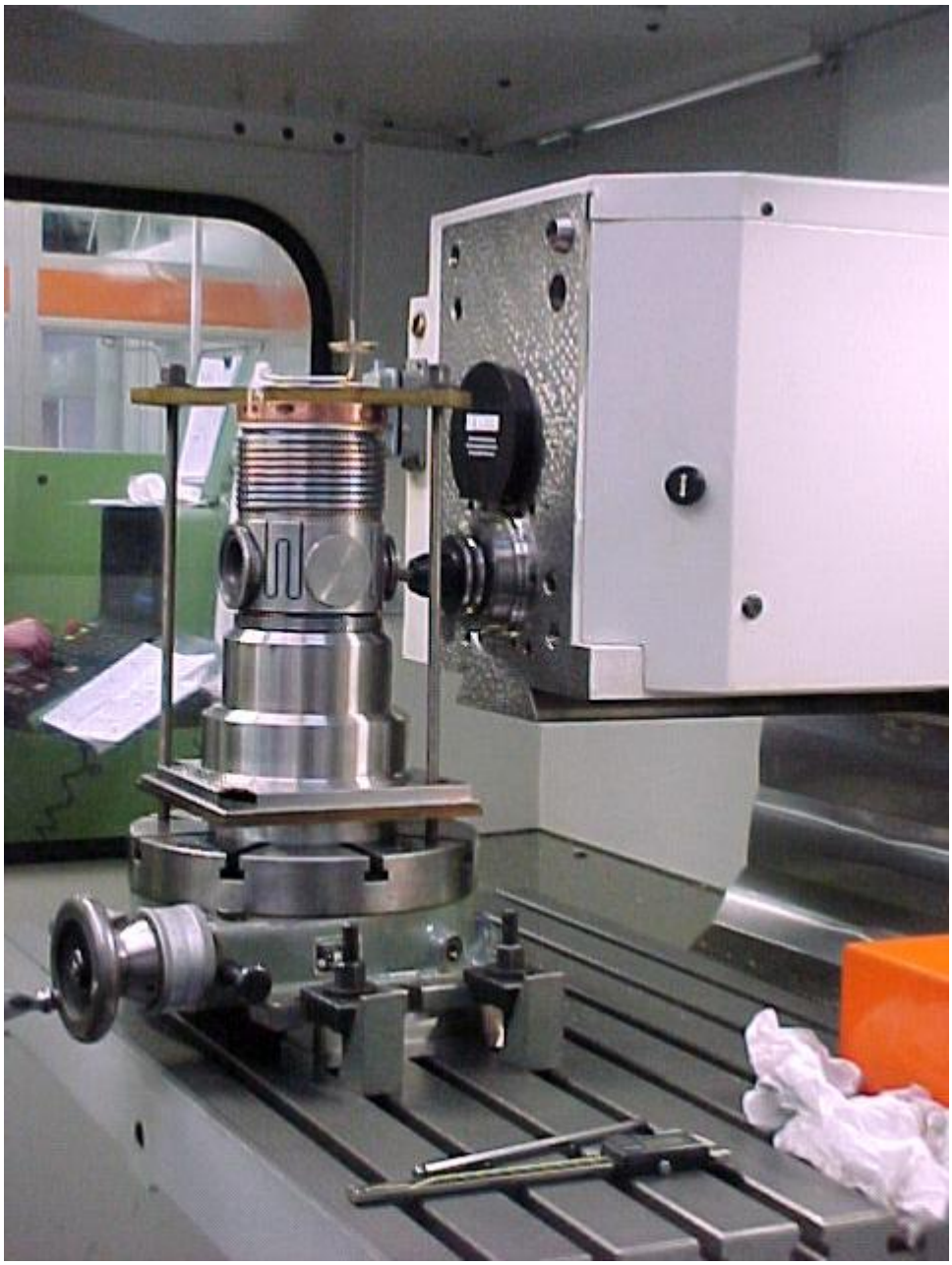


Figure 9. The HTI (High Temperature Insert) during manufacturing at the University of Amsterdam; this insert will be used to study water at 373°C.

Temperature control is handled by the CRE. The regulation electronics and software are able to control different types of thermostats inserted in the first box. In the previous experiments, one used fluids with a critical point of about 45°C; this will be done again in the first insert.

Another insert will study the critical point of water, which is near 373°C. At this temperature, water shows an unexpected aggressive behavior, which is scientifically very interesting. Currently, five different experiment inserts are being produced that will be situated inside Declic, all having different characteristics. For the critical point inserts developed in Amsterdam, we have chosen to use platinum resistors for temperature measurements, because it's the only sensor still functional at high temperatures that can maintain the

required accuracy. All these sensors are directed to microcontroller boards with analog-to-digital converters of 24 bits. Still, these 24 bits are not enough to reach the high accuracy expected, so all values of the A/D converter first are filtered digitally by an FPGA, a programmable chip. The microcontroller sends the data to the CRE Pentium PC; the CRE CPU gathers all data from different microcontrollers and sends the collected data on a TCP stream to the PDHS.

µC/OS and Linux

The complete Declic contains about ten small microcontrollers for dedicated hardware-related tasks, including the activation of heaters, the acquisition of a pressure or temperature sensor and controlling a stepper motor. The microcontrollers we use in Declic have a UDP/IP interface and run the real-time operating system µC/OS-II. Development of the source code for these controllers, however, was initialized on Linux, using an OS emulation layer.

Developing software for microcontrollers often is an annoying process of switching between downloading cross-compiled data over a serial link to a test board, debugging, recompiling your source code and resetting the test board. If you are lucky, the cross-compiler is Linux-friendly; unfortunately, many compiler environments are not. By using the Microsoft Windows emulator Wine, many of the cross-compilers can work together with Linux. In such a situation, you can use all the Linux tools that make software development so much easier. For our system, we chose µC/OS-II as a small real-time operating system able to run on 8-bit microcontrollers. µC/OS-II (often abbreviated to ucos) is distributed in source code form. You can purchase the book from the author and receive the source files of the OS, which can be ported to numerous microcontroller types. All the fundamental real-time OS aspects are there, including semaphores, mutexes and multitasking.

Every ucos function has a GNU C equivalent. For development, we used fake-ucos, a simple set of wrapper functions around standard GNU C equivalents. By using fake-ucos, it's possible to develop your microcontroller code on Linux. Then, you simply exchange the fake-ucos library for the real ucos source code and cross-compile the code for your favorite microcontroller. Of course, you need to extend all hardware-specific code details afterward, but it's certainly a great help in the early phase of a project.

Insert Definition Files

The Declic microcontrollers need to do different tasks, but the process actually comes down to reading something or controlling/activating something—read an AD converter, activate heater x, move stepper motor 1 to location y. The CRE computer, running generic software, needs to know the capability of each microcontroller and the insert characteristics. Each insert is different, and for a

future insert it's impossible to know any characteristic. How do we control an unknown set of things and read an unknown set of sensors of unknown type by software we want to keep as generic as possible?

We moved all hardware-specific control elements (AD conversions, activating heaters and stepper motors) into the microcontroller software. The microcontrollers are inside the insert, so each insert has its own hardware microcontrollers and software. A generic C program runs on the CRE main Linux computer for interfacing the PDHS with all the microcontrollers. This program handles the regulation algorithms and a Tcl command interface, which I cover later, and needs to collect all data coming in at different rates from all the controllers. The program is generic because it can control any type of insert, including future ones. High-accuracy thermal control algorithms are dependent on the type of insert; but these parts of the software are written in as a separate module. The command set is insert-independent, because it references only items and the items are described in XML format. An item can be a sensor (something that has a value we can read) or an actuator (something that can be written to). All these items are described in an Insert Definition File in XML format. A short example is given in Listing 1.

Listing 1. Part of an Insert Definition File

```
<ins_def> <board>

    <item>
        <pseudo_sensor name="BUILD_VERSION"
            max_sample_frequency="1"
            device_data_type="CHAR32">
        </pseudo_sensor>
    </item>

    <item>
        <sensor name="YSI_PRESSURE"
            max_sample_frequency="1"
            device_data_type="INT24"
            SI_data_type="FP32"
       </sensor>
    </item>
    .....
    <item>
        <desc>The Dallas board temperature sensor.
        </desc>
        <sensor name="DAL_ALI_POWER_BOARD2"
           dallas="1"
           device_data_type="INT16S"
           SI_data_type="FP32"
           unit="C">

           <parameter type="CHAR32" value="">
           </parameter>

        </sensor> </item>

    <item>
        <actuator name="PWM_OTSF"
            reg_actuator="1"
            upper_limit_SI="8.8"
            device_data_type="INT16U"
            SI_data_type="FP32"
            unit="W">
```

```
            <parameter type="FP32" value="65.0">
            <desc>Resistance of heater</desc>
            </parameter>

            <parameter type="FP32" value="30.0">
            <desc>power supply voltage for this
            channel</desc> </parameter>

        </actuator>
    </item>
    .....
    </board>
    ...
</ins_def>
```

The Insert Definition File describes the insert from a software point of view. It consists of a description of all sensors and actuators (items) that can be controlled by a certain microcontroller board. Every item can have a device and an SI value. The device value represents the raw data from, for example, an AD converter, while the SI value is the human-readable converted value, such as Watts, Ohms or degrees Celsius. In this way, we are able to write a Watts value to a heater, and it's up to the controller to figure out what exactly should be written to an FPGA to get the heater to produce this number of Watts. The microcontroller uses the parameters for these calculations, which make the source code independent from such hardware characteristics as heater resistance or power supply voltage. When we read the XML file, the items get numbers in two different ways: an incremental counting for all items of the insert and a local counting for the items controlled by a specific microcontroller board. The scientist has a simple list of all available items that can be controlled with a set of Tcl commands.

Experiment Description in Tcl

In past experiments on the Russian Mir station, all experiment timelines had to be stored on a computer. Experiment control existed by switching on the system and executing a series of commands. The experiment command list had few possibilities to act on specific occurring phenomena or experiment phases. In an early stage of the Declic development, the French company EREMS, responsible for the PDHS development, came up with the idea to use Tcl for the interface. Scientists now are given a complete programming language to formulate the experiment. They can execute commands, store values from read commands in variables and make decisions using the Tcl language for configuring the next executed command. The science script is located on one of the PDHS hard disks and started from the ground. For example, a Tcl science script can contain Tcl statements to bring the experiment to a specific temperature, start the video if the insert has reached a stable temperature and start the acquisition of some interesting sensors.

When writing a science Tcl script, the scientist doesn't need to know much about the hardware; a simple list of items with their numbers is enough. Reading a certain temperature sensor can be done in this way by executing the Tcl command `cre_get_values -item 34`. The science script executes on the PDHS, and a CRE command results in passing the Tcl command to the CRE computer. The CRE knows that sensor 34 is actually the third sensor on a Platinum sensor controller board. It sends a binary equivalent command to this board, `cre_get_values -item 3`. The controller makes a data acquisition of the sensor and responds with the value. This response once again is sent back to the PDHS and the running Tcl script.

Now, suppose you want to add an item to the Insert Definition File—how do you keep this XML file consistent with the microcontroller software? It's easy for a Linux system to interpret XML files, but small 8-bit controllers certainly cannot do such things. Adding one item to the Insert Definition File, however, can change all item numbers by one, making the local item numbers no longer consistent with the software implementation. The solution is found in generating C code and header files automatically from the XML file. Putting both files into the Makefile keeps everything consistent. Listing 2 gives a small example of such a generated C file.

Listing 2. Excerpt of a C File Generated from an Insert Definition File in XML

```
....
char item_names[]={
    "BOARDVERSION",
    "BOARDID",
    "DAL_ALI_POWER_BOARD1",
    "PWM_OTSF",
    ....
    "U_FPE"};

void init_items(void){ t_item *items;

    item = g_items[ITEM_NR_BOARDID];
    item-> item_type=pseudo_sensor;
    item-> itemnr=0;
    item-> device_data_type=11;
    MALLOC(item-> device_value, void, 32);
    item-> SI_data_type=0;
    item-> SI_value=NULL;
    item-> in= NULL;
    item-> out= NULL;
    ...
```

In Listing 2, the last two entries are pointers to functions that perform the actual acquisition for this item when it's a sensor; it also performs the action for an actuator.

Using the 2.6 Kernel

We saw that a single Tcl command results in a multistage command transmission between different Declic subsystems. It's possible to acquire multiple sensors with a single Tcl command, even if they are distributed over several microcontrollers. This results in the arrival of several packets on the CRE computer that need to be collected and sent as one TCP packet to the PDHS. In such cases, a responsive Linux kernel is crucial.

The central CPU of the CRE uses the new 2.6 kernel. It is compiled into a Net-bootable image: the CPU boots using the bootp and TFTP protocols for LAN booting. The image is located on the hard disk of the PDHS, which is booted in a normal way from hard disk. All the microcontrollers boot in the same way. When data transfers take place between microcontrollers and the CPU, a responsive operating system is necessary: we don't want to have the system interrupting the transfers for more than 20ms as sometimes happens in kernel 2.4.

The requirements are soft real time here, and 2.6 fulfills these requirements. All we need is a kernel that immediately triggers a C thread in the main CRE application when data arrives from a microcontroller. The microcontrollers have hard real-time requirements, because they need to be responsive to such hardware events as hardware counters and interrupting devices. Our major application running on the CRE central processor has no such demands. When we started to run the application, using Linux with a 2.4 kernel with several microcontrollers sending data, we encountered timeout problems for packets that were not received in time; although the contents of these packets showed they were sent with correct timestamps. The major improvement with the 2.6 kernel is the low latency of system interrupts.

Conclusion

Linux nowadays is a common tool for space-related projects, whereas several years ago, proprietary systems, such as VRTX, QNX or VxWorks, were leading. In addition, this past year even led to FlightLinux, a standard Linux distribution adapted to spacecraft environments. Open-source software is of crucial importance for these kinds of projects, and our experience with Linux has proven that it has a great future in space.

Resources for this article: /article/7621.

Taco Walstra is a software engineer at the University of Amsterdam. He enjoys rock climbing and playing different types of lutes. He can be reached at walstra@science.uva.nl.

Snooping the USB Data Stream

Greg Kroah-Hartman

Issue #124, August 2004

Follow along with the kernel hacker's actual problem-solving process as the quest to add support for a new device begins.

Day 1: I open the box to see a small USB device—no bigger than a quarter—a CD and a note from my editor, “Make this work on Linux!” “Okay”, I think, “this should be easy.”

Figure 1. MARX Software Security's CrypToken

I plug the device in to my laptop and run a small program called usbview to learn what the Linux kernel thinks this device is (Figure 2). This device must be calling itself a USB CrypToken, as that is the string contained in the device. Unfortunately, the device name is in red, which means no kernel driver is

bound to the device. I either have to write one or find a way to use libusb to talk to the device from user space (see my article "Writing a Real Driver—in User Space", *LJ*, June 2004, for more information about libusb).

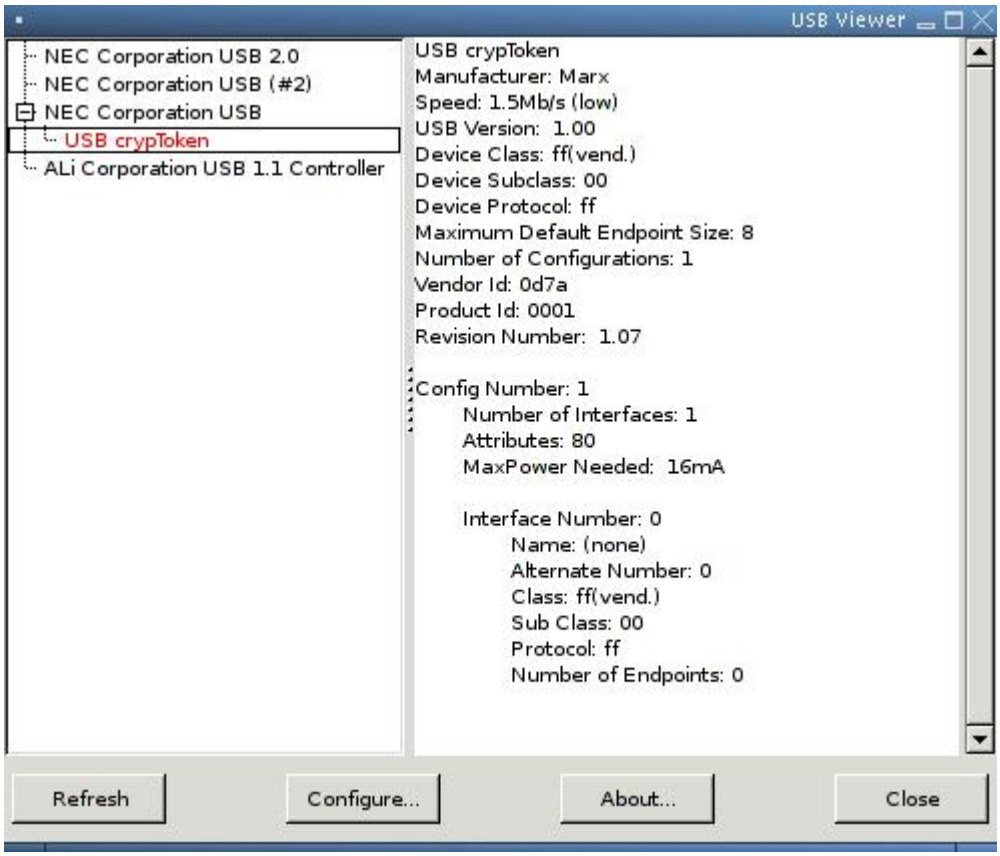


Figure 2. usbview identifies the device by vendor and product ID.

Not content to rely on pretty GUI programs, I poke around in the /proc/bus/usb/devices file to get the raw device information. The section that describes this device looks like this:

```
T:  Bus=01 Lev=02 Prnt=03 Port=02 Cnt=01 Dev#=  4 Spd=1.5 MxCh= 0
D:  Ver= 1.00 Cls=ff(vend.) Sub=00 Prot=ff MxPS= 8 #Cfgs=  1
P:  Vendor=0d7a ProdID=0001 Rev= 1.07
S:  Manufacturer=Marx
S:  Product=USB crypToken
C:* #Ifs= 1 Cfg#= 1 Atr=80 MxPwr= 16mA
I:  If#= 0 Alt= 0 #EPs= 0 Cls=ff(vend.) Sub=00 Prot=ff Driver=(none)
```

Curious to know whether any other Linux user has tried this device, I consult the Linux USB Working Devices List (see the on-line Resources section). Plugging the vendor ID of 0d7a in to the Quick Search field results in no records found. Perhaps this project will take more work than I thought.

Day 2: The CD, where did I toss it? I locate it and put it in the drive and, look, there's a file called linux.txt on it. Wow, a vendor that acknowledges that Linux might be a viable operating system to support—things sure have changed over the years. After poking around further and reading the documentation on the CD, I realize the device is a small crypto token that can be used to do all sorts of

fun things, such as read a unique serial number from the device (each device is different), encrypt data through the device with a 128-bit key stored only in the device and save data on the device in a secure storage area.

On the CD is a shared library that can be used to talk to the USB device to allow a program to access the functions provided by this device. Also present is a small test program that shows how the different library functions work. The library uses libusb to talk directly to the device from user space, which means that a kernel driver is not necessary for this device. The library's license does not allow it to be used within a program that would be licensed under the GPL, however, which is unfortunate for many potential uses. I need to find some way to allow GPL programs to talk to the USB device.

Day 3: While rummaging through my old collection of USB patches, I dig up a reference to a developer who modified the kernel usbfs core code to log all data that flows through it. This patch would allow anyone to read the raw USB data for any program that uses usbfs to talk to a USB device. Because libusb uses usbfs to communicate with USB devices, this might offer a way to reverse engineer this device. Unfortunately, the patch wasn't present with the reference, and no amount of digging on the Internet turned up any real code.

Day 4: As there is no available patch to do what I want to do, I might as well do it myself. So, off to grab the latest 2.6 kernel source tree and dive in.

The files inode.c, devices.c and devio.c in the drivers/usb/core/ directory of the kernel source tree implement the usbfs filesystem. The main filesystem code is in the inode.c file. It contains all of the various VFS code that creates a virtual filesystem and the virtual files within it. The file devices.c handles the creation and reading of the /proc/bus/usb/devices file. This file shows all USB devices and information for those devices in the system at the present time.

The file devio.c controls the raw access of USB devices through the usbfs filesystem. For a user-space program to talk to a USB device through usbfs, it needs to use the ioctl() command on a file that represents the USB device. All of the different ioctl messages that can be sent to the USB devices through usbfs are detailed in the include/linux/usbdevfs.h file.

So, in order to log all accesses to all devices through usbfs, the devio.c file should be modified. Digging into the file, the function usbdev_ioctl looked like the proper place to do this logging. It is called for every ioctl call to a usbfs file. Within that function is a big switch statement that calls the proper functions, depending on the different ioctl command. That is the perfect place to log what kind of command was sent to the device. So, I added a simple printk() call to each case statement, causing them to look like this:

```
...
case USBDEVFS_CLAIMINTERFACE:
  printk("CLAIMINTERFACE\n);
  ret = proc_claiminterface(ps, (void __user *)arg);
  break;

case USBDEVFS_RELEASEINTERFACE:
  printk("RELEASEINTERFACE\n");
  ret = proc_releaseinterface(ps, (void __user *)arg);
  break;
...
```

A simple compile, install and module load later confirmed that every usbfs access is now logged to the kernel log, which can be seen by running the dmesg program. I determined that running the lsusb program as `lsusb -v` produced a lot of usbfs accesses as the program retrieves all of the raw USB configuration data from all devices.

Day 5: Now that the different kinds of usbfs accesses can be noticed easily, it is time to log the data these accesses generate. In looking at the description of the device in the /proc/bus/usb/devices file, it appears that I care only about the accesses to the control endpoint, because there are no endpoints assigned to this device.

Digging further into the devio.c file, I determine that the proc_control() function handles all control messages. There, the code determines whether the request is a read or write control message with the code:

```
if (ctrl.bRequestType & 0x80) {
```

The USB bRequestType variable is a bitfield, and the uppermost bit determines the direction of the request. So, in the read section of this if statement I add the lines:

```
printk("control read: "
       "bRequest=%02x bRrequestType=%02x "
       "wValue=%04x wIndex=%04x\n",
       ctrl.bRequest, ctrl.bRequestType,
       ctrl.wValue, ctrl.wIndex);
```

to log the control request information. After the read has completed, I add the following lines to log the actual data read from the device:

```
printk("control read: data ");
for (j = 0; j < ctrl.wLength; ++j)
    printk("%02x ", ctrl.data[j]);
printk("\n");
```

After doing much the same modification to the write section of the if statement, I build, reload the usbcore modules and verify that I now can log all control messages to and from the device. The messages returned are:

```
CONTROL
control read: bRequest=06 bRrequestType=80 wValue=0300 wIndex=0000
control read: data 00 00 61 63
```

Day 6: Looking at the modifications I have made to the kernel code, I think this work might be something other users might like to have. So, it is time to clean up the code to a state that the USB maintainer might accept for the main kernel source tree.

First, I recognized that the calls to printk() are incorrect. All printk() calls must be accompanied by a proper logging level. These logging levels are added to printk calls by pre-appending the proper KERN_ values to the message. The file include/linux/kernel.h contains the following valid values that must be used:

```
#define KERN_EMERG   "<0>" /* system is unusable               */
#define KERN_ALERT   "<1>" /* action must be taken immediately */
#define KERN_CRIT    "<2>" /* critical conditions              */
#define KERN_ERR     "<3>" /* error conditions                 */
#define KERN_WARNING "<4>" /* warning conditions               */
#define KERN_NOTICE  "<5>" /* normal but significant condition */
#define KERN_INFO    "<6>" /* informational                    */
#define KERN_DEBUG   "<7>" /* debug-level messages             */
```

So, I change the printk calls in the usbfs_ioctl() function from:

```
printk("CLAIMINTERFACE\n);
```

to:

```
printk(KERN_INFO "CLAIMINTERFACE\n);
```

Now the kernel janitors should not complain about improper printk() usage.

In looking further at the logging messages, however, it is hard to determine for what exact device the message is being logged. More information needs to be added to the printk() calls. Luckily, some macros already in the include/linux/device.h file can help us. They are the dev_printk() macro and its helper macros, dev_dbg(), dev_warn(), dev_info() and dev_err(). These macros all need an additional pointer to a struct device variable, which allows them to print out the unique device ID for the message. So I change the printk() calls again to look like this:

```
dev_info(&dev->dev, "CLAIMINTERFACE\n");
```

Then the control message printk() calls are changed to:

```
dev_info(&dev->dev, "control read: "
       "bRequest=%02x bRrequestType=%02x "
       "wValue=%04x wIndex=%04x\n",
       ctrl.bRequest, ctrl.bRequestType,
       ctrl.wValue, ctrl.wIndex);

dev_info(&dev->dev, "control read: data ");
for (j = 0; j < ctrl.wLength; ++j)
    printk("%02x ", ctrl.data[j]);
printk("\n");
```

The printk calls that dump the data do not need to be changed, as they still are printing on the same line as the call to dev_info().

Now the log messages are much more informative, looking like the following:

```
usb 1-1: CONTROL
usb 1-1: control read: bRequest=06 bRrequestType=80 wValue=0300 wIndex=0000
usb 1-1: control read: data 00 00 61 63
```

I can determine exactly what USB device is being talked to, which helps me weed out the messages for devices I do not care about.

Day 7: Oops, I now realize that if I expect this kernel change to be accepted by the community, I had better not always generate these messages. Otherwise, everyone would have their system logs overflowing with messages they do not care about. How to log messages only when asked?

I first look into making a new kernel build configuration option. A simple modification of the drivers/usb/core/Kconfig file adding a new option is simple, but in examining the required code changes, I soon realize that wrapping all of the new logging statements in a #ifdef CONFIG_USBFS_LOGGING statement would make the USB maintainer reject my kernel patch. #ifdef is not generally allowed within kernel code, as it cuts down on readability and makes maintaining the code over time almost impossible.

Instead, I look at making an option that can be changed at runtime. I add the following lines of code to the devio.c file:

```
static int usbfs_snoop = 0;
module_param (usbfs_snoop, bool, S_IRUGO | S_IWUSR);
MODULE_PARM_DESC (usbfs_snoop, "true to log all usbfs traffic");
```

This adds a new module parameter to the main usbcore module called usbfs_snoop. This can be seen after building the code by running the modinfo program:

```
$ modinfo usbcore
license: GPL
parm:    blinkenlights:true to cycle leds on hubs
parm:    usbfs_snoop:true to log all usbfs traffic
```

By loading the module with the following line:

```
modprobe usbcore usbfs_snoop=1
```

the option can be enabled by the user.

I used the definition module_param() instead of the old-style MODULE_PARM(), as this is the proper way to describe module parameters in the 2.6 kernel. The main difference is this definition has a third parameter. This third parameter, if set to something besides 0, causes the parameter to show up in sysfs and allows a user to query and modify the option while the module is loaded. With this code included, the usbcore module's directory in sysfs looks like:

```
$ ls -l /sys/module/usbcore/
-r--r--r--  1 root root 4096 May 13 15:33 blinkenlights
-r--r--r--  1 root root 4096 May 13 15:33 refcnt
-rw-r--r--  1 root root 4096 May 13 15:33 usbfs_snoop
```

The module now can be loaded as normal:

```
modprobe usbcore
```

When I decide to turn on logging I simply do:

```
echo 1 > /sys/module/usbcore/usbfs_snoop
```

and the kernel variable usbfs_snoop in the devio.c file is changed on the fly.

Now that I can determine whether the user wants to print out snooping messages, I need to modify the dev_info() calls again. I create the following macro to do this:

```
#define snoop(dev, format, arg...)             \
    do {                                       \
        if (usbfs_snoop)                       \
            dev_info( dev , format , ## arg);  \
    } while (0)
```

This macro tests the value of the usbfs_snoop variable, and if true, the dev_info() line is called. The macro is wrapped in a `do { } while (0)`

statement to allow it to be used in any kind of code without having to worry about any side effects. All kernel macros containing more than one line of code are written in this way for this reason. For more details about this, read the kernel newbies FAQ (see Resources).

I next change all previously added calls to dev_info() to a call to snoop(), causing the code to look like:

```
snoop(&dev->dev, "control read: "
       "bRequest=%02x bRrequestType=%02x "
       "wValue=%04x wIndex=%04x\n",
       ctrl.bRequest, ctrl.bRequestType,
       ctrl.wValue, ctrl.wIndex);
```

But where the data is printed out, the snoop() macro does not work properly. I need to check the value of the usbfs_snoop variable directly, wrapping the code in an if statement:

```
if (usbfs_snoop) {
    dev_info(&dev->dev, "control read: data ");
    for (j = 0; j < ctrl.wLength; ++j)
        printk("%02x ", ctrl.data[j]);
    printk("\n");
}
```

I'm happy, and hopefully the USB maintainer also will be happy with the changes. I read how to generate a proper kernel patch by consulting the file Documentation/SubmittingPatches, generate a diff file and e-mail it off.

We now have a way to snoop all usbfs traffic, which can help us reverse engineer any device that uses libusb to communicate with a USB device. It also allows us to snoop any USB accesses from a guest OS running in a VMware session, allowing the possibility to reverse engineer Microsoft Windows USB drivers much more easily. But all of that has to wait until the next column.

Resources for this article: /article/7605.

Greg Kroah-Hartman currently is the Linux kernel maintainer for a variety of different driver subsystems. He works for IBM, doing Linux kernel-related things, and can be reached at greg@kroah.com.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

At the Forge

Weblogs and Slash

Reuven M. Lerner

Issue #124, August 2004

User journals with comments are one feature of the complex but battle-tested community system Slash. Give users the power to build friend and foe networks, moderate one another's comments and more.

Last month, we looked at the installation and basic administration of Slash, the open-source Weblog and community system that powers the popular Slashdot site, among others. Slash, which is distributed under the GNU Public License, takes advantage of Perl, mod_perl and Apache.

Creating a Journal

Slash uses the term journals for its Weblogs. Each user on the system can keep his or her own journal; this functionality is available by clicking Journal on the You menu, which typically is displayed along the left side of the screen. This invokes journal.pl, which is located inside of your site's Slash directory. On my computer, named chaim-weizmann, I found journal.pl in /usr/local/slash/site/chaim-weizmann/htdocs/journal.pl. The code is easier to read than I imagined, but even if you are an experienced Perl hacker, you should find that a great many functions are centralized and customized for the Slash environment. That said, changing Slash does not appear to be terribly difficult, if you are interested in tinkering with it.

The first time you click on the Journal link, you see a screen that looks like the screenshot pictured in Figure 1. A message there indicates you have not created any journal entries, and several links offer you the chance to write in your journal or edit existing entries.

Figure 1. A New Slash Journal without Any Entries

Let's create a new journal entry by clicking on the Write in Journal link. This opens a new page, shown in Figure 2. We enter a subject, a topic (a combination of the global list of topics, along with user journal), an indication of whether you want to allow others to comment on your journal entry and the entry itself.

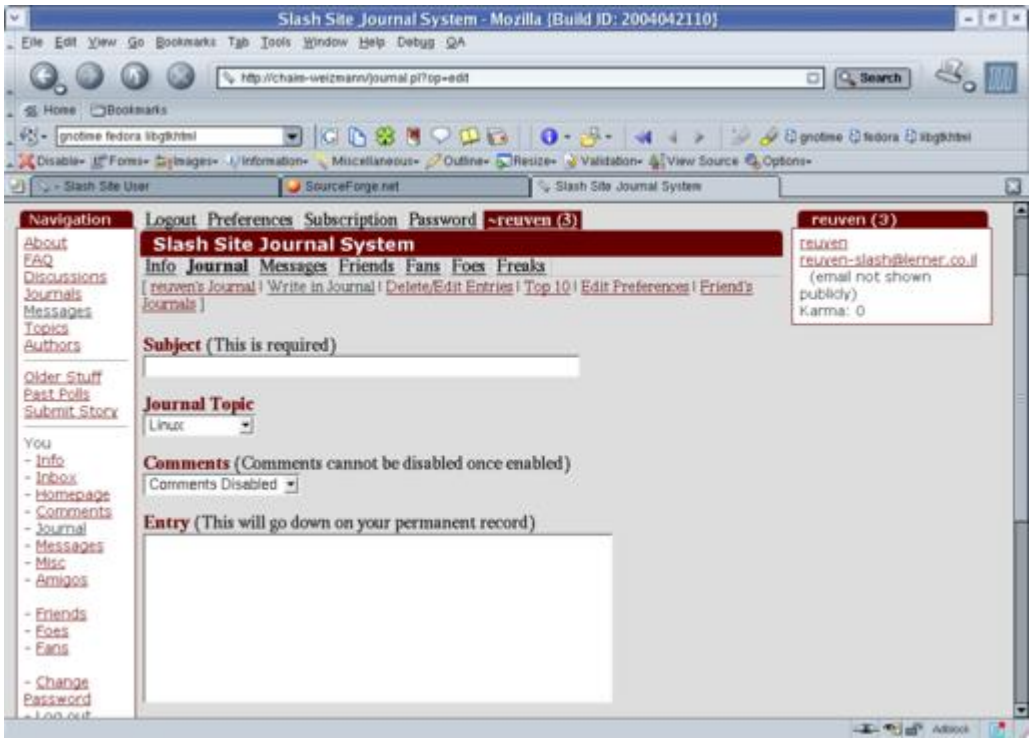


Figure 2. The journal entry page is where you compose new text for your journal.

When you have finished writing a journal entry, you can click the Preview button, which allows you to look at your entry before posting it. This seems a bit patronizing to me; although I understand it is useful and important for people to proofread and double-check their work before submitting it, at times I want to move ahead and prefer not to preview my work.

Next to the Preview button is a selection list that allows you to indicate how your journal entry should be formatted. The default, HTML format, allows you to stick HTML tags into your journal, so that you can create boldface and italic text. Of course, HTML does not differentiate between types of whitespace, which means choosing this formatting method requires you to separate paragraphs with <p> tags. It also means you can enter a literal < or > character only by using the appropriate HTML entity, < or &rt;.

The extrans formatting option would have been my preference, if I had known what it was from the beginning: extrans assumes that every character should be taken literally and converts multiple newline characters into HTML paragraph breaks. I realize that the option says “HTML tags to text”, but that seems less important than the fact that paragraph separations are preserved in the final copy.

Once you have previewed your entry at least once, a Save button appears between the Preview button and formatting selection list. You can continue to modify and preview your journal entry, or you can save it and make it viewable by everyone else by clicking on the Add button). Indeed, anyone can view the journal I created on chaim-weizmann by pointing their browsers to the URL chaim-weizmann/~reuven/journal.

Comments

As often is the case with other Weblog and journal software, Slash makes it possible and easy to solicit comments from other users. By default, this option is off, and the instructions indicate clearly and repeatedly that turning comments on means they remain on forever.

This option is set for each individual journal entry; some can allow comments and others can forbid them, as users see fit. You can change the default setting by clicking the Edit Preferences link at the top of the journal page and then selecting comments disabled or comments enabled, as appropriate. Because you are setting only a default value, it has no effect on already existing journal entries and comments.

Adding comments to a journal entry that has enabled them is somewhat less than straightforward to the uninitiated. Each journal entry is followed by a menubar (Figure 3), which both controls the display of the discussion and

allows users to participate in it. I say that this is confusing because it is easy to miss the Reply button, which allows you to add to the discussion, and the rest of the menubar, which changes the way the discussion is viewed.

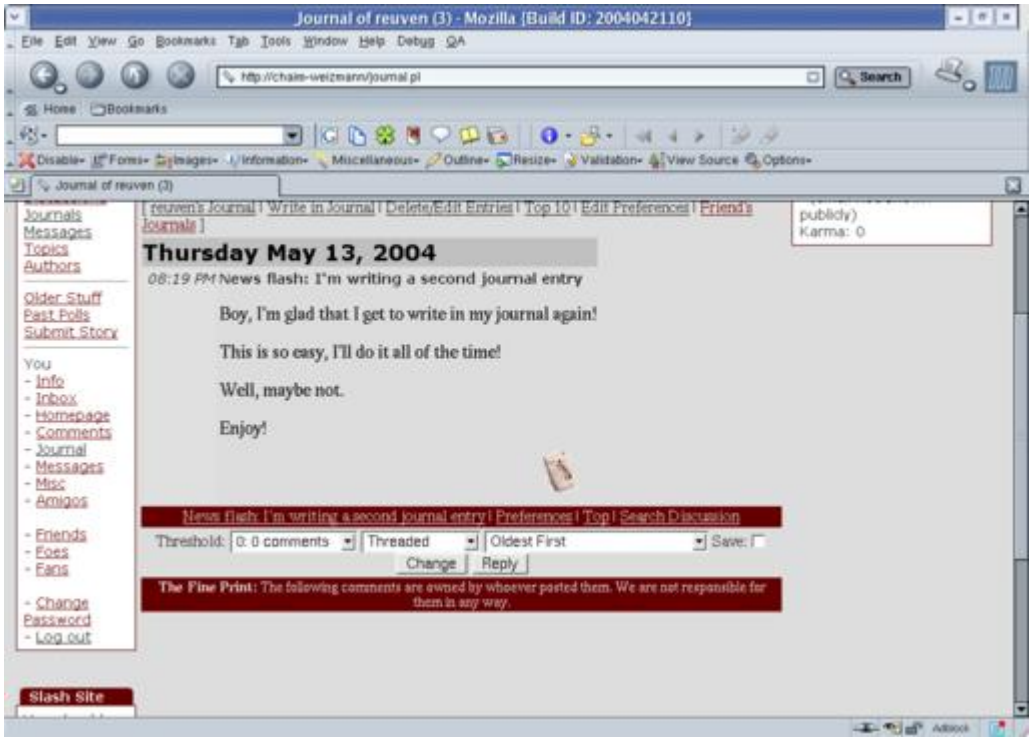


Figure 3. A Slashdot-style menu follows each journal entry.

Replying actually is slightly more complicated than this. To reply to the original posting, click on the Reply button that immediately follows the article. But, if you want to reply to a comment, thus creating a threaded discussion, you instead click on Reply to This link, which appears immediately beneath each comment. This structure makes logical sense, but I must admit that even after years of following and participating in discussions on Slash-powered sites, it took me some time to find and understand the distinction between the two methods.

Interface aside, adding a comment is identical to adding a new posting, except you cannot restrict people from commenting on what you have written. Enter a subject and the text of the comment, indicate the formatting and then either preview or add your comment. Slash allows you to post comments as an anonymous user, known as Anonymous Coward, by checking the post anonymously box next to the entry. However, many administrators have configured their systems to forbid such anonymous postings on the assumption that anonymity reduces accountability.

Finally, the display settings make it possible to view a discussion in any of several ways. The threshold setting allows you to selectively view comments, based on scores assigned by other members of the site's community. Moderation and the related meta-moderation feature can be activated by the

site's administrator, and they allow community members to determine which comments deserve the most attention.

The display setting changes the way in which threaded discussions are shown. I always have preferred to see such discussions in nested format, meaning that responses always are visible, indented somewhat from their parent. By default, Slash sites show the comments in threaded mode, which requires that you explicitly ask to view a comment before it is visible.

Finally, you can ask to see the comments in various orders. Journal entries always are displayed, Weblog-style, starting with the newest entry and ending with the oldest one. Comments to these journal articles, by contrast, normally are displayed in chronological order, with the oldest comment at the top. Therefore, keeping up with a discussion over time requires scrolling down to the bottom of the screen.

Journal Communities

From what we have seen so far, Slash seems to provide a simple way for many users to create and maintain their own journals. However, there is no real interaction among these users or their journals; everyone is insulated from one another.

But Slash was written to promote on-line communities, and it comes with a number of features that promote collaboration and integration. To begin with, Slash keeps track of statistics across all of the journals in the system. By clicking on the Top 10 link in journal.pl (the main journal page), you can find out which journals were updated most recently, which people have written the most and which friends have written most.

The term “friends” I refer to here is not a synonym for community member. Rather, every user in a Slash system can categorize other users as friends and foes, creating an interesting web of interpersonal relationships similar to but distinct from such sites as Orkut and LinkedIn.

The easiest way to mark someone as a friend or foe is to go to his or her home page, typically ~username. Thus on my system, anyone can go to my home page with the URL chaim-weizmann/~reuven. Next to the person's user name is an icon indicating whether he or she currently is a friend (smiley face), foe (red sad face) or neutral (the default, with what appear to be sunglasses and an odd smirk). Clicking on this icon allows you to change your relationship with this other person.

One big difference between Slash and various other personal networking and community Web sites is the fact that such relationships are public. Any user on

a Slash site can find out who my friends and foes are. Although this probably stops people from marking others as foes, because of the public embarrassment and fallout that might result, it does mean that Slash can create fascinating personal networks and relationship combinations. You not only see a list of someone's friends, but the person's friends' friends, as well.

Each of these relationships is one-sided; A can be B's friend, but B can be A's foe. When you go to someone's home page, you can look not only at the person's friends and foes, but also at his or her fans (others who have marked this person as a friend) and freaks (others who have marked this person as a foe).

The biggest practical advantage to setting up a list of friends is the fact that Slash keeps track of their journals and journal updates for you. Clicking on the Friend's Journals link at the top of your home page brings up a list of your friends with journals. This is the Slash equivalent of bookmarks or of an RSS news aggregator. Putting people on your list of friends means you easily can keep up with the journals that your friends have written.

Should You Use Slash?

I have looked at Slash several times over the years, and each time I came away fairly unimpressed. The code seemed hard to understand, the user interface was ugly and the functionality seemed limited. Slash-based sites remain relatively ugly, although this now is changeable, thanks to its use of the Template Toolkit. The functionality still is quite limited when you compare it with other community infrastructures and toolkits, such as Xoops and OpenACS.

But, Slash was not designed for broad needs; rather, it tries to implement a limited set of functionality and to do it well. In that regard, they really have succeeded. use.perl.org is a great example of such a site, which both distributes news articles and allows users to keep their own journals. If you want to provide limited news and announcements, while making it possible for large numbers of users to keep and comment on journals, Slash might be a good way to go.

Further, I must admit that the code has improved dramatically over the years; it now is possible to understand what is happening and even to modify or add functionality if you are an experienced Web/database hacker. Granted, Slash has many convenient functions that require something of a learning curve before you can jump in and make changes, but this is true of all Web/database toolkits, so it's unfair to say that Slash is different in this regard.

My main criticism of Slash, aside from issues having to do with distribution versions (which remain in CVS) and documentation, is the lack of a standard system for adding new functionality in the way that Xoops, OpenACS and Zope have done through their various modules and packages.

Conclusion

Slash, like much open-source software, is powerful, scalable, difficult for newcomers to install and poorly documented. Unlike many other packages, it also focuses on depth rather than breadth, providing more features than many other toolkits, at the expense of extensibility and generalizability. And, if your site is even beginning to approach the number of users or visitors that Slashdot attracts, you would be wise to consider using it.

Resources for this article: /article/7607.

Reuven M. Lerner, a longtime Web/database consultant and developer, is now a first-year graduate student in the Learning Sciences program at Northwestern University. His Weblog is at altneuland.lerner.co.il, and you can reach him at reuven@lerner.co.il.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Kernel Korner

Storage Improvements for 2.6 and 2.7

Paul E. McKenney

Issue #124, August 2004

The Linux 2.6 kernel has improved Linux's storage capabilities with advances such as the anticipatory I/O scheduler and support for storage arrays and distributed filesystems.

Storage has changed rapidly during the last decade. Prior to that, server-class disks were proprietary in all senses of the word. They used proprietary protocols, they generally were sold by the server vendor and a given server generally owned its disks, with shared-disk systems being few and far between.

When SCSI moved up from PCs to mid-range servers in the mid 1990s, things opened up a bit. The SCSI standard permitted multiple initiators (servers) to share targets (disks). If you carefully chose compatible SCSI components and did a lot of stress testing, you could build a shared SCSI disk cluster. Many such clusters were used in datacenter production in the 1990s, and some persist today.

One also had to be careful not to exceed the 25-meter SCSI-bus length limit, particularly when building three- and four-node clusters. Of course, the penalty for exceeding the length is not a deterministic oops but flaky disk I/O. This limitation required that disks be interspersed among the servers.

The advent of FibreChannel (FC) in the mid-to-late 1990s improved this situation considerably. Although compatibility was and to some extent still is a problem, the multi-kilometer FC lengths greatly simplified datacenter layout. In addition, most of the FC-connected RAID arrays export logical units (LUNs) that can, for example, be striped or mirrored across the underlying physical disks, simplifying storage administration. Furthermore, FC RAID arrays provide LUN masking and FC switches provide zoning, both of which allow controlled disk sharing. Figure 1 illustrates an example in which server A is permitted to access

disks 1 and 2 and server B is permitted to access disks 2 and 3. Disks 1 and 3 are private, while disk 2 is shared, with the zones indicated by the grey rectangles.

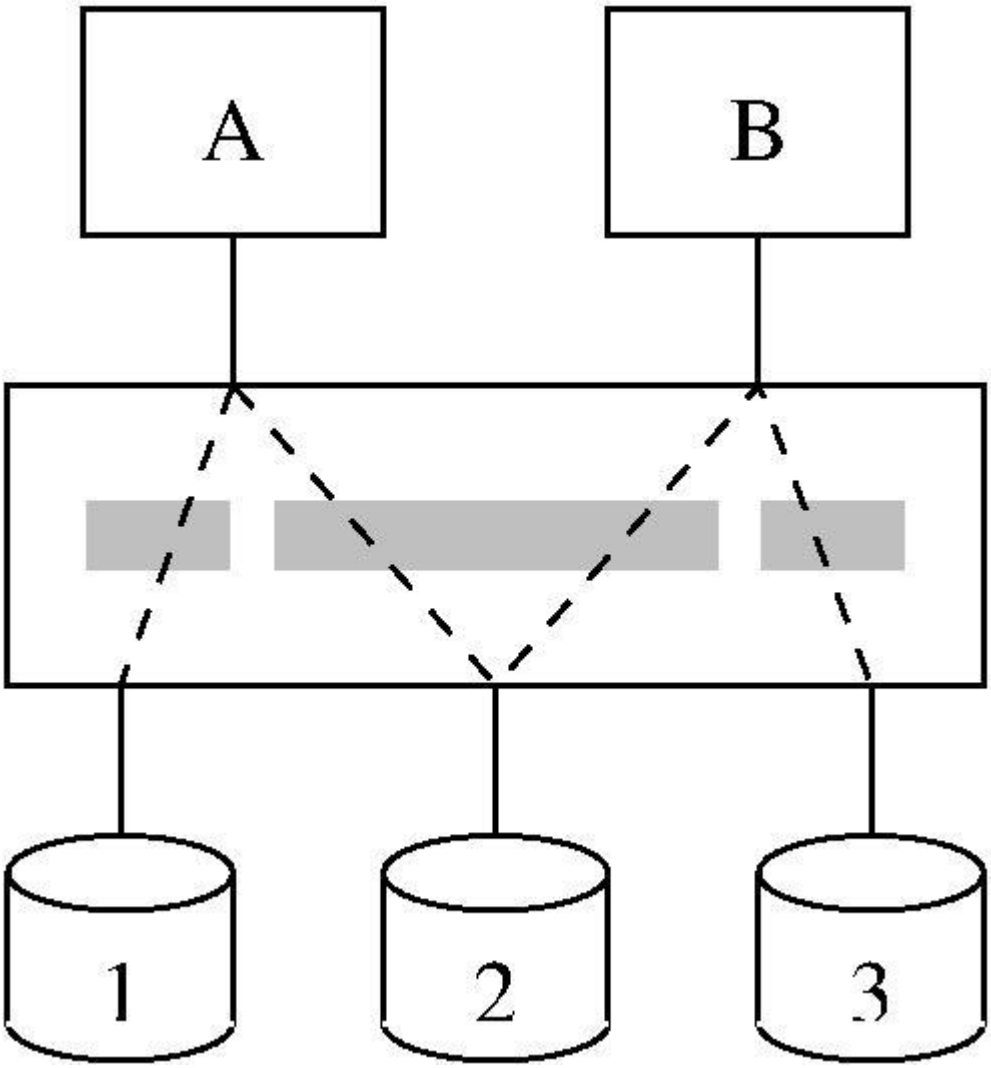


Figure 1. FibreChannel allows for LUN masking and zoning. Server A can access disks 1 and 2, and server B can access 2 and 3.

This controlled sharing makes block-structured centralized storage much more attractive. This in turn permits distributed filesystems to provide the same semantics as do local filesystems, while still providing reasonable performance.

Block-Structured Centralized Storage

Modern inexpensive disks and servers have reduced greatly the cost of large server farms. Properly backing up each server can be time consuming, however, and keeping up with disk failures can be a challenge. The need for backup motivates centralizing data, so that disks physically located on each server need not be backed up. Backups then can be performed at the central location.

The centralized data might be stored on an NFS server. This is a reasonable approach, one that is useful in many cases, especially as NFS v4 goes mainstream. However, servers sometimes need direct block-level access to their data:

1. A given server may need a specific filesystem's features, such as ACLs, extended attributes or logging.
2. A particular application may need better performance or robustness than protocols such as NFS can provide.
3. Some applications may require local filesystem semantics.
4. In some cases, it may be easier to migrate from local disks to RAID arrays.

However, the 2.4 Linux kernel presents some challenges in working with large RAID arrays, including storage reconfiguration, multipath I/O, support for large LUNs and support for large numbers of LUNs. The 2.6 kernel promises to help in many of these areas, although there are some areas of improvement left for the 2.7 development effort.

Storage Reconfiguration

Because most RAID arrays allow LUNs to be created, removed and resized dynamically, it is important that the Linux kernel to react to these actions, preferably without a reboot. The Linux 2.6 kernel permits this by way of the /sys filesystem, which replaced the earlier /proc interfaces. For example, the following command causes the kernel to forget about the LUN on busid 3, channel 0, target 7 and LUN 1:

```
echo "1" > \
/sys/class/scsi_host/host3/device/3:0:7:1/delete
```

The busid of 3 is redundant with the 3 in host3. This format also is used, however, in contexts where the busid is required, such as in /sys/bus/scsi/devices.

To later restore only that particular LUN, execute:

```
echo "0 7 1" > /sys/class/scsi_host/host3/scan
```

To resize this same LUN, use:

```
echo 1 > /sys/bus/scsi/devices/3:0:7:1/rescan
```

To scan all channels, targets and LUNs, try:

```
echo "- - -" > /sys/class/scsi_host/host3/scan
```

and to scan only one particular target, enter:

```
echo "0 7 -" > /sys/class/scsi_host/host3/scan
```

Although this design is not particularly user-friendly, it works fine for automated tools, which can make use of the libsys library and the systool utility.

Multipath I/O

One of FC's strengths is it permits redundant paths between servers and RAID arrays, which can allow failing FC devices to be removed and replaced without server applications even noticing that anything happened. However, this is possible only if the server has a robust implementation of multipath I/O.

One certainly cannot complain about a shortage of multipath I/O implementations for the Linux 2.4 kernel. The reality is quite the opposite, as there are implementations in the SCSI mid-layer, in device drivers, in the md driver and in the LVM layer.

In fact, too many I/O implementations in 2.6 can make it difficult or even impossible to attach different types of RAID arrays to the same server. The Linux kernel needs a single multipath I/O implementation that accommodates all multipath-capable devices. Ideally, such an implementation continuously would monitor all possible paths and determine automatically when a failed piece of FC equipment had been repaired. Hopefully, the ongoing work on device-mapper (dm) multipath target will solve these problems.

Support for LUNs

Some RAID arrays allow extremely large LUNs to be created from the concatenation of many disks. The Linux 2.6 kernel includes a CONFIG_LBD parameter that accommodates multiterabyte LUNs.

In order to run large databases and associated applications on Linux, large numbers of LUNs are required. Theoretically, one could use a smaller number of large LUNs, but there are a number of problems with this approach:

1. Many storage devices place limits on LUN size.
2. Disk-failure recovery takes longer on larger LUNs, making it more likely that a second disk will fail before recovery completes. This secondary failure would mean unrecoverable data loss.

3. Storage administration is much easier if most of the LUNs are of a fixed size and thus interchangeable. Overly large LUNs waste storage.
4. Large LUNs can require longer backup windows, and the added downtime may be more than users of mission-critical applications are willing to put up with.

The size of the kernel's dev_t increased from 16 bits to 32 bits, which permits i386 builds of the 2.6 kernel to support 4,096 LUNs, though at the time of this writing, one additional patch still is waiting to move from James Bottomley's tree into the main tree. Once this patch is integrated, 64-bit CPUs will be able to support up to 32,768 LUNs, as should i386 kernels built with a 4G/4G split and/or Maneesh Soni's sysfs/dcache patches. Of course, 64-bit x86 processors, such as AMD64 and the 64-bit ia32e from Intel, should help put 32-bit limitations out of their misery.

Distributed Filesystems

Easy access to large RAID arrays from multiple servers over high-speed storage area networks (SANs) makes distributed filesystems much more interesting and useful. Perhaps not coincidentally, a number of open-source distributed filesystem are under development, including Lustre, OpenGFS and the client portion of IBM's SAN Filesystem. In addition, a number of proprietary distributed filesystems are available, including SGI's CXFS and IBM's GPFS. All of these distributed filesystems offer local filesystem semantics.

In contrast, older distributed filesystems, such as NFS, AFS and DFS, offer restricted semantics in order to conserve network bandwidth. For example, if a pair of AFS clients both write to the same file at the same time, the last client to close the file wins—the other client's changes are lost. This difference is illustrated in the following sequence of events:

1. Client A opens a file.
2. Client B opens the same file.
3. Client A writes all As to the first block of the file.
4. Client B writes all Bs to the first block of the file.
5. Client B writes all Bs to the second block of the file.
6. Client A writes all As to the second block of the file.
7. Client A closes the file.
8. Client B closes the file.

With local-filesystem semantics, the first block of the file contain all Bs and the second block all As. With last-close semantics, both blocks contain all Bs.

This difference in semantics might surprise applications designed to run on local filesystems, but it greatly reduces the amount of communication required between the two clients. With AFS last-close semantics, the two clients need to communicate only when opening and closing. With strict local semantics, however, they may need to communicate on each write.

It turns out that a surprisingly large fraction of existing applications tolerate the difference in semantics. As local networks become faster and cheaper, however, there is less reason to stray from local filesystem semantics. After all, a distributed filesystem offering the exact same semantics as a local filesystem can run any application that runs on the local filesystem. Distributed filesystems that stray from local filesystem semantics, on the other hand, may or may not do so. So, unless you are distributing your filesystem across a wide-area network, the extra bandwidth seems a small price to pay for full compatibility.

The Linux 2.4 kernel was not entirely friendly to distributed filesystems. Among other things, it lacked an API for invalidating pages from an mmap()ed file and an efficient way of protecting processes from oom_kill(). It also lacked correct handling for NFS lock requests made to two different servers exporting the same distributed filesystem.

Invalidating Pages

Suppose that two processes on the same system mmap() the same file. Each sees a coherent view of the other's memory writes in real time. If a distributed filesystem is to provide local semantics faithfully, it needs to combine coherently the memory writes of processes mmap()ing the file from different nodes. These processes cannot have write access simultaneously to the file's pages, because there then would be no reasonable way to combine the changes.

The usual solution to this problem is to make the nodes' MMUs do the dirty work using so-called distributed shared memory. The idea is only one of the nodes allows writes at any given time. Of course, this currently means that only one node may have any sort of access to a given page of a given file at a time, because a page can be promoted from read-only to writable without the underlying filesystem having a say in the matter.

When some other node's process takes a page fault, say, at offset 0x1234 relative to the beginning of the file, it must send a message to the node that currently has the writable copy. That node must remove the page from any user processes that have it mmap()ed. In the 2.4 kernel, the distributed filesystem must reach into the bowels of the VM system to accomplish this, but the 2.6 kernel provides an API, which the second node may use as follows:

```
invalidate_mmap_range(inode->mapping, 0x1234, 0x4);
```

The contents of the page then may be shipped to the first node, which can map it into the address space of the faulting process. Readers familiar with CPU architecture should recognize the similarity of this step to cache-coherence protocols. This process is quite slow, however, as data must be moved over some sort of network in page-sized chunks. It also may need to be written to disk along the way.

Challenges remaining in the 2.6 kernel include permitting processes on multiple nodes to map efficiently a given page of a given file as read-only, which requires that the filesystem be informed of write attempts to read-only mappings. In addition, the 2.6 kernel also must permit the filesystem to determine efficiently which pages have been ejected by the VM system. This allows the distributed filesystem to do a better job of figuring out which pages to evict from memory, as evicting pages no longer mapped by any user process is a reasonable heuristic—if you efficiently can work out which pages those are.

NFS Lock Requests

The current implementation of NFS lockd uses a per-server lock-state database. This works quite well when exporting a local filesystem, because the locking state is maintained in RAM. However, if NFS is used to export the same distributed filesystem from two different nodes, we end up with the situation shown in Figure 2. Both nodes, running independent copies of lockd, could hand out the same lock to two different NFS clients. Needless to say, this sort of thing could reduce your application's uptime.

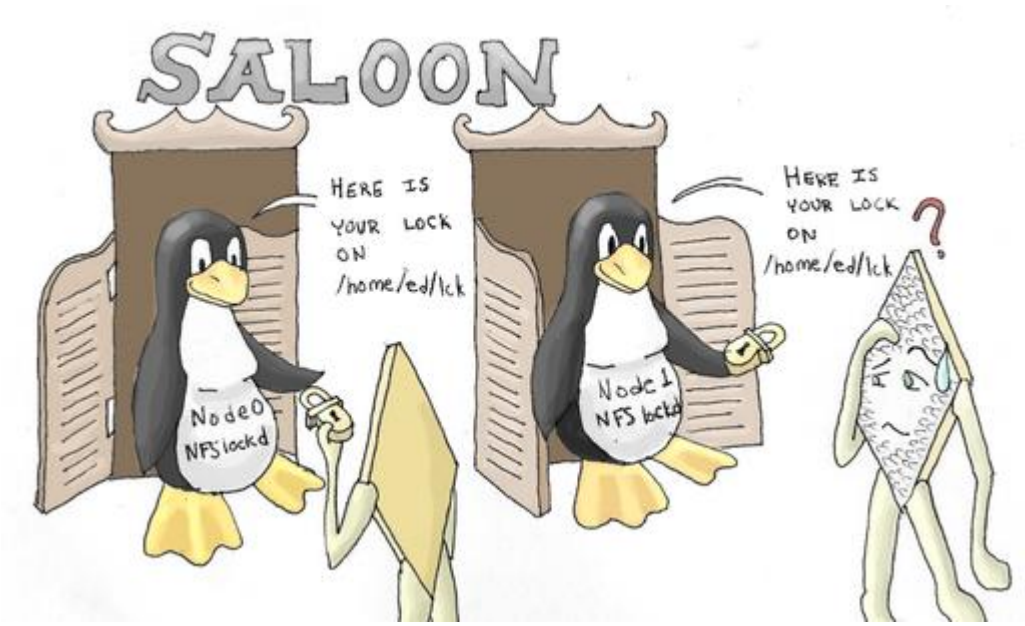


Figure 2. One lock, two clients, big trouble.

One straightforward way of fixing this is to have lockd acquire a lock against the underlying filesystem, permitting the distributed filesystem to arbitrate concurrent NFS lock requests correctly. However, lockd is single-threaded, so if the distributed filesystem were to block while evaluating the request from lockd, NFS locking would be stalled. And distributed filesystems plausibly might block for extended periods of time while recovering from node failures, retransmitting due to lost messages and so on.

A way to handle this is to use multithread lockd. Doing so adds complexity, though, because the different threads of lockd must coordinate in order to avoid handing out the same lock to two different clients at the same time. In addition, there is the question of how many threads should be provided.

Nonetheless, patches exist for these two approaches, and they have seen some use. Other possible approaches include using the 2.6 kernel's generic work queues instead of threads or requiring the underlying filesystem to respond immediately but permitting it to say “I don't know, but will tell you as soon as I find out”. This latter approach would allow filesystems time to sort out their locks while avoiding stalling lockd.

Don't Kill the Garbage Collector

Some distributed filesystems use special threads whose job it is to free up memory containing cached file state no longer in use, similar to the manner in which bdflush writes out dirty blocks. Clearly, killing such a thread is somewhat counterproductive, so such threads should be exempt from the out-of-memory killer oom_kill().

The trick in the 2.6 kernel is to set the CAP_SYS_RAWIO and the CAP_SYS_ADMIN capabilities by using the following:

```
cap_raise(current->cap_effective,
          CAP_SYS_ADMIN|CAP_SYS_RAWIO);
```

Here, current indicates the currently running thread. This causes oom_kill() to avoid this thread, if it does choose it, to use SIGTERM rather than SIGKILL. The thread may catch or ignore SIGTERM, in which case oom_kill() marks the thread so as to refrain from killing it again.

Future Trends

It appears that storage systems will continue to change. The fact that LAN gear is much less expensive than SAN gear augurs well for iSCSI, which runs the SCSI protocol over TCP. However, widespread use of iSCSI raises some security issues, because failing to disable IP forwarding could let someone hack your storage system. Some believe that serial ATA (SATA) is destined to replace SCSI

in much the same way that SCSI itself replaced proprietary disk-interface protocols. Others believe that RAID arrays will be replaced by object stores or object-store targets, and in fact there is an emerging standard for such devices. Either way, interfacing to storage systems will continue to be challenging and exciting.

Acknowledgements

I owe thanks to the Linux community but especially to Daniel Phillips and Hugh Dickins for most excellent discussions and to Mike Anderson and Badari Pulavarty for their explanations of recent 2.6 kernel capabilities and their review of this paper. I also am grateful to Bruce Allan and Trond Myklebust for their thoughts on resolving the NFS lockd issue.

Legal Statement

This work represents the view of the author and does not necessarily represent the view of IBM.

Paul E. McKenney is a distinguished engineer at IBM and has worked on SMP and NUMA algorithms for longer than he cares to admit. Prior to that, he worked on packet-radio and Internet protocols, but long before the Internet became popular. His hobbies include running and the usual house-wife-and-kids habit.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Cooking with Linux

The Ultimate Cooking Box

Marcel Gagné

Issue #124, August 2004

Organize your recipes with or without a MySQL database server, whichever pleases your palate.

It isn't that strange, *mon ami*. Every year at this time, we talk about the ultimate Linux box, François. We try to outdo one another in terms of graphics, speed, memory, disk space and so on. The ultimate, invariably, is also a question of the latest and greatest. The trouble I have, François, is that as new as all this hardware extravaganza is, it is all the same thing. Then it occurred to me that if my Linux box could help out in the kitchen, that truly would be something new.

Non, François, you do not have to worry, your job is safe. Besides, this is exactly the sort of thing that might help make your job easier. In fact, I would not be surprised if many of our guests find that using Linux to help when it comes to cooking is simply natural. Speaking of guests, I see they have arrived. Welcome, *mes amis*, to *Chez Marcel*. *Vite*, François! Go to the cellar and bring back the 1997 Brunello di Montalcino, *immédiatement*!

While my faithful waiter fetches the wine, I should tell you about today's menu. In honor of this issue's theme of the ultimate Linux box, I've located a few programs to make your Linux box cook. I don't mean screaming performance, *mes amis*, but cooking with food and wine.

Looking for a way to get my Linux box to help out in the kitchen led me to **Krecipes** (see the on-line Resources section). Unai Garro, Jason Kivlighn and Bosselut Cyril have been putting together a nice open-source package that makes creating and maintaining your own list of favorite recipes a breeze. Krecipes lets you create and edit your own recipes as well as import them from popular exchange formats, including RecipeML, MasterCook, Meal-Master and others. It saves its own recipes in KreML (Krecipes XML format). You also can

maintain categories, track calories (fiber, fat and so on), create shopping lists based on selected meals and more.

Ah, François, you have returned. Please, pour for our guests.

Krecipes stores its recipes in a database, so you need to have either MySQL or SQLite (see Resources) installed before compiling and using the software. SQLite is a small program-embeddable database that requires somewhat less overhead and administration than MySQL, but it's not as full-featured or powerful. For an application such as this, users may find it to be an attractive option. The beauty of SQLite is no database process needs to be running on the system in order to take advantage of SQL database capabilities, storage and access.

If you have both MySQL and SQLite on your system, support is compiled for both at build time. Speaking of building, this is a classic case of the extract and build five-step:

```
tar -xzvf krecipes_alpha_0.4.1.tar.gz
cd krecipes
./configure --prefix=/usr
make
su -c "make install"
```

During your first time with Krecipes (the program name is krecipes), a wizard guides you through some basic setup processes, including a choice of database in which to store the information. If you have compiled the program with both MySQL and SQLite, you can choose either. Because we've already covered many programs that use a MySQL back end in this restaurant, I thought it might be nice to go with SQLite. I made that my choice and clicked Next.

Once you have made your selection, Krecipes offers to populate the program with some sample recipes. Make sure you click on this check box before continuing with the setup so you wind up with a few examples to help familiarize yourself with the software.

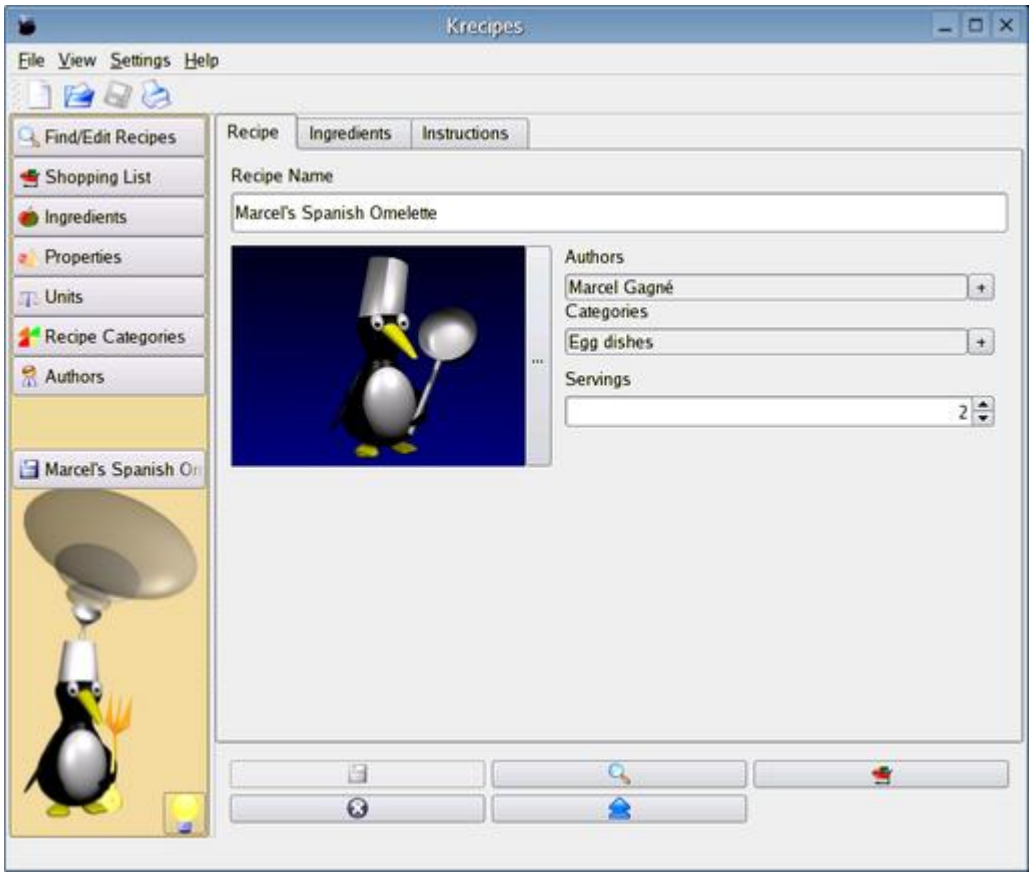


Figure 1. Marcel enters the details of his famous Spanish Omelette using Krecipes.

To create a new recipe, click File on the menubar and select New or simply click the new recipe button in the top left-hand side of the icon bar. The recipe dialog has three tabs. One is for the recipe basics—the name, the author, which category it should be filed under (you can create new categories) and how many people the dish serves. See Figure 1 for an example. The other two tabs are for the ingredients list and the description. In all cases, you can save your work at any time or return to it later for updates.

If you would rather take advantage of the many thousands of recipes available on the Internet in Meal-Master and RecipeML format, it's easy to import them. All this talk of food, *mes amis*, just reinforces the importance of a well-stocked wine cellar.

Another great recipe manager you may want to investigate is Douglas Squirrel's **LargoRecipes**. LargoRecipes (named after the author's dog) lets you manage recipes, share them with friends (through Web pages), create shopping lists, build meal plans and more. You also can import Meal-Master and RecipeML format recipes. Have a look at Figure 2 for a sample of LargoRecipes in action.

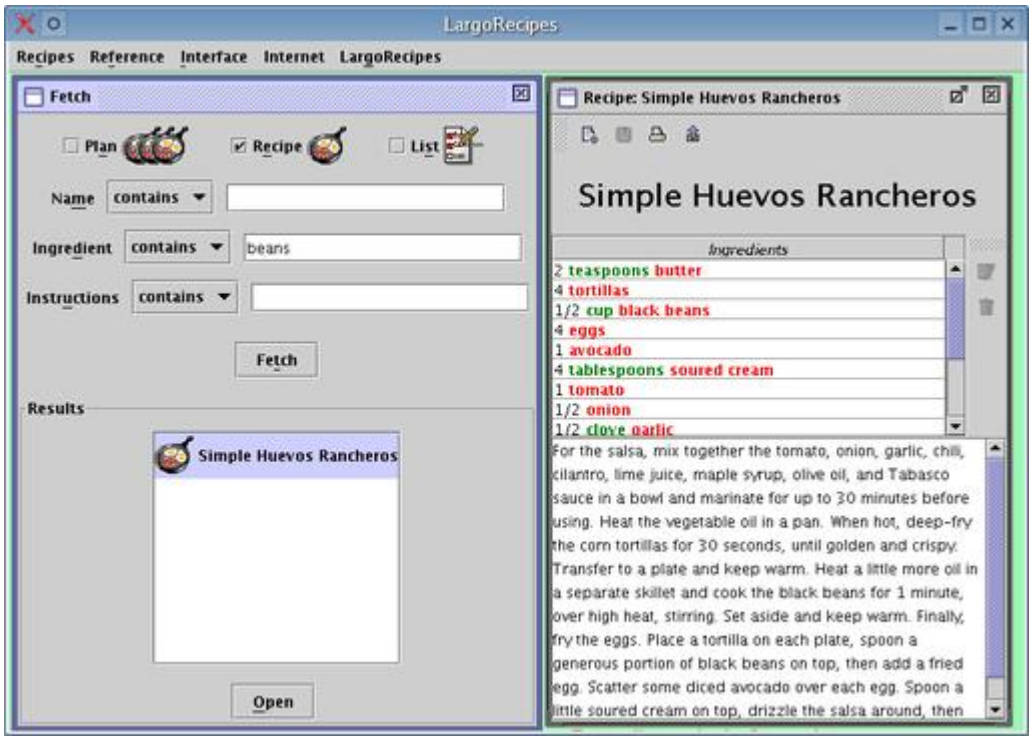


Figure 2. Dishing Up Some Huevos Rancheros with LargoRecipes

To install LargoRecipes, you need at least Java 1.4, which should tell you that no compiling is involved in the installation. Download and save two files from the LargoRecipes Web site (see Resources). The first is the largorecipes distribution; I'll get to the second file shortly. At the time of this writing, version 0.9.2.1 was available. To install the package, save the bundle to a directory of your choice—I created a Largo directory in my home directory—and execute the following command:

```
cd ~/Largo
java -jar largorecipes-0.9.2.1.jar
```

That also is the command you use to run it on subsequent uses. On your first run, an installation dialog appears. All of the supporting data files and directories are created from where you run the installation. One of those directories is called demo. This is where you save the second file you download, the LargoRecipes demo file. It also is available from the LargoRecipes Web site's download page.

To activate the sample recipes, for this session only, click LargoRecipes on the menubar and select Demonstration. If you would rather skip this and start importing recipes, consider checking out the LargoRecipes RecipeML archive. There are 10,000 recipes in zipped bundles available on the site; look for the link on the main page.

To share your recipes with others, LargoRecipes provides a Web page export function. Click Internet on the menubar and select Web Page, and a list of

available recipes appears in the right-hand window. Select the ones you want, and then press Add to add them to the export list. When you have made all your choices, choose a title for the page but don't click Go yet. You should see a check box labeled Include XML Download. Make sure that is checked to provide a link on each recipe's page so that visitors to your site can download RecipeML format copies of the recipes. They then can import those into their favorite recipe system.

For those of you who are curious, check out the Resources section for a link to the RecipeML format specification. It's always good to know how these things work, *non?* I've also provided a link to the Meal-Master Web site. There are plenty of links there from which you can find a huge number of recipes ready for import into your favorite package.

Mon Dieu, mes amis, closing time has arrived and all I have done is make you more hungry. Perhaps François will be kind enough to refill our glasses a final time. In the meantime, I shall bring out my famous baked double-butter brie with spicy mixed-berry coulis. With all these tempting tastes loaded on our Linux systems, appetizers certainly are in order. Until next time, *mes amis*, let us all drink to one another's health. *A votre santé Bon appétit!*

Resources for this article: /article/7608.

Marcel Gagné (mggagne@salmar.com) lives in Mississauga, Ontario. He is the author of *Moving to Linux: Kiss the Blue Screen of Death Goodbye!* (ISBN 0-321-15998-5) from Addison Wesley. His first book is the highly acclaimed *Linux System Administration: A User's Guide* (ISBN 0-201-71934-7). In real life, he is president of Salmar Consulting, Inc., a systems integration and network consulting firm.

Archive Index Issue Table of Contents

Advanced search

Linux for Suits

Missing Pieces

Doc Searls

Issue #124, August 2004

Would you take project advice from a bozo? Doc covers a new opportunity for open source in business.

Robert L. “r0ml” Lefkowitz (that's a zero, not an o...you figure it out) is a tall, smart, charismatic and good-humored man with a shaved head and a red clown nose in his pocket that he sometimes wears to explain, visually, that he's “just a bozo”. Except for the nose, he's not. Lefkowitz is an IT veteran who stepped out of the frying pan of Wall Street and into the fire of the telco business, specifically from Merrill Lynch to his current job as Chief Technical Architect and VP Information Technology for AT&T Wireless. He calls himself geek-in-chief.

r0ml is a longtime advocate of open source. But he's also the rare open-source advocate who tells you all the ways open source either is inadequate as a development methodology or is lacking the tools and solutions large IT organizations require. He makes it his job both to prod the Open Source community to produce the missing goods and to find new goods for possible adoption by his company. He likes when his own engineers already have found them.

During the July 2003 O'Reilly Open Source Convention, Lefkowitz gave a talk titled “Six Missing Open-Source Projects”. It was as much a primer on The Real World of IT as a call to action for the open-source programmers who packed the room. Relationship Management was one of his projects. “Relationships are more important to most companies than code”, he said, which is why they spend more money on marketing than on programming. He said there is plenty of CRM—customer resource management—software out there for big companies, but none yet from the open-source world. There also is none for following relationships between IT departments and development

communities. There were so many questions such a system could answer. Which industries have the highest patch-submission to running-copies ratio? What percentage of bug reports come from financial services firms? How about from other industries? Who's working on what, anyway? And for how long?

He also called for help with asset management, distributed cron, change management, messaging, single sign-on and source termination. He said we should work to create a definitive literature for problem domains to make it easier to integrate open-source systems. He called for open-source developers to start thinking about operations—accounting and financial systems, help desk automation, customer relations and marketing. “All the stuff businesses care about.”

During the Open Source Business Conference in San Francisco, March 2004, r0ml gave a talk titled “(More) Missing Open-Source Projects”. In it he proposed four criteria that must be met for open-source projects to take root:

- Everybody (for some definition of everybody) needs it or can use it.
- Those who use it (everybody, by definition above) want to improve it (for some definition of improve).
- Significant business value associated with the use of the software does not reside in the software itself.
- It's cool.

He opened his talk by holding up two CDs and explaining they were what was left of a telecommunications billing system called Flexcell, which was orphaned when AT&T Wireless ate Vanguard Wireless, the North Carolina company that wrote it. “How cool would it be if there were an open-source company in North Carolina?” r0ml quipped, tongue planted deeply in his cheek. “That would be fortuitous. Particularly if they were interested in enterprise systems.”

He went on to explain that telecommunications billing systems were terrific ways to show off micropayment chops. Phone billing is personalized, goes down to the second and keeps track of many variables all at once. “Very kewl”, he said. Then, the pitch:

> So we are interested in testing the hypothesis. How will you take something this cool and turn it into an open-source project. Would anybody actually be interested in working on that? We won't invest a heap of money. But if other people are willing to invest some time or energy or money, then certainly that would encourage us to work with them a little bit.
>
> So the offer I'm making, if anybody is interested, is doing the world's largest and coolest micropayment system, I've got the source code. (And) I do have

> authorization to look into how to open source such a thing. I'll be happy to take any licenses or whatnot. It's all open for discussion.

Then he pitched CRM: “Since I was so successful at convincing you that billing systems are really cool, I'd like to give it to you that CRM systems are really cool.” Next, he detailed a peer-to-peer CRM system he called Carester. He went on to pitch projects in visual programming, business process integration (BPI), messaging and business intelligence (data warehousing). “Other than 'kewlness', is there any other reason why open source doesn't tackle billing, CRM, BPI and business intelligence?” he asked. Is it scale? Performance? Lack of a market? “Maybe this just isn't a value network that open source can ensconce itself in. I don't know. I'm just an open-source guy and a bozo in IT.” Then he brought out the nose.

It was a downbeat talk—kind of a Swiss cheese treat where all they serve is the holes. Later in a telephone interview, I asked r0ml to name some examples of cases where open source was succeeding, even in the categories he had mentioned. He said:

> Take the data warehouse space. You're dealing with expensive, specialized hardware. Teradata NCR. Old style hardware-software bundles that do large specialized databases. Now they have a competitor, Netezza (netezza.com). When a query comes into a Netezza box, it hits a quad-CPU that's running Red Hat Linux and PostgreSQL. Since PostgresSQL has a BSD license, they hacked it to do all this funky parallelization, so they can run it on their platform, which is kind of a blade thing with these souped-up disks. The executive, if you will, that does all the dispatching to all the subnodes and collects all the responses and sends them back to make these special-purpose queries, is Linux-based. If I buy them, I'm buying special-purpose hardware that's able to come into the market to undercut the established players. And they can do that because they build their stuff up from open-source basics like Linux and PostgreSQL, which they can take and adapt to their needs. They're doing DIY: Do-It-Yourself. They're being smart and resourceful. I like that.

The DIY-IT environment, r0ml says, is a complicated place. And, it will never submit to simplistic DIY efforts, least of all those that limit their interest to open source:

> Most environments are mixed. In some mixed environments, especially those using large enterprise-class software, you have some projects that probably never will be handled by open source and probably never come from anybody other than a big vendor....Here at AT&T Wireless, none of the large

> enterprise-class software we're using is available in open source, outside of databases. Our billing system uses Siebel on Sun. It also runs on HP-UX, AIX or Windows. But not Linux. Our billing system runs on something other than Linux. We also use Vitria business ware, which just added Linux support.

On the other hand, he said, “There are two classes of vendors in this space now: those that say they'll deliver products on Linux any day now and those that are thinking about whether there's a market. Meanwhile, there's a market.” In other words, it's catch-up time.

Meanwhile, your bread-and-butter enterprise systems are going to be provided by vendors, not by customers. Sometimes a vendor drives things. For example, The Wall Street Linux Roundtable was sponsored by Intel. There were representatives of various vendors including Reuters, which does a lot of business on Wall Street. “When you have representatives from Deutche Bank, Morgan Stanley, Merrill Lynch, Goldman Sachs, all sitting in the same room saying, 'We want stuff on Linux,' the vendors are going to go back to the office and say 'We underestimated the size of the market.'”

Back at the Open Source Business Conference, I led a panel on DIY-IT that included r0ml, Kevin Foreman of RealNetworks, Win Coekaerts of Oracle and Ted Shelton of CallTrex. Everybody made interesting points, and each panelist said something important about what each kind of company contributes to the market ecosystem. Wim Coekaerts, for example, made this point about testing:

> It's not just about contributing source code. It's about having the resources in hardware. Big iron hardware that you run test suites on and you do stuff with. There is the communication that you have internally. That's how a big part of what we contribute is testing. We have to do that for our customers. They expect a product to be reasonably well tested and to meet minimal criteria. Also, if you look at other UNIX operating systems or Windows, hardware and software come from the same company—Solaris on SPARC, AIX on PowerPC and so on. The companies themselves that built those systems have the hardware to do the testing. They've had tons of people dedicated to working with Oracle to run the database on their platforms. The testing involves a very small community. Now if you look at Linux, none of the distribution developers can afford the big iron hardware. But what we're doing now is working with the distributions to get the hardware vendors to participate in the testing. So we're trying to set up this virtual Linux test environment where...we'll deal with Dell, HP and other companies to say, “Here is a subset of tests you should run at the vendor side, in your early cycles. So that when they ship a hardware product, it has been pre-tested. In the past they didn't

> do that. You have the hardware specification list, you get a CD and good luck."

Ted Shelton offered an interesting answer to r0ml's question about what it would take for open source to appear in some of these large-scale enterprise categories:

> I've heard that things like CRM and ERP are not going to be addressed by open source. If you only look at open source as a very broad-based group of otherwise disconnected people coming together to do something, that's probably true. But sometimes you get a company that builds a big solution in a vertical market and decides to take it out in the open-source market, because they realize their profits are in selling hardware, not software. And they've now developed a big community that provides bug fixes and applications running on top of it. That's Asterisk (www.asterisk.org, the open-source Linux PBX).

Then Ted turned to Wim, and said, "So, when Oracle gets done buying PeopleSoft, why not open source all their stuff?" Ted did it for laughs, but you could see some heads in the audience nodding along.

Doc Searls (info@linuxjournal.com) is senior editor of *Linux Journal*. His monthly column is Linux for Suits and his biweekly newsletter is SuitWatch. He also presides over Doc Searls' IT Garage (garage.docsearls.com), a sister site to *Linux Journal* on the Web.

Archive Index Issue Table of Contents

Advanced search

EOF

Open Source Is for Pigs

Evan Leibovitch

Issue #124, August 2004

While the US tries to sandbag a UN conference on information technology, international open-source groups are making connections.

Do a search these days on the word “farm” on your favourite database of open-source software applications. You'll find that the result of your quest turns up, at best, a handful of projects related to server farms. This is fine for those of you who like to cultivate a roomful of servers. But what about conventional farms—the ones that are used to raise animals and grow crops? Without these kinds of real-world applications, open source is having a hard time realizing its potential within the developing world.

One would imagine that some of the greatest benefits from using open source would be realized in the world's poorest economies. The low cost, the ethos of sharing and community and building upon the work of others, make open-source methodologies perfectly suited to environments where the practice of self-help is not an option.

Unfortunately, a number of factors having nothing to do with logic are starting to cause new impediments to open-source growth in areas where it should be most embraced. Having failed to blunt the advance of open source through attacks on its quality, innovation, performance or support, opponents have resorted to the only weapon they have left (in abundance)—cash. Proprietary software vendors have been opening their chequebooks to developing countries and development agencies, in an attempt to blunt the adoption of open source that would occur in a truly neutral environment.

In February 2004, FOSSFA, the main African open-source group, issued a clarion call warning to governments of the long-term limitations of choice that could be associated with some of these donations. Even the $1 billion in gifts promised

by Microsoft alone certainly would come back to the company many-fold if it could maintain long-term dependence on its wares.

Meanwhile at the United Nations-sponsored World Summit on the Information Society (WSIS) conference in December 2003, original language that supported the use of open-source software systematically was taken out of ongoing drafts, until the final version merely recognized the existence of different software models. Although the negotiations that led to this conclusion were done behind closed doors, most sources suggested that the main arm-twisters against support of open source were the US delegation and the International Chamber of Commerce.

So how does the open-source world fight this blatant use of handouts and secret deals to maintain dependence? In this writer's opinion, the answer lies amongst the pigs and cows and vegetables mentioned at the beginning of this piece. Without the bankroll of open source's opponents, the community is fighting back using a tactic that no money can buy, with benefits right down on the farm.

The energy and enthusiasm of the community can—and must—go beyond commodity software, such as operating systems and utilities, and into job-specific applications that will take open source from the university lab to the village. Providing open-source systems and graphics software is of no value to someone who can't use the technology to sell more grain or fabric, or operate a school or hospital. Offering this kind of software allows for a turnkey open-source solution that can replace an entire proprietary system, not merely small pieces of it.

Think of a foundation with a structure similar to that of the Open Source Development Labs but creating software for dairy farms (as an example) instead of the kernel. Programmers get paid, costs of development are far below the cost of importing proprietary solutions, and the local community maintains a stake in both process and outcome. The idea of such a model was well received at the Idlelo conference earlier this year in Capetown, and some African governments already have expressed interest in supporting such foundations.

At the WSIS conference as well, community triumphed over money. While the politicians and policy makers were watering down the language of official documents, open-source advocates still were finding receptive ears amongst the delegates. The Linux Professional Institute (LPI) booth at the adjoining ICT-4D tradeshow was one of the busiest amongst hundreds of participants, drawing visitors from hours before the show started to hours after it ended each day. The LPI booth had 22 staff, coming from every continent, and

included three people from the Geneva Linux user groups. This was a perfect way to demonstrate the strength of a movement that was global but has roots everywhere. The message taken away—as well as the 5,000 Debian and SuSE CDs given out—offered a bright counterpoint to the dreary and generally pointless policy work coming from the conference.

It is here—in our community's strength of local grassroots, common ownership and self-reliance—that open-source technology makes the leap from being simply an alternative to a compelling and undeniable option. And, that particular strength can't be bought or sold, at any price.

Evan Leibovitch is President and cofounder of the Linux Professional Institute, creators of the most-popular Linux skills certification program worldwide. He has been involved with the community since 1995 and has written extensively about business and professionalism issues related to open source. He's based in Toronto, Canada.

Archive Index Issue Table of Contents

Advanced search

From the Editor

Money Talks

Don Marti

Issue #124, August 2004

And it says, “Hooray for Linux servers!” It's almost time for the billion-dollar-quarter party.

At the end of May, IDC released a thorough report on server sales in the first quarter of 2004. Linux servers came in at more than $900 million for the quarter, up 56.9% from last year.

At that growth rate, the best of any server OS, we're soon due for the first billion-dollar quarter for the Linux server business. So let's plan to celebrate it.

Even if servers aren't your bag, it's important to recognize milestones in this mature, successful area of the Linux business. The smart choices that made for Linux server success—including a commitment to GPL-licensed device drivers instead of problematic binary-only ones—will be a recipe for success in other fields too. Generic and hackable beats restrictive and specialized.

On the embedded hardware side, there's more good news—you can get a generic, hackable platform at your favorite computer store. Get a Linksys wireless access point with Linux onboard, and you can run your custom firewall, traffic control or any application you want on a platform that's well under a hundred dollars.

“Linux on Linksys Wi-Fi Routers” by James Ewing (page 50) gets you started in embedded Linux with hardware that fits your budget and beginner projects that get some real work done.

We've quietly made a change in our Resources sections at the ends of articles. Instead of a list of links, we point you at one jump page per article. Not only does that save you typing some long URLs, it also saves us some space in print,

and now we can check our logs to find out which articles got you interested enough to take the next step.

So, are you looking to reorganize your servers for easier management with serial consoles (page 66)? Are you planning to develop Linux support for a new USB device, and want to follow along step-by-step as the Linux USB master does it (page 36)? Or, are you planning to speed up your database application with Memcached (page 72)? We want to know.

So, congratulations to all the great people doing support, engineering, sales and everything else in the Linux server business. And whatever you use Linux for, you'll find something in this issue.

Don Marti is editor in chief of *Linux Journal*.

Archive Index Issue Table of Contents

Advanced search

Letters

Readers sound off.

We'll Believe It When We See It

The May 2004 issue of *Linux Journal* arrived yesterday, and I've completed reading the second part of Meng Weng Wong's SPF articles. It's obvious that a lot of work has gone into the authentication concept. Unfortunately, it's actually not required merely to be spam-free. All the effort for both you and others simply penalizes the good guys. Although authentication is a positive thing, it still solves the wrong problem, as Wong's article admits: “Remember, spammers can publish SPF too.” That's a lot of effort only to end up with that caveat.

More than two years ago, the company I work for started development of EVS Mail (the E-mail Validation Service) for a specific client who had requested it. Within months we were spam-free—we still are today. We don't have to care about how the rest of the Internet tries to cooperate to get rid of spam—our techniques work with how SMTP works, and that's the only cooperation we have to consider. Although SPF and others may be more effective than RBLs, I don't think they will be more successful. Our client never did sign on—go figure.

Fundamental pieces of EVS Mail are the automated white listing and challenge/response (authentication, or what we call validation). However, they are relatively minor items and do not suffer from the problems normally associated with those types of systems. This is due to ongoing development, of course. At this point, more than two years later, we have eliminated 100% of all baggage messages—they are completely unnecessary. Just today I put the finishing touches on a new process that will eliminate the generation of all explicit challenge messages (yet still accomplish the challenge function). For anyone who is the victim of a joe-job, we will now be able to guarantee that they will not get a challenge message from our servers. And, interestingly, our clients do not receive joe-job-related messages, so are not victims themselves.

The end result is that we are at the point where we can reduce bandwidth usage to less than what is used by either normal (unprotected) SMTP or by

other spam-control products and services. We also have extremely low resource usage and other benefits.

Why have you never heard of EVS Mail? Marketing, of course, which is directly related to money. We do have a superior service, but we are a small shop with a shoestring budget. We are seeking investment and will be able to produce a black box gateway running EVS Mail within three months of getting it.

—
Roger Walker

Reader Wheels

I just wanted to say how much I love your magazine. The articles are truly inspiring. And as a die-hard Linux fan, I wanted to show my devotion to Linux and open source. When temperatures in New York are at record lows, you need all the reliability you can get. I am planning to design and build various systems in this vehicle that are powered by the penguin.

—
Dave

Boo, Politics

Do you really think it wise to use as a demonstration in a magazine about Linux a personal Web site containing the author's personal views on such sensitive issues? [See “COREBlog”, *LJ*, April 2004.] I understand that by their very nature Weblogs are linked to opinion and comment, but there is plenty of scope for

opinion and comment within the realm of technology. Readers may disagree with you on your preference for Emacs over vi, but as far as I'm aware, nobody has died over this long-running dispute. In the future, please keep such personal Web sites out of articles about technology, or I may reach the opinion that your magazine has some other motive than just Linux evangelism.

—
Geraint Williams

Real Web sites sometimes have information that people feel strongly about. We'd rather keep our examples realistic than give new readers the impression that Linux users spend all their time bickering over Emacs vs. vi. —Ed.

Yay, Politics

I appreciated Doc Searls' article in the June 2004 issue of *Linux Journal* entitled “Hacking Democracy”. I also see a link between freedom, democracy and technology. I very much hope all our members of democracy can see that link and maintain it. It is definitely being attacked by those who hold greed as their creed as they march on—even attempting to redefine the word innovation.

—
Valden Longhurst

Hooray for Meng!

Meng Weng Wong is a terrific writer. I just finished his current article on SPF [*LJ*, May 2004]. He has a way of taking complicated, technical and potentially dry material and expressing it in a clear, conversational way. As far as I am concerned, he's the benchmark.

—
Jeff Jourard

Check our Web site for a follow-up from Meng on the new generation of SPF. —Ed.

More Success Stories Please

I would like to say thank you for an excellent magazine. I always look forward to my *LJ* coming in the mail. I typically read the entire magazine within the first couple of days. I recently read an article about Weather.com switching from Solaris/Websphere to Linux/Tomcat for delivering its Web information. The article also said they may be moving from Oracle to MySQL for their database. I

would like to see more articles or sidebars about companies that switched from proprietary to open-source platforms and the successes and failures they had. I liked your recent article about HEC in Canada [May 2004] and their new e-mail system. I especially liked the amount of detail given about the setup. I work as a Windows Network administrator, and it is nice to see the Linux alternatives that are available. I realize companies tend to be a little secretive about their networks, so it may not be possible to provide the information I am seeking.

—
Stephen Haywood

Another SPF User

I just wanted to thank you for the excellent articles on SPF published in *LJ* [see Meng Weng Wong's articles in the April and May 2004 issues]. Thanks to your wizard (spf.pobox.com/wizard.html), I could enhance and verify my setup in a couple of minutes. Excellent work!

—
Carlos Vidal

Processor Count Correction

The June 2004 issue of *LJ* has a minor error on page 12, *LJ* Index, #17. The number of Opterons to be used in the Dawning 4000A supercomputer is “More than 2,000”, not 800.

—
Charles N. Burns

Photo of the Month: /bin/cat

I ran across this old photo (SuSE 7.2) and thought you might like it for your magazine's Letters section. As you can see, my cats enjoy a new distribution of Linux just as much as I do—or at least the box!

—
Jeffrey K. Brown

Photo of the Month gets you a one-year extension to your subscription. Photos to info@linuxjournal.com. —Ed.

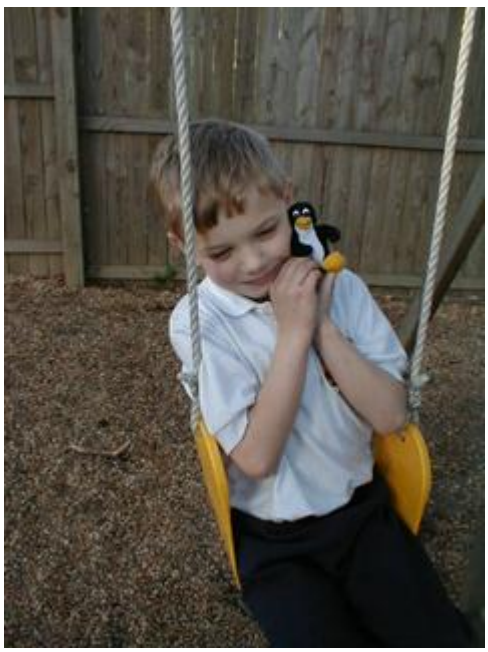
Little Box Puts Old Hardware on the Net

One of Linux's greatest assets is its ability to add value to legacy technology investments by connecting and interfacing with old equipment or software. Recently, I found a solution to add networking capabilities to an old telephone system. After seeing an advertisement for Cyclades in a previous issue of *Linux Journal*, I purchased the Linux-based Cyclades TS-100. Much like the way Linux can be used as an e-mail gateway to enhance a legacy mail server, I was able to add networking services to my old phone system.

—
nick marsh

Penguin Love

LJ is about the only computer magazine I pay for—LOVE IT! I've been a longtime fan of Linux, and now my kids are starting to use it. They love the Linux penguin and call him Pengi. We took a few snapshots a while back with one of them.



—
Wade Hampton

Archive Index Issue Table of Contents

Advanced search

UpFront

- diff -u: What's New in Kernel Development
- Linux Audio Conference 2004
- *LJ* Index—August 2004
- Who New? Qunu
- Linux 2.6-Based Box Set: SuSE 9.1 Professional
- They Said It

diff -u: What's New in Kernel Development

Zack Brown

Issue #124, August 2004

For more than a year, **Terence Ripperda** has been thinking about adding **Page Attribute Table** (PAT) support to Linux. Back in mid-2003, the issue did not seem so important to him, because only the AGP aperture and framebuffer really would benefit. With **PCI Express** systems coming out, however, the old workarounds are too slow. PCI Express lacks AGP's central aperture that can be marked WC (write-combined) in certain cases. Instead, individual memory pages must be marked as WC separately in the page tables. This cannot be done efficiently using the old ways, but by adding PAT support, a high level of efficiency still can be attained. Terence has been working closely with **Andi Kleen** to get something ready for inclusion in the kernel. Other folks, such as **Andy Whitcroft**, also have been looking into adding PAT support; so, one way or another, it seems that this enhancement will go through.

Maneesh Soni, acting on a tip from **Alexander Viro**, has taken on an annoying **sysfs** bug. Apparently, symbolic links in sysfs have been following the target that existed at the time of the link's creation no matter what, even when the intended target changes during use. A patch posted by Maneesh causes the target to be recalculated every time the link is read. Maintaining a consistent and proper sysfs interface is crucial to providing a clean global interface into the kernel. The procfs mishmash is one of the main reasons sysfs was created in the first place.

Carl-Daniel Hailfinger secretly has been implementing a successor for ATA RAID in 2.6, called raiddetect, and finally unveiled his work for comment in April 2004. Raiddetect is intended to identify vendor software RAID superblocks, verify their validity, group them by vendor and set them up for use. Although Carl-Daniel's work admittedly could have been done as an extension of **Wilfried Weissmann**'s **EVMS** plugin, Carl-Daniel opted for the easier method of starting from scratch—though he affirms that his work could be incorporated into EVMS even now if folks preferred that. So far, there doesn't seem to be much dissent, and people like **Jeff Garzik** are filled with praise for Carl-Daniel's work.

Marcelo Tosatti has decided to merge **Serial ATA** (SATA) support into the 2.4 tree—probably one of the last big features to go into 2.4. Marcelo says the reason for this is that a lot of new computer systems are shipping with SATA-only disks, and until 2.6 becomes fully stable it makes the best sense for 2.4 to be able to run on these new systems. The decision met with some criticism, most notably the counter-contention that the 2.6 tree is plenty stable and specifically includes the same security fixes that recently went into 2.4. Regardless, it seems that there is at least some good reason for including SATA support in 2.4, and Marcelo apparently fully intends to do so.

Herbert Xu has written some code to allow Software Suspend to be compiled as a module, but with all the hubbub over whether loadable modules should be unloadable at all, the question of whether a given feature should be compilable as a module is becoming less critical. Some developers, including **Alan Cox**, have been leaving modularity support out of their driver work, preferring to get the basic behavior correct and worry about less important modularity features later. Software Suspend is apparently one of these as well, and folks like **Pavel Machek** and **Nigel Cunningham** have expressed their doubts that a modular Software Suspend is more valuable than the nonmodular version. The whole question of modularity seems up in the air in the 2.6 series, because the ability to load and unload modules always has been dear to the hearts of many users, and its disappearance has been more for the convenience of developers than for users. If a clear solution to the problem of how to unload modules properly is found, it is likely that 2.7 would see the reappearance of that feature.

Linux Audio Conference 2004

Dave Phillips

Issue #124, August 2004

From April 29 through May 2, 2004, the Center for Arts and Media Technology (ZKM) in Karlsruhe, Germany, hosted the second International Linux Audio Conference. Developers from around the world presented open-source

software for hard-disk recording, software sound synthesis, music typesetting, digital audio signal processing and many other sound and music-related areas. Several concerts and a sound installation demonstrated how this software can be used for composition and production. Linux-based music and sound production hardware, the Lionstracs Mediastation and the Hartmann Neuron, also made an appearance. Visitors were able to set up their own computer-based Linux audio systems with the assistance of experts from the AGNULA/Demudi and Planet CCRMA distributions.

The conference showed considerable growth over last year's successful event. More than 30 topic presentations took place, and many extra sessions were held to accommodate requests for more information about particular topics. All presentations were given in English and were comprehensible and engaging, a testament to the linguistic abilities of the presenters, many of whom were not native English speakers. Highlight topics included Paul Davis' demonstration of recent advances in his Ardour digital audio workstation project, Steve Harris' exposition of the JAMin audio mastering suite, Fons Adriaensen's demo of his Aeolus pipe-organ emulator and Stefan Kersten's introduction to his work with the SuperCollider3 sound synthesis environment.

Other memorable presentations included Orm Finnendahl's explanation of his use of common UNIX tools such as sed and awk in his compositions, Ivica Ico Bukvic's thoughts on getting Linux into school music curricula and Dave Topper's report on his GAIA, a graphic front end for sound synthesis languages.

With four concerts, a dance, ongoing hardware demonstrations and various workshops demonstrating the use of Linux audio software, there was ample opportunity to hear how Linux audio software has evolved. Huge kudos to Frank Neumann, Matthias Nagorni and Goetz Dipper for keeping everything running smoothly. Vast thanks also to SuSE and ZKM for their support. See linux-sound.org for links to software, recordings and a conference program. Next year's conference already is scheduled for ZKM again.

LJ Index—August 2004

- 1. Thousands of PCs covered by a study of possible conversion to desktop Linux in Paris: 17
- 2. Thousands of Linux desktops planned for an Extremadura study in Spain: 300
- 3. Number of Linux seats at Modena in Italy: 750
- 4. Number of Linux seats at Brescia in Italy: 250
- 5. Number of Linux seats at Robur in Italy: 190

- 6. Minimum number of factors at which Google looks on a Web page to maximize accurate results: 100
- 7. Minimum billions of nodes in Google's matrix computation: 3
- 8. Minimum billions of edges in Google's matrix computation: 30
- 9. Millions of blogs watched by Technorati: 2.35
- 10. Millions of links tracked by Technorati: 304.73
- 11. Number of employees working for California Digital: 55
- 12. Number of four-processor Itanium 2 servers in the Linux-based Thunder supercomputer built by California Digital for Lawrence Livermore National Laboratory: 1,024
- 13. Trillions of operations per second Thunder can perform: 19.94
- 14. Position Thunder would occupy on the Top 500 list of leading supercomputers, if it had made the deadline for the latest list: 2
- 15. Maximum number of wireless, solar-powered parking payment stations running embedded Linux being rolled out in Montréal: 800
- 16. Number of parking meters each station will replace: 12
- 17. Minimum percentage by which Linux TCO (total cost of ownership) can be driven down: 10
- 18. Maximum percentage by which Linux TCO can be driven down: 40
- 19. Millions of results for a Google “Linux” search: 117
- 20. Millions of results for a Google “Windows” search: 122

- 1: Microcost, at Desktop Linux Summit
- 2, 3–5: David Orban, Questar.it
- 6–8: Nelson Minar, Google
- 9, 10: Technorati, May 13, 2004
- 11–14: CNet
- 15, 16: LinuxDevices
- 17, 18: Meta Group
- 19, 20: Google

Who New? Qunu

Doc Searls

Issue #124, August 2004

I always know when I get an e-mail from Murray Gray, because he writes to me from the future. He's in Australia, near the leading edge of tomorrow. I'm in California, which is yesterday, relatively speaking. Murray's subject, however is

instantaneous. He's into IM (instant messaging). And, here's what he's doing about it, with a new project called Qunu (www.qunu.com):

> Let's say you're stuck in GIMP with a layer problem or have issues installing a particular printer driver. What are the chances of getting instant support from someone who knows their stuff and actually wants to help? Pretty low. You could ask a buddy on your IM list, visit countless sites, forums or knowledge bases on the Net, or even go to the manufacturer's Web site. You may eventually find what you are looking for, but how long can it take sometimes to achieve resolution?
>
> Now, let's imagine an open IM system that allows you to connect immediately with the very person you need to talk to, who's on-line when you are, and who's passionate and knowledgeable in the area you're having problems. Qunu makes this possible though an innovative cross-platform SDK that integrates with any software application as well as a fully fledged standalone application.
>
> In short, it's community helping community. Anyone can download the software and ask for help, and likewise, anyone can provide help. Experts can receive direct requests for help or dive into the pool of requests and help with issues in which they're knowledgeable.
>
> Software publishers and IT companies now have an incredible opportunity to allow their dedicated, passionate users to evangelize on their behalf, and an even greater opportunity to connect with users and solve problems before they end up as PR nightmares.

Qunu traces back to an idea by Joseph E. Trent on the (now defunct) BeNews forum in January 2000. The idea piqued the interest of Helmar Rudolph (formerly of Opera Software and Sonork) and Murray, who joint-financed development by Justin Kirby of openaether.org using his in-house toolkit for XMPP/Jabber development.

Qunu is based on Jabber's XMPP (eXtensible Message and Presence Protocol) and Mozilla. Here's Murray's technical case:

> The core portability layer of Qunu is a thin wrapper around the Apache Portable Runtime (APR), which has been released under the APL in the spirit of the libraries it's directly built upon. The Jabber protocol then uses the APR wrapper for threads and networking portability. The XML parsing library that is used is the Xerces-c XML parser, whose maturity and feature set are truly astounding.

> Qunu is both a cross-platform SDK framework and standalone application, making it suitable for any type of implementation environment, software or topic area. For reasons of brevity, we have glossed over the details of setting up Jabber streams, creating XUL windows and handling events, but suffice it to say, the Mozilla application framework and Jabber protocols are complex and elegant, and Qunu has leveraged this into a useful and extensible framework.

Murray invites readers to download the code, play with it and contribute improvements back to the new community.

Linux 2.6-Based Box Set: SuSE 9.1 Professional

Don Marti

Issue #124, August 2004

If you've been reading about the 2.6 kernel in *Linux Journal* but waiting until a distribution integrates it nicely before you take the plunge, have a look at SuSE 9.1 Professional.

The install is nothing special, which is a good thing considering the current standard for high-end Linux distributions. A great feature is the check for any updated packages from the Net before letting you out on your own. If a security problem comes up, new SuSE 9.1 installs from CD will get the fix as part of the normal install.

KDE 3.2, the default desktop environment, is slick, with features including easy CD burning, a file manager that lets you drag and drop files anywhere you're allowed to sftp to and great GNU Privacy Guard integration in KMail.

The hard part about doing a Linux distribution, however, is hardware support, and this is where SuSE 9.1 really stands out. SaX, SuSE's X configuration tool, set up a multihead display and a Wacom tablet with point-and-click ease—no configuration or documentation reading required. USB printers and storage devices on our test system also simply work.

Although the price is higher than a lot of distributions, the box set includes good printed user and administrator guides, along with install support. We recommend this distribution for power users of other OSes who are trying Linux for the first time.

SuSE has a good record for laptop support, too, so watch for more on SuSE 9.1 from Doc Searls, who is running this distribution on his IBM ThinkPad.

They Said It

Just as in the 1990s the COTS sector caught up with the military sector in applications of cryptography, this decade will see the self-same overtaking but this time of traffic analysis. You do not need to examine content if you can deploy enough sensors and make sense of their findings.

—Dan Geer, on the Politech mailing list

Brazil hosted the recent Fourth International Forum on Free Software, held in the World Social Forum's stronghold of Porto Alegre. There, jazzy pop star Gilberto Gil, now Brazil's minister of culture, promised to “tropicalize digitalization”, presumably a reference to bridging the developed and developing worlds. If Extremadura is the harbinger, those tropicalized digits will be globally connected, fiercely patriotic, and free as sunshine.

—Bruce Sterling in *Wired*

Many on Jack Valenti's side of the divide treasure their creative freedom and fight like dogs against any who would block it. They would never dream of permitting a system in which every film had to be approved by the state, but they are advocating a system in which every program has to be approved by the state, because a lot of them think that all programs come either from faceless corporations like Microsoft or from criminal vandals.

We software creators need to insist that creative applies to us.

—Joe Buck, in a comment on lessig.org

Nat [Friedman] used to say “If you write a thousand lines of code, you are violating someone's patent today.”

The picture is not pretty for anyone in the software industry.

But this is similar to what happens to biology students: in their first four semesters as they learn about all the dangers, infections, vectors for infections and bacteria, they stop eating everything, they start washing their hands with special products, they double clean their utensils, they wash their fruits ten times a day.

Two years later they are eating food with their bare hands again.

—Miguel de Icaza, primates.ximian.com/~miguel

Archive Index Issue Table of Contents

Advanced search

On the Web

Tune-up for IT at Doc's Garage

Heather Mead

Issue #124, August 2004

Information technology power is flowing out from Linux-using smart companies. Join the conversation.

Senior Editor Doc Searls has spent a good portion of the last year or so talking to IT guys inside some big-name companies about how and where they are using Linux and open source. More than a single story, Doc's tapped in to a revolution in the way IT departments work. He's dubbed the revolution DIY-IT. Response to his reports has been overwhelming, so we've launched a new SSC Web site devoted to the DIY-IT movement. Here's Doc describing in his own words what this new site is:

> Early last year, Don Marti assigned me to write a long piece about “how Linux helps make smart companies smarter”. What I found was there's a lot more going on out there than anybody's talking about in the mainstream press, or pretty much anywhere. It was nothing less than a vast reform movement inside IT (information technology) organizations everywhere—one in which the demand side was starting to supply itself. Much of this has been with Linux and other open-source tools and applications, but the phenomenon goes far beyond that. It's a cultural as well as a technical revolution, yet it's also very practical. It's about getting stuff done.
>
> So, we decided to start a site on the Web where people involved in what we call DIY-IT—Do It Yourself IT—could talk about their work and report on trends happening in the field. We call it Doc Searls' IT Garage (garage.docsearls.com) or just IT Garage: a place for “News, ideas and real-world stories about how IT folks solve their own problems”. If it takes off, maybe it will become a print magazine, but we don't know yet. We're still shaking the thing down and signing up regular contributors.

If your company is part of the revolution or you'd like it to be, check out the site. See what you can learn and take back to the rest of the team. Or, if you've got your own story to share, write about it on the site.

Back on the *Linux Journal* site, after constructing this year's Ultimate Linux Box, Don Marti decided he needed to write about building the ultimate quiet Linux box. His article “This Linux Box Is Too Loud!” (www.linuxjournal.com/article/7601) is a roundup of computer-silencing techniques that work under Linux.

Finally, if you appreciated this month's article “Linux Serial Consoles for Servers and Clusters”, be sure to read Poul E. J. Petersen's Web article, “Project Hydra: the USB Multiheaded Monster” (www.linuxjournal.com/article/6518). Petersen writes, “It would be really cool to have a dedicated, remotely accessible console server with a lot of serial ports connected to all of our servers.” Read about how they accomplished exactly that using “a readily available USB bus with USB-to-serial adapters”.

As always, if you or your company has a cool new project or has found a better way of handling everyday tasks, send me an article proposal at info@linuxjournal.com.

Heather Mead is senior editor of *Linux Journal*.

Archive Index Issue Table of Contents

Advanced search

Best of Technical Support

Our experts answer your technical questions.

Changing Red Hat's Firewall Level

I'm not able to change the security level on either Red Hat 7.2 or 9. It is always high on both versions and on all three computers on which I have installed these distributions. I have installed all recent attempts since the first install of 7.2 without Firewall. Firewall still installs and is at High. No one else has had this problem when I read over installs of 7.2. What am I doing so wrong?

—
Jeff Douglass

jdouglas25@yahoo.com

If you want to change graphically using env, you can click on Start Here from the desktop, then System Settings and then Security Level. If you are not running as root, enter root's password and you can change security levels. I believe something similar is offered during the installation of Red Hat.

—
Usman S. Ansari

uansari@yahoo.com

The firewall portion of Red Hat's installer is a bit confusing. Most importantly, in the Customize section the Trusted Devices options truly are trusted, allowing any and all traffic on them. When I first looked at that I assumed if I wanted to allow incoming SSH on eth0, I would click SSH on Allow incoming as well as selecting eth0 under trusted—not the case. This gives blanket permissions on all ports on the selected trusted device.

I'm not sure what went wrong during the installation, but you can change the configuration afterward by running lokkit to reconfigure the firewall. There also

is a GUI-based utility that does the same thing called redhat-config-securitylevel. Run `/etc/init.d/iptables restart` after making changes.

—
Timothy Hamlin

thamlin@nmt.edu

Bear in mind that it is not a good idea to operate a system with a relaxed or disabled security, especially if linked to the Internet. You should learn to configure the firewall to let through the traffic you need but no more.

—
Felipe Barousse Boué

fbarousse@piensa.com

Mouse Pointer in VNC?

How can I change the mouse pointer to a big white cursor when displayed through a VNC viewer?

—
Marcos Machado

pimentamac@hotmail.com

Currently there are several flavors of VNC-based utilities, with many enhancements and differences among each them. A method you can try is to change the mouse configuration of your account locally (including the cursor or pointer size) with a tool like gnome-mouse-properties. Then, later on when you establish a remote session through VNC, you will get a larger cursor, again, depending on the VNC client and server you are using.

—
Felipe Barousse Boué

fbarousse@piensa.com

Distributing One POP Account to Multiple Users

We have registered a domain and one POP3 e-mail account with our registrar. Unfortunately, our DSL ISP (Earthlink) does not permit us to have SMTP port 25 open to send and receive mail directly. All outbound e-mail must be sent to our ISP's servers, and then they relay them onwards.

We have a small network consisting of six users. All users must see the same e-mail, thus one POP3 mail account for all. I have Postfix configured to send our outbound e-mail via the ISP without any problems. I have been playing with fetchmail to retrieve our inbound e-mail from our remote POP3 account but have not had any luck getting the e-mail distributed to our local users on our network. fetchmail polls and downloads the mail no problem, but when it hits our Postfix server it says:

```
X-Fetchmail-Warning: recipient address myaddress@earthlink.net didn't match any local name
```

I have tried to configure aliases using Webmin with success. I guess the problem is with multidrop distribution.

—
Walter

trance_fool@hotmail.com

Keep things simple. Either get several mail accounts on your ISP's servers—one per user and configure their workstations to log in to their POP accounts at the ISP's server, or arrange for open SMTP and POP or IMAP ports to your server. That way, it will be much easier for you to manage your e-mail without adding complexity to an already difficult-to-manage service (e-mail). You don't want to complicate your life when you have to filter spam, viruses and all that crap while having a home-crafted solution as you are describing in your post.

—
Felipe Barousse Boué

fbarousse@piensa.com

You don't need to use fetchmail multidrop if you want all six users to get copies of the same mail from the POP account. Just make an “all” alias in /etc/aliases, which you can do with Webmin, then configure fetchmail to deliver to “all” via SMTP:

```
poll pop.example.net:
   user joe there has password secr3t
   is all here
```

Postfix will do the rest.

—
Don Marti

info@linuxjournal.com

Adding a Nonstandard Kernel Module

I wanted to update my kernel to include a module that isn't provided by default. First, I thought I'd try building the kernel identical to what Red Hat provided. I've built Linux (a few years ago) without a problem, but when I tried to build the Red Hat configuration, copied from the configs subdirectory, it failed during the make modules step. The errors don't make sense to me. There's about 1,200 lines of errors generated. Why doesn't it compile right out of the box?

—
Chris Carlson

cwcarlson@cox.net

First, you do not have to compile the kernel to add a new module. You simply can compile with the header files from the running kernel, and it should work fine. As far as your problem with kernel compilation is concerned, I think you are missing the `make oldconfig` step, which would read the config file you mention. By the way, did you remember to rename it to .config?

—
Usman S. Ansari

uansari@yahoo.com

ADSL under Knoppix?

What is the easiest way to install an ADSL Internet connection using Knoppix?

—
Andrew Catchpole

krubby@hotmail.com

That really depends on the kind of ADSL modem you have and on the actual settings of your ISP's service. This page may be of help: www.rhapsodyk.net/adsl/HOWTO and this one too: christophe.delord.free.fr/en/adsl/debian.html.

—
Felipe Barousse Boué

fbarousse@piensa.com

Upgrading from Red Hat to SuSE?

I have tried to upgrade Red Hat 9 to SuSE 9.0 without success. Can this be done? Or does one need to reinstall the system?

—
L W Randerson

luthrw@att.net

You are trying to upgrade a system installed from one vendor of Linux distribution to another vendor. It is impossible that this will work. Many times upgrades from the same vendor have problems. I suggest that you start from scratch: repartition and make new filesystems. If you have enough disk space, you can have both SuSE and Red Hat installed at same time on different partitions.

—
Usman S. Ansari

uansari@yahoo.com

Perhaps it would be possible with a lot (a whole lot) of hacking, but generally, you don't want to upgrade across different distributions. Red Hat to Red Hat should work, and SuSE to SuSE, but the layouts are different, and it would be terrifically complicated. Back up all the user files you have, and do a fresh install rather than attempt an update.

—
Timothy Hamlin

thamlin@nmt.edu

New Products

Verari QuatreX-64, Paracel BLAST 1.6, Sangoma AFT Cards and more.

Verari QuatreX-64

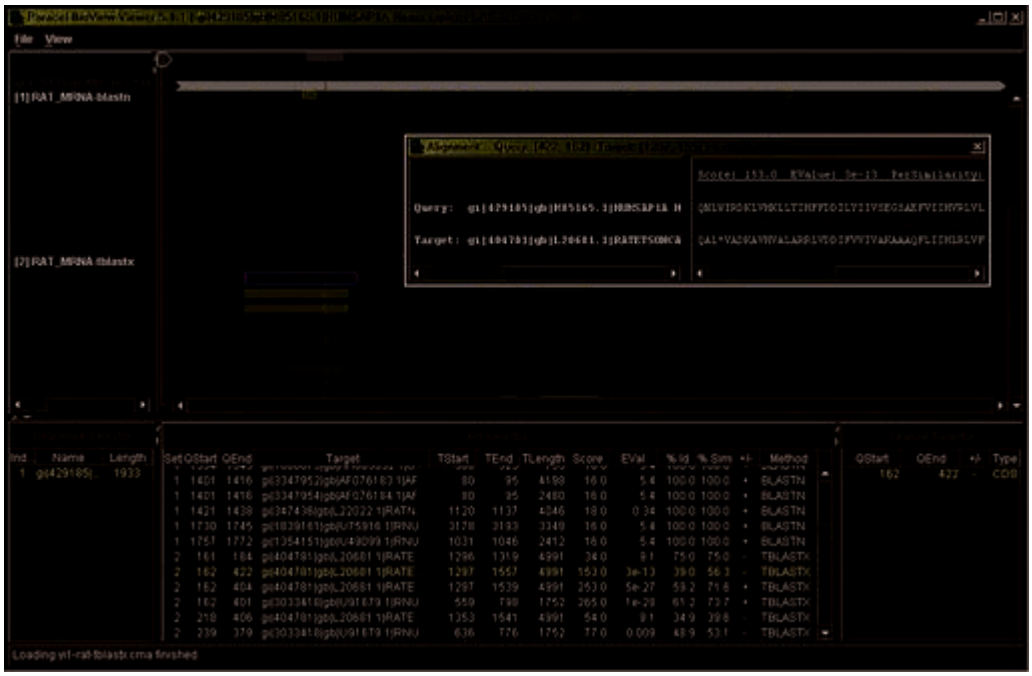
Verari Systems, formerly RackSaver, introduced the QuatreX-64, a four-way server powered by the AMD Opteron 850 processor. On SPEC CPU2000 benchmarks, the QuatreX-64 reached a CINT2000 rate of 63.4 base and 68.5 peak, as well as a CFP2000 rate of 47.2 base and 50.5 peak. A platform-independent server, the QuatreX-64 features up to 32GB of DDR RAM. Up to four SCSI hard drives with optional removable drive bays are included for storage needs. The server also uses AMD's Direct Connect Architecture to reduce bottlenecks by connecting I/O directly to the CPU and connecting CPUs to one another.

Verari Systems, Inc., 9449 Carroll Park Drive, San Diego, California 92121, 858-874-3800, www.verari.com.

Paracel BLAST 1.6

Version 1.6 of Paracel BLAST, an enhanced version of NCBI BLAST software re-engineered specifically for large-scale cluster systems, offers native support for the Opteron 64-bit processor. Designed for pharmaceutical and research institutions, Paracel BLAST 1.6 automatically handles query packing, database splitting and distribution of tasks among processors. In conjunction with Paracel BlastMachine2, a Linux-based cluster that runs on Opteron or Xeon processors, Paracel BLAST 1.6 can perform large-scale analyses rapidly.

Paracel, Inc., 1055 East Colorado Boulevard, Fifth Floor, Pasadena, California 91106, 888-727-2235, www.paracel.com.



Sangoma AFT Cards

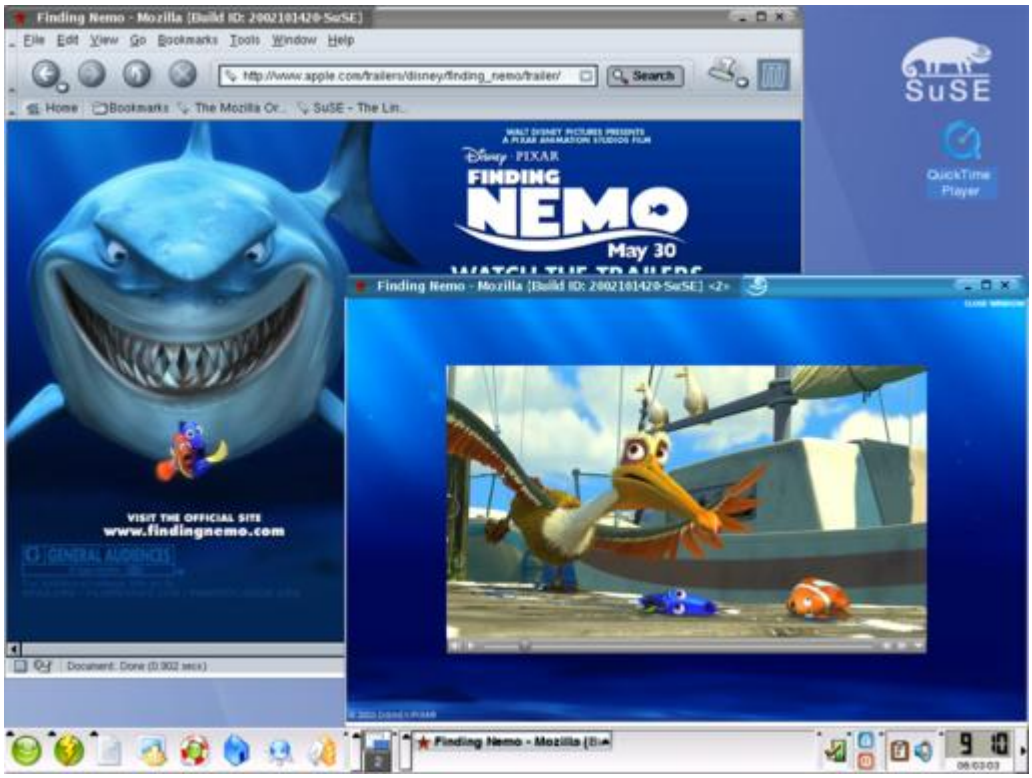
Sangoma Technologies' new line of Advanced Flexible Telecommunications (AFT) cards are designed to be easily upgradable to support different telco interfaces and line speeds. The AFT cards are based on programmable hardware technology that uses field programmable gate arrays (FPGAs) to handle line protocols in hardware. An accompanying line of self-configuring drivers automates the process of line and protocol setup. The first card to be released is the 2U form factor A101, measuring 4.7" × 2.2". It comes with a T1/E1/J1 interface and an optional 3.3v/5v PCI card that supports full channelization of the DS0s. Future upgrades of the A101 will have hardware-based ATM and SS7 MTP2 support.

Sangoma Technologies Corporation, 50 McIntosh Drive, Suite 120, Markham, Ontario L3R 9T3, CANADA, 800-388-2475, www.sangoma.com.

CrossOver Office 3.0

CrossOver Office allows users to install Windows applications and plugins in Linux, without needing a Microsoft operating system license. New for version 3.0 is Linux support for Lotus Notes, MS Outlook XP and MS Project. Also new for the 3.0 release is CrossOver Office Standard, a version for home users. CrossOver Office Professional is designed for corporate use and offers enhancements for enterprise-level deployment as well as the ability to run in shared multi-user mode. CrossOver Plugin, CodeWeaver's browser plugin application, now is integrated into all CrossOver products.

CodeWeavers, Inc., 2356 University Avenue West, Suite 420, St. Paul, Minnesota 55114, 651-523-9300, www.codeweavers.com.



InstallShield X

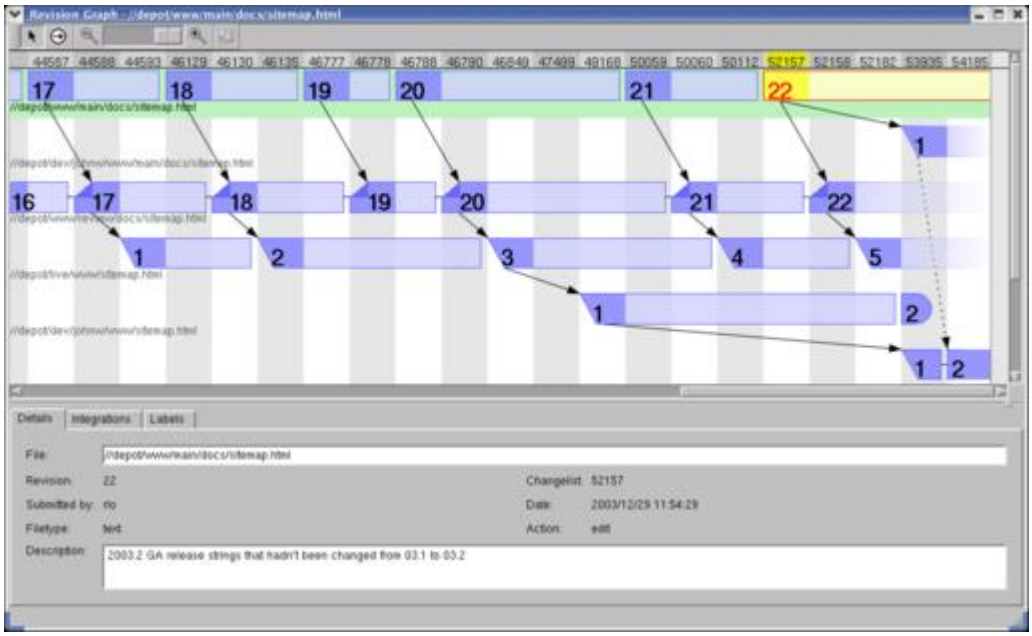
InstallShield X is installation authoring software that enables software application developers to write industry-standard installations targeting almost any platform, OS and device. This new offering combines former products InstallShield DevStudio and InstallShield MultiPlatform and adds new functionality for handling deployments. InstallShield X also includes Update Service Starter Edition, which adds updating and user messaging services for managing software life cycles. Other new InstallShield X features include SQL server and IIS Web services support, plus enhanced mobile device support for creating standalone device installations.

InstallShield Software Corporation, 900 National Parkway, Suite 125, Schaumburg, Illinois 60173, 800-374-4353, www.installshield.com.

Perforce SCM 2004.1

Perforce Software released version 2004.1 of its Software Configuration Management (SCM) system. Perforce SCM 2004.1 tracks and manages software development in both small and large work settings. It handles distributed development and supports developers across more than 50 OSes. The two main new visualization features are a revision graph representing branch history and a folder compare feature. The revision graph feature displays a tree-style graph outlining the branching history of a specific file, including all points, edits and merges. The folder compare displays two folders side by side in expandable tree views, allowing users to compare folder structure and file content. Both components are part of the Perforce Visual Client, a cross-platform interface for Linux, Mac OS X and Windows.

Perforce Software, Inc., 2320 Blanding Avenue, Alameda, California 94501, 510-864-7400, www.perforce.com.



Archive Index Issue Table of Contents

Advanced search